ORACLE

State of the Dolphin

MySQL 8.0

Frédéric Descamps
Community Manager
MySQL
October 2020

Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purpose only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied up in making purchasing decisions. The development, release, timing and pricing of any features or functionality described for Oracle´s product may change and remains at the sole discretion of Oracle Corporation.

Who am I ?

about.me/lefred

@lefred

Frédéric Descamps

- @lefred
- MySQL Evangelist
- Managing MySQL since 3.20
- devops believer
- living in Belgium 🇧🇪
- https://lefred.be

Where are we in 2020 ?

MySQL 8.0.22

@lefred

DB-Engines 2020

Database Ranking

Rank			DBMS	Database Model	Oct 2020
Oct 2020	Sep 2020	Oct 2019			
1.	1.	1.	Oracle	Relational, Multi-model	1368.77
2.	2.	2.	MySQL	Relational, Multi-model	1256.38
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model	1043.12
4.	4.	4.	PostgreSQL	Relational, Multi-model	542.40
5.	5.	5.	MongoDB	Document, Multi-model	448.02

DB-Engines 2020

Database Ranking

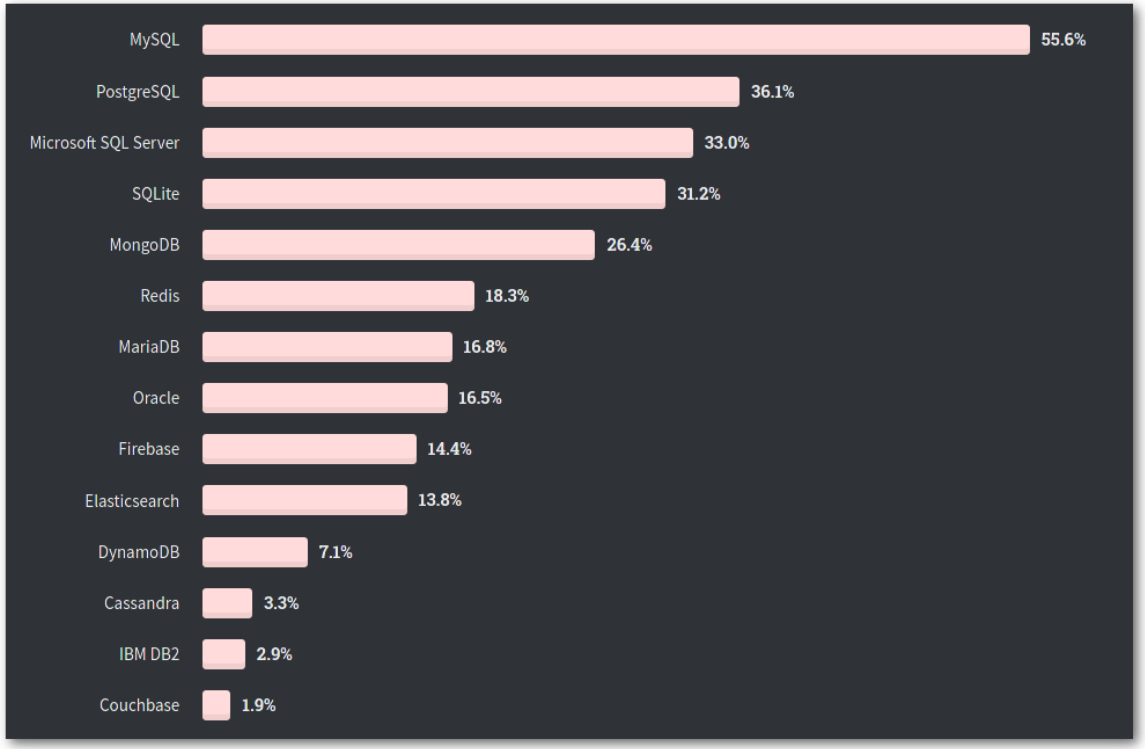
Rank Oct 2020	Rank Sep 2020	Rank Oct 2019	DBMS	Database Model	Oct 2020
1.	1.	1.	Oracle	Relational, Multi-model	1368.77
2.	2.	2.	MySQL	Relational, Multi-model	1256.38
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model	1043.12
4.	4.	4.	PostgreSQL	Relational, Multi-model	542.40
5.	5.	5.	MongoDB	Document, Multi-model	448.02

MySQL is the most popular Open Source database

MySQL is the DBMS of the Year 2019 !

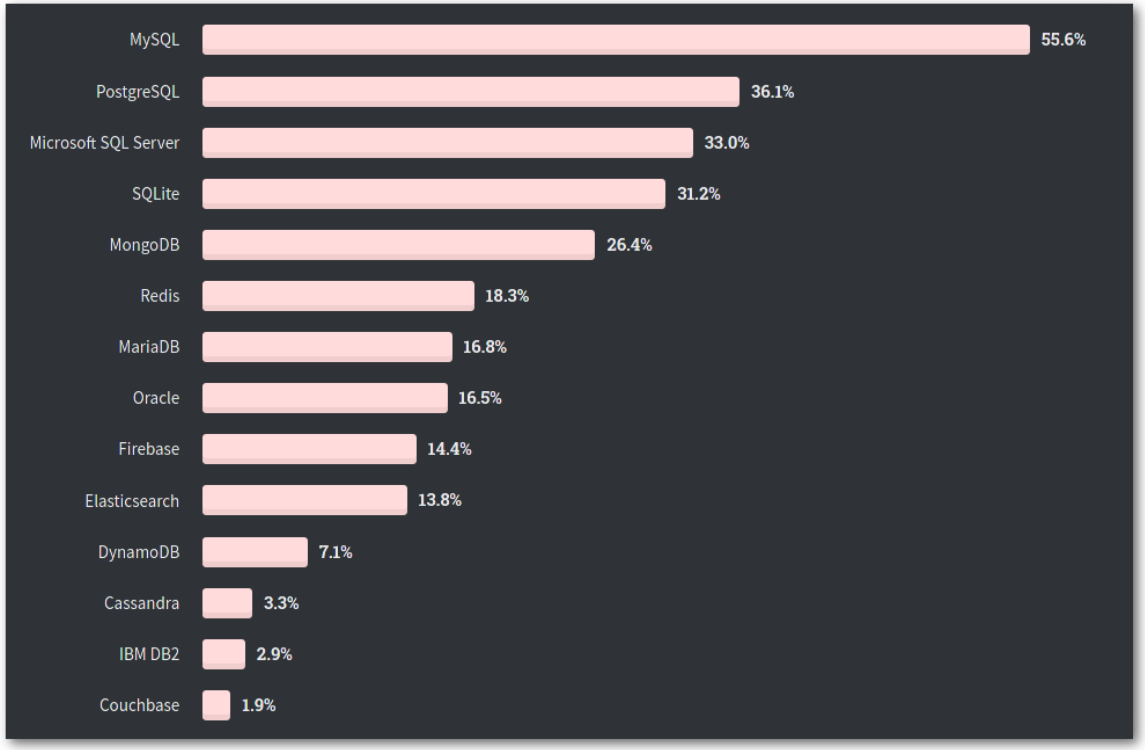
MySQL Developer Popularity

Stack Overflow Developer Survey 2020



MySQL Developer Popularity

Stack Overflow Developer Survey 2020



MySQL is the most popular database with developers

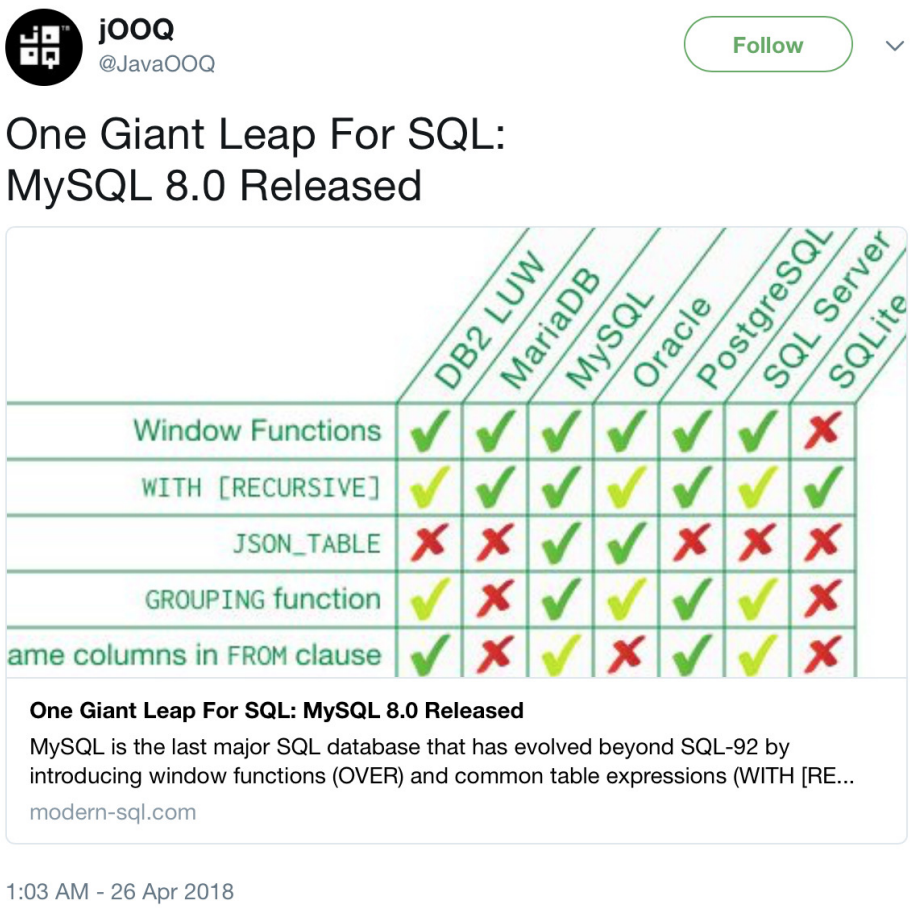
Happy 25th Anniversary MySQL

One giant leap for SQL

MySQL 8.0

@lefred

MySQL 8.0: one giant leap for SQL



"This is a landmark release as MySQL eventually evolved beyond SQL-92 and the purely relational dogma. Among a few other standard SQL features, MySQL now supports window functions (over) and common table expressions (with). Without a doubt, these are the two most important post-SQL-92 features."

and still innovating !

Credits: @MarkusWinand - @ModernSQL

	Db2 (LUW)	BigQuery	MariaDB	MySQL	Oracle DB	PostgreSQL	SQL Server	SQLite
with on top-level	✓	✓	✓	✓	✓	✓	✓	✓
with in subqueries	✗	✓[0]	✓[0]	✓	✓	✓	✗	✓

[0] Seems like CTE in subquery cannot see global CTEs.

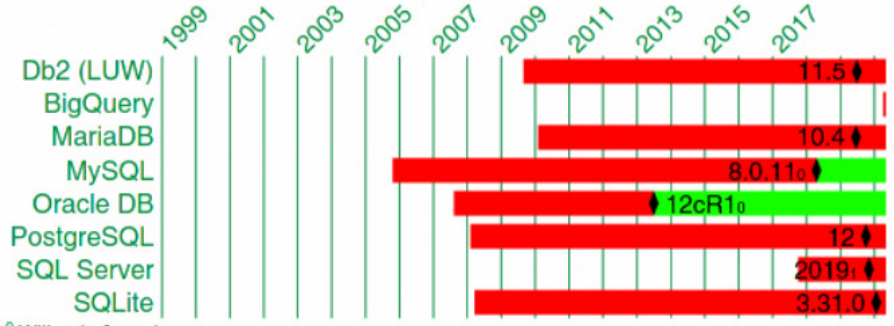
	Db2 (LUW)	BigQuery	MariaDB	MySQL	Oracle DB	PostgreSQL	SQL Server	SQLite
with on top-level	✓	✓[0]	✓	✓	✓	✓	✓	✓
with in subqueries	✗	✓[1]	✓[2]	✓	✓	✓	✗	✓
insert … with … select	✓	✓[0]	✓	✓	✓	✓[3]	✗[3]	✓[3]
with masks schema objects	✓	✓	✓	✓	✓	✓	✓	✓
with doesn't imply recursive	✗	✓	✓	✓	✗	✓	✗	✗
views bypass with	✓	✓	✓	✓	✓	✓	✓	✗
qualified names bypass with	✗	✓	✓	✓	✗	✓	✓	✓

[0] Without column names: WITH name AS (SELECT…)
[1] Without column names: WITH name AS (SELECT…) CTE in subquery cannot see global CTEs.
[2] CTE in subquery cannot see global CTEs.
[3] Supports proprietary syntax with … insert … select

	Db2 (LUW)	BigQuery	MariaDB	MySQL	Oracle DB	PostgreSQL	SQL Server	SQLite
Base table PRIMARY KEY	✗	[0]	[0]	✓	✗	✓	✗	[0]
Base table UNIQUE	✗	[0]	[0]	✓	✗	✗	✗	[0]
Joined tables	✗	[0]	[0]	✓	✗	✓[1]	✗	[0]
WHERE clause	✗	✗	[0]	✓	✗	✗	✗	[0]
GROUP BY clause	✗	✗	[0]	✓	✗	✗	✗	[0]

[0] No proper group by handling (per default).
[1] Not following joins to PRIMARY KEYs or UNIQUE constraints

json_table



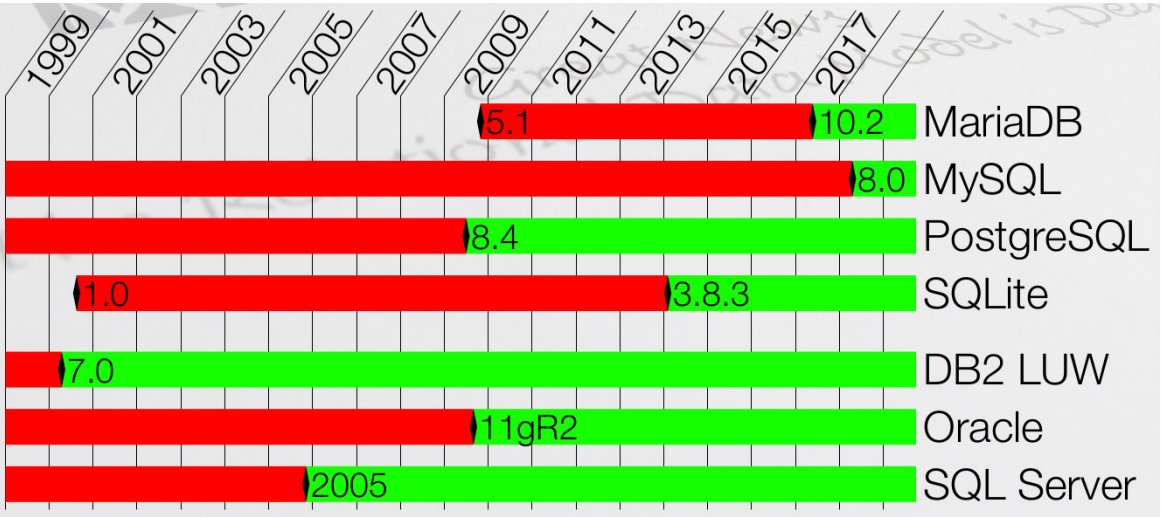
[0] Without plan clause.
[1] Use OPENJSON

	Db2 (LUW)	BigQuery	MariaDB	MySQL	Oracle DB	PostgreSQL	SQL Server	SQLite
ulti-row insert … values	✓[0]	✓[0]	✓[0]	✓	✗	✓[0]	✓[0]	✓[0]
Stand-alone values	✓[0]	✗	✓[0]	✓[1]	✗	✓[0]	✗	✓[0]
rom|Join (values …) t	✓[0]	✗	✓[0]	✓[1]	✗	✓[0]	✓[2]	✓[0]
With t as (values …)	✓[3]	✗	✓[0]	✓[1]	✗	✓[0]	✗	✓[0]
[not] in (values …)	✓[0]	✗	✓[4]	✓[1]	✗	✓[0]	✗	✓[0]

[0] Only without keyword row.
[1] Requires keyword row: values row('r1c1','r1c2'), row('r2c1', 'r2c2')
[2] Needs from clause column renaming. Only without keyword row.
[3] Requires column names in with clause. Only without keyword row.
[4] Column references not supported. Only without keyword row.

SQL: RECURSION / CTEs

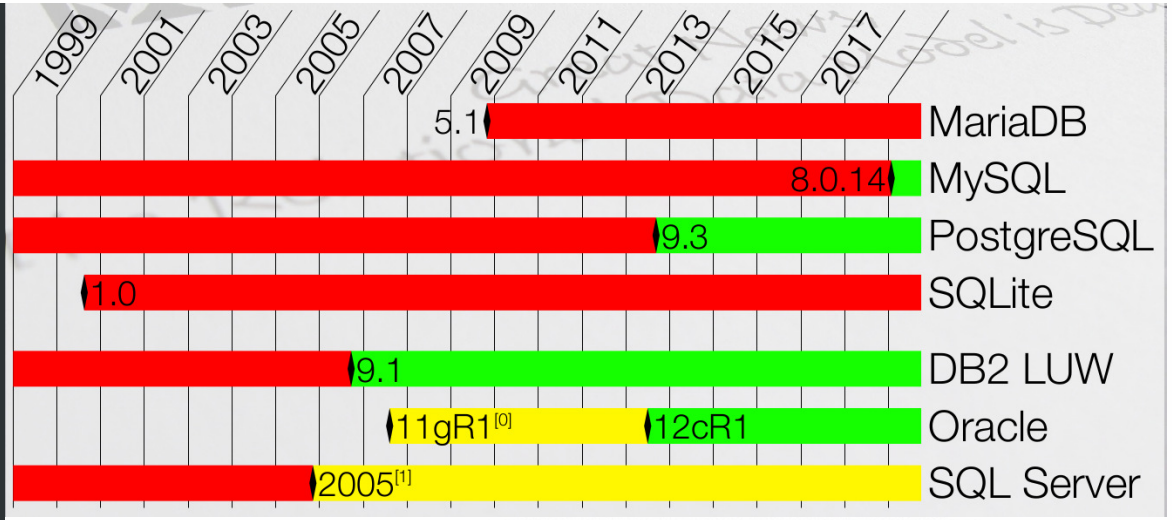
```
MySQL  localhost:33060+  test  2019-10-25 09:35:38
SQL  WITH RECURSIVE prev(id, parent) AS (
       SELECT t.id, t.parent FROM parents AS t
       WHERE t.id = 2
     UNION ALL
       SELECT t.id, t.parent FROM parents AS t
       JOIN prev ON t.parent = prev.id
     )
     SELECT * FROM prev;
+----+--------+
| id | parent |
+----+--------+
|  2 |      1 |
|  4 |      2 |
|  5 |      2 |
|  7 |      4 |
+----+--------+
4 rows in set (0.0011 sec)
```



Credits: @MarkusWinand - @ModernSQL

SQL: LATERAL

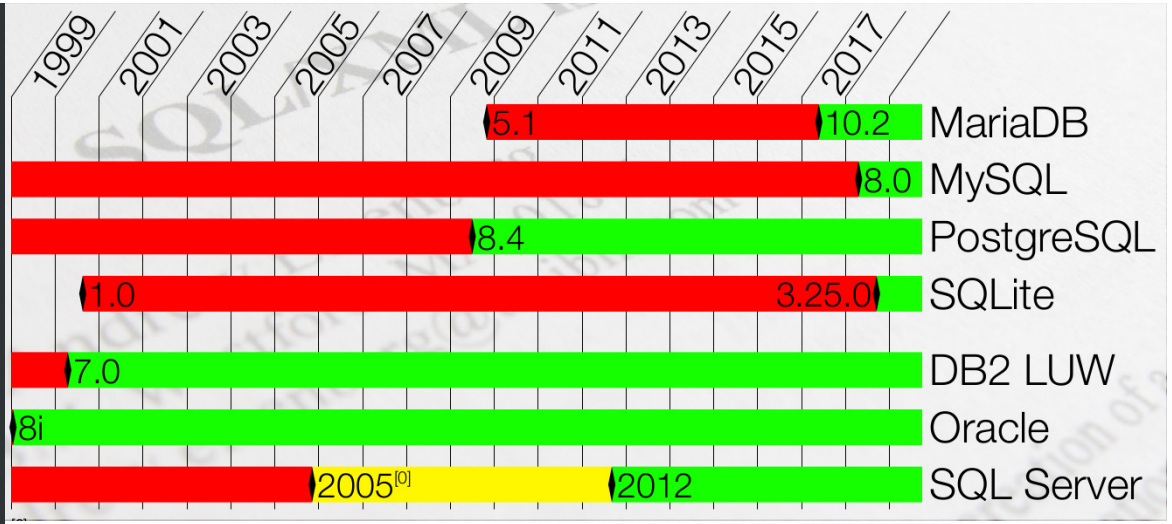
```
MySQL  localhost:33060+  world  2019-10-25 12:07:23
SQL  SELECT dt.population, dt.name, c.countrycode
       FROM (
        SELECT DISTINCT countrycode FROM city
       ) AS c,
       LATERAL (
        SELECT name, population FROM city
         WHERE city.countrycode = c.countrycode
         ORDER BY population desc LIMIT 1
       ) AS dt
     ORDER BY population desc LIMIT 5;
+------------+-----------------+-------------+
| population | name            | countrycode |
+------------+-----------------+-------------+
|   10500000 | Mumbai (Bombay) | IND         |
|    9981619 | Seoul           | KOR         |
|    9968485 | São Paulo       | BRA         |
|    9696300 | Shanghai        | CHN         |
|    9604900 | Jakarta         | IDN         |
+------------+-----------------+-------------+
5 rows in set (0.0122 sec)
```



Credits: @MarkusWinand - @ModernSQL

SQL: Analytical / Window Functions

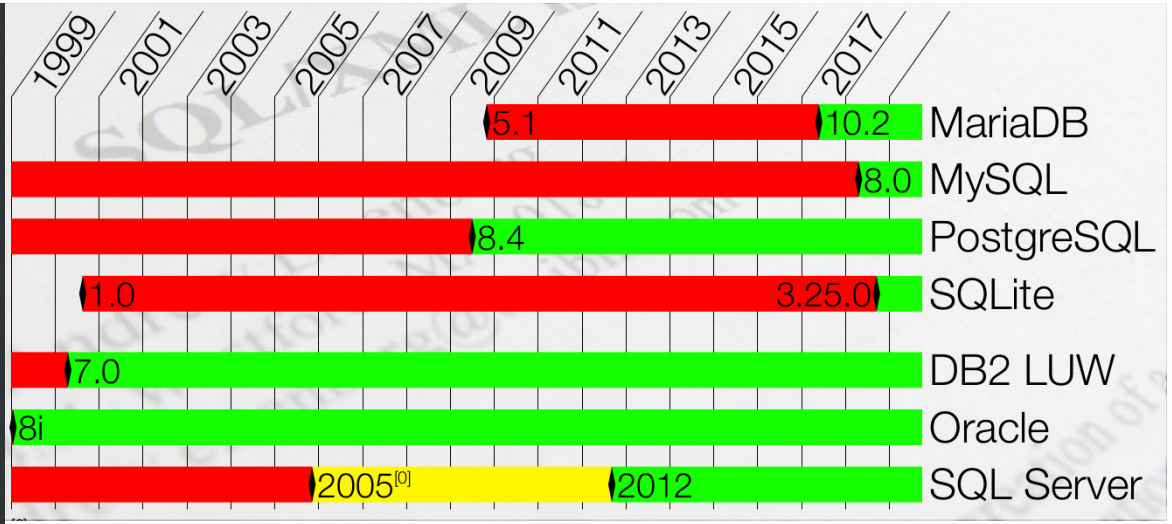
```
MySQL  localhost:33060+  world  2019-10-25 12:20:15
SQL  SELECT id, name, population,
            SUM(population) OVER (
              ORDER BY id
               ROWS BETWEEN
                    UNBOUNDED PRECEDING
                AND CURRENT ROW ) pop
     FROM city WHERE countrycode = 'BEL';
+-----+--------------------+------------+---------+
| id  | name               | population | pop     |
+-----+--------------------+------------+---------+
| 175 | Antwerpen          |     446525 |  446525 |
| 176 | Gent               |     224180 |  670705 |
| 177 | Charleroi          |     200827 |  871532 |
| 178 | Liège              |     185639 | 1057171 |
| 179 | Bruxelles [Brussel] |     133859 | 1191030 |
| 180 | Brugge             |     116246 | 1307276 |
| 181 | Schaerbeek         |     105692 | 1412968 |
| 182 | Namur              |     105419 | 1518387 |
| 183 | Mons               |      90935 | 1609322 |
+-----+--------------------+------------+---------+
9 rows in set (0.0013 sec)
```



Credits: @MarkusWinand - @ModernSQL

SQL: Analytical / Window Functions

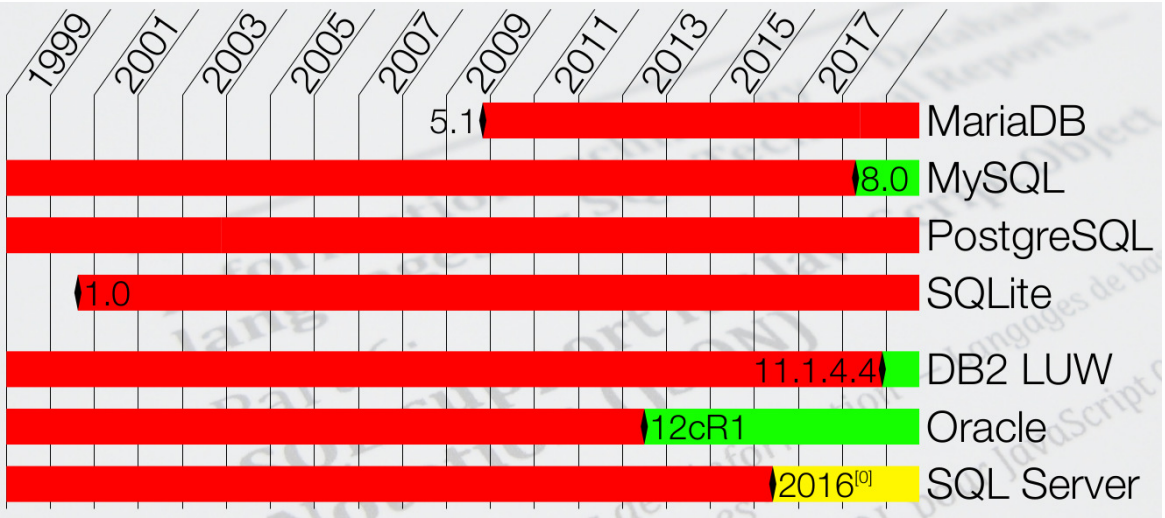
```
MySQL  localhost:33060+  world  2019-10-25 12:20:15
SQL  SELECT id, name, population,
            SUM(population) OVER (
              ORDER BY id
              ROWS BETWEEN
                UNBOUNDED PRECEDING
              AND CURRENT ROW ) pop
     FROM city WHERE countrycode = 'BEL';
+-----+---------------------+------------+---------+
| id  | name                | population | pop     |
+-----+---------------------+------------+---------+
| 175 | Antwerpen           |     446525 |  446525 |
| 176 | Gent                |     224180 |  670705 |
| 177 | Charleroi           |     200827 |  871532 |
| 178 | Liège               |     185639 | 1057171 |
| 179 | Bruxelles [Brussel] |     133859 | 1191030 |
| 180 | Brugge              |     116246 | 1307276 |
| 181 | Schaerbeek          |     105692 | 1412968 |
| 182 | Namur               |     105419 | 1518387 |
| 183 | Mons                |      90935 | 1609322 |
+-----+---------------------+------------+---------+
9 rows in set (0.0013 sec)
```



Credits: @MarkusWinand - @ModernSQL

SQL: JSON_TABLE

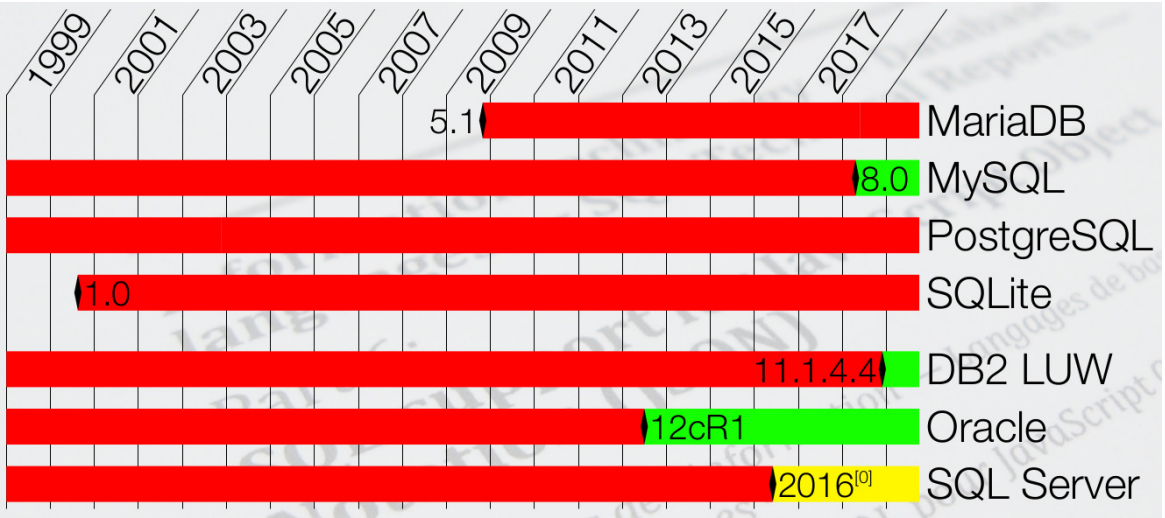
```
MySQL  localhost:33060+  docstore  2019-10-25 13:05:28
SQL  SELECT JSON_PRETTY(doc) FROM all_recs LIMIT 1\G
*************************** 1. row ***************************
JSON_PRETTY(doc): {
  "_id": "5ad5b645f88c5bb8fe3fd337",
  "name": "Morris Park Bake Shop",
  "grades": "",
  "address": {
    "coord": [
      -73.856077,
      40.848447
    ],
    "street": "Morris Park Ave",
    "zipcode": "10462",
    "building": "1007"
  },
  "borough": "Bronx",
  "cuisine": "Bakery",
  "restaurant_id": "30075445"
}
```



Credits: @MarkusWinand - @ModernSQL

SQL: JSON_TABLE

```
MySQL  localhost:33060+  docstore  2019-10-25 13:07:34
SQL  SELECT name, cuisine FROM all_recs,
       JSON_TABLE(doc, "$"
         COLUMNS(
           name VARCHAR(30) PATH "$.name",
           cuisine VARCHAR(20) PATH "$.cuisine")
       ) AS r
     LIMIT 5;
+-------------------------------+---------------+
| name                          | cuisine       |
+-------------------------------+---------------+
| Morris Park Bake Shop         | Bakery        |
| Wendy'S                       | Hamburgers    |
| Dj Reynolds Pub And Restaurant | Irish         |
| Riviera Caterer               | American      |
| Tov Kosher Kitchen            | Jewish/Kosher |
+-------------------------------+---------------+
5 rows in set (0.0006 sec)
```



Credits: @MarkusWinand - @ModernSQL

SQL: JSON_VALUE NEW in 8.0.21

```
MySQL  localhost:33060+  docstore  2020-10-07 01:00:30
SQL  SELECT JSON_VALUE(doc, '$.address.zipcode' RETURNING UNSIGNED)
     FROM restaurants LIMIT 3;
+-------------------------------------------------------+
| JSON_VALUE(doc, '$.address.zipcode' RETURNING UNSIGNED) |
+-------------------------------------------------------+
|                                                  NULL |
|                                                 10462 |
|                                                 11225 |
+-------------------------------------------------------+
3 rows in set (0.0004 sec)
MySQL  localhost:33060+  docstore  2020-10-07 01:01:38
SQL
```

This function is described in SQL 2016, chapter 6.27 and is also implemented in:

- Oracle
- SQL Server
- DB2

MySQL Document Store

I ❤️ JSON & NoSQL

@lefred

MySQL supports JSON & CRUD operations

* CREATE
* READ
* UPDATE
* DELETE

- `col->add({title: 'MySQL is Great', author: 'lefred' })`
- `col->find()`
- `collection.modify('author = "lefred"').set('author', 'kenny')`
- `col.remove('author = "lefred"')`

MySQL supports JSON & CRUD operations

* CREATE
* READ
* UPDATE
* DELETE

- `col->add({title: 'MySQL is Great', author: 'lefred' })`
- `col->find()`
- `collection.modify('author = "lefred"').set('author', 'kenny')`
- `col.remove('author = "lefred"')`

It's possible to use MySQL without a single line of SQL !

MySQL supports JSON & CRUD operations

```
MySQL 127.0.0.1:33060 JS restaurants.find("cuisine='French' AND borough!='Manhattan'").
fields(["name","cuisine","borough"]).limit(2)
[
    {
        "borough": "Queens",
        "cuisine": "French",
        "name": "La Baraka Restaurant"
    },
    {
        "borough": "Queens",
        "cuisine": "French",
        "name": "Air France Lounge"
    }
]
2 documents in set (0.00 sec)
```

NoSQL + SQL = MySQL

MySQL 8.0 is also refactoring

New Volcano Iterator

@lefred

MySQL New Iterator Executor

- Modular
- Easy to Extend
- Each iterator encapsulates one operation
- Same interface for all iterators
- All operations can be connected

MySQL New Iterator Executor

- Modular
- Easy to Extend
- Each iterator encapsulates one operation
- Same interface for all iterators
- All operations can be connected

EXPLAIN FORMAT=TREE

```
MySQL  localhost:33060+  world  2019-10-25 15:57:06
SQL  EXPLAIN FORMAT=tree
    -> select dt.population, dt.name, c.name from
    ->   (select distinct countrycode, name from city) as c,
    ->    LATERAL (
    ->    select name, population from city
    ->    where city.countrycode = c.countrycode
    ->    order by population desc limit 1) as dt\G
*************************** 1. row ***************************
EXPLAIN: -> Nested loop inner join
    -> Invalidate materialized tables (row from c)
        -> Table scan on c
            -> Materialize
                -> Table scan on <temporary>
                    -> Temporary table with deduplication
                        -> Table scan on city  (cost=425.05 rows=4188)
    -> Table scan on dt
        -> Materialize (invalidate on row from c)
            -> Limit: 1 row(s)
                -> Sort: city.Population DESC, limit input to 1 row(s) per chunk  (cost=6.32 rows=18)
                    -> Index lookup on city using CountryCode (CountryCode=c.countrycode)

1 row in set, 1 warning (0.0015 sec)
Note (code 1276): Field or reference 'c.countrycode' of SELECT #3 was resolved in SELECT #1
```

EXPLAIN ANALYZE

- Instruments and executes the query
  - Estimated cost
  - Actual execution statistics
    - Time to return first row
    - Time to return all rows
    - Number of rows returned
    - Number of loops
- Uses the new tree output format also available in EXPLAIN

EXPLAIN ANALYZE

```
MySQL  localhost:33060+  world  2019-10-25 16:04:40
SQL  EXPLAIN ANALYZE
    -> select dt.population, dt.name, c.name from
    ->   (select distinct countrycode, name from city) as c,
    ->    LATERAL (
    ->     select name, population from city
    ->     where city.countrycode = c.countrycode
    ->     order by population desc limit 1) as dt\G
*************************** 1. row ***************************
EXPLAIN: -> Nested loop inner join  (actual time=7.238..567.236 rows=4056 loops=1)
    -> Invalidate materialized tables (row from c)  (actual time=7.164..7.996 rows=4056 loops=1)
        -> Table scan on c  (actual time=0.001..0.386 rows=4056 loops=1)
            -> Materialize  (actual time=7.163..7.762 rows=4056 loops=1)
                -> Table scan on <temporary>  (actual time=0.001..0.424 rows=4056 loops=1)
                    -> Temporary table with deduplication  (actual time=5.331..6.274 rows=4056 loops=1)
                        -> Table scan on city  (cost=425.05 rows=4188) (actual time=0.076..2.289 rows=4079 loops=1)
    -> Table scan on dt  (actual time=0.000..0.000 rows=1 loops=4056)
        -> Materialize (invalidate on row from c)  (actual time=0.138..0.138 rows=1 loops=4056)
            -> Limit: 1 row(s)  (actual time=0.126..0.126 rows=1 loops=4056)
                -> Sort: city.Population DESC, limit input to 1 row(s) per chunk  (cost=6.32 rows=18) (actual time=0.126..0.126 rows=1 loops=4056)
                    -> Index lookup on city using CountryCode (CountryCode=c.countrycode)  (actual time=0.008..0.106 rows=150 loops=4056)

1 row in set, 1 warning (0.5687 sec)
```

EXPLAIN ANALYZE

```
MySQL  localhost:33060+  world  2019-10-25 16:04:40
SQL  EXPLAIN ANALYZE
    -> select dt.population, dt.name, c.name from
    ->   (select distinct countrycode, name from city) as c,
    ->    LATERAL (
    ->     select name, population from city
    ->     where city.countrycode = c.countrycode
    ->     order by population desc limit 1) as dt\G
*************************** 1. row ***************************
EXPLAIN: -> Nested loop inner join  (actual time=7.238..567.236 rows=4056 loops=1)
    -> Invalidate materialized tables (row from c)  (actual time=7.164..7.996 rows=4056 loops=1)
        -> Table scan on c  (actual time=0.001..0.386 rows=4056 loops=1)
            -> Materialize  (actual time=7.163..7.762 rows=4056 loops=1)
                -> Table scan on <temporary>  (actual time=0.001..0.424 rows=4056 loops=1)
                    -> Temporary table with deduplication  (actual time=5.331..6.274 rows=4056 loops=1)
                        -> Table scan on city  (cost=425.05 rows=4188) (actual time=0.076..2.289 rows=4079 loops=1)
    -> Table scan on dt  (actual time=0.000..0.000 rows=1 loops=4056)
        -> Materialize (invalidate on row from c)  (actual time=0.138..0.138 rows=1 loops=4056)
            -> Limit: 1 row(s)  (actual time=0.126..0.126 rows=1 loops=4056)
                -> Sort: city.Population DESC, limit input to 1 row(s) per chunk  (cost=6.32 rows=18) (actual time=0.126..0.126 rows=1 loops=4056)
                    -> Index lookup on city using CountryCode (CountryCode=c.countrycode)  (actual time=0.008..0.106 rows=150 loops=4056)

1 row in set, 1 warning (0.5687 sec)
```

Actual time to get the first row

EXPLAIN ANALYZE

```
MySQL  localhost:33060+  world  2019-10-25 16:04:40
SQL  EXPLAIN ANALYZE
    -> select dt.population, dt.name, c.name from
    ->   (select distinct countrycode, name from city) as c,
    ->    LATERAL (
    ->     select name, population from city
    ->     where city.countrycode = c.countrycode
    ->     order by population desc limit 1) as dt\G
*************************** 1. row ***************************
EXPLAIN: -> Nested loop inner join  (actual time=7.238..567.236 rows=4056 loops=1)
    -> Invalidate materialized tables (row from c)  (actual time=7.164..7.996 rows=4056 loops=1)
        -> Table scan on c  (actual time=0.001..0.386 rows=4056 loops=1)
            -> Materialize  (actual time=7.163..7.762 rows=4056 loops=1)
                -> Table scan on <temporary>  (actual time=0.001..0.424 rows=4056 loops=1)
                    -> Temporary table with deduplication  (actual time=5.331..6.274 rows=4056 loops=1)
                        -> Table scan on city  (cost=425.05 rows=4188) (actual time=0.076..2.289 rows=4079 loops=1)
    -> Table scan on dt  (actual time=0.000..0.000 rows=1 loops=4056)
        -> Materialize (invalidate on row from c)  (actual time=0.138..0.138 rows=1 loops=4056)
            -> Limit: 1 row(s)  (actual time=0.126..0.126 rows=1 loops=4056)
                -> Sort: city.Population DESC, limit input to 1 row(s) per chunk  (cost=6.32 rows=18) (actual time=0.126..0.126 rows=1 loops=4056)
                    -> Index lookup on city using CountryCode (CountryCode=c.countrycode)  (actual time=0.008..0.106 rows=150 loops=4056)

1 row in set, 1 warning (0.5687 sec)
```

Actual time to get all rows

EXPLAIN ANALYZE

```
MySQL  localhost:33060+  world  2019-10-25 16:04:40
SQL  EXPLAIN ANALYZE
    -> select dt.population, dt.name, c.name from
    ->   (select distinct countrycode, name from city) as c,
    ->    LATERAL (
    ->     select name, population from city
    ->     where city.countrycode = c.countrycode
    ->     order by population desc limit 1) as dt\G
*************************** 1. row ***************************
EXPLAIN: -> Nested loop inner join  (actual time=7.238..567.236 rows=4056 loops=1)
    -> Invalidate materialized tables (row from c)  (actual time=7.164..7.996 rows=4056 loops=1)
        -> Table scan on c  (actual time=0.001..0.386 rows=4056 loops=1)
            -> Materialize  (actual time=7.163..7.762 rows=4056 loops=1)
                -> Table scan on <temporary>  (actual time=0.001..0.424 rows=4056 loops=1)
                    -> Temporary table with deduplication  (actual time=5.331..6.274 rows=4056 loops=1)
                        -> Table scan on city  (cost=425.05 rows=4188) (actual time=0.076..2.289 rows=4079 loops=1)
    -> Table scan on dt  (actual time=0.000..0.000 rows=1 loops=4056)
        -> Materialize (invalidate on row from c)  (actual time=0.138..0.138 rows=1 loops=4056)
            -> Limit: 1 row(s)  (actual time=0.126..0.126 rows=1 loops=4056)
                -> Sort: city.Population DESC, limit input to 1 row(s) per chunk  (cost=6.32 rows=18) (actual time=0.126..0.126 rows=1 loops=4056)
                    -> Index lookup on city using CountryCode (CountryCode=c.countrycode)  (actual time=0.008..0.106 rows=150 loops=4056)

1 row in set, 1 warning (0.5687 sec)
```

Actual number of rows read

EXPLAIN ANALYZE

```
MySQL  localhost:33060+  world  2019-10-25 16:04:40
SQL  EXPLAIN ANALYZE
    -> select dt.population, dt.name, c.name from
    ->   (select distinct countrycode, name from city) as c,
    ->    LATERAL (
    ->     select name, population from city
    ->     where city.countrycode = c.countrycode
    ->     order by population desc limit 1) as dt\G
*************************** 1. row ***************************
EXPLAIN: -> Nested loop inner join  (actual time=7.238..567.236 rows=4056 loops=1)
    -> Invalidate materialized tables (row from c)  (actual time=7.164..7.996 rows=4056 loops=1)
        -> Table scan on c  (actual time=0.001..0.386 rows=4056 loops=1)
            -> Materialize  (actual time=7.163..7.762 rows=4056 loops=1)
                -> Table scan on <temporary>  (actual time=0.001..0.424 rows=4056 loops=1)
                    -> Temporary table with deduplication  (actual time=5.331..6.274 rows=4056 loops=1)
                        -> Table scan on city  (cost=425.05 rows=4188) (actual time=0.076..2.289 rows=4079 loops=1)
    -> Table scan on dt  (actual time=0.000..0.000 rows=1 loops=4056)
        -> Materialize (invalidate on row from c)  (actual time=0.138..0.138 rows=1 loops=4056)
            -> Limit: 1 row(s)  (actual time=0.126..0.126 rows=1 loops=4056)
                -> Sort: city.Population DESC, limit input to 1 row(s) per chunk  (cost=6.32 rows=18) (actual time=0.126..0.126 rows=1 loops=4056)
                    -> Index lookup on city using CountryCode (CountryCode=c.countrycode)  (actual time=0.008..0.106 rows=150 loops=4056)

1 row in set, 1 warning (0.5687 sec)
```

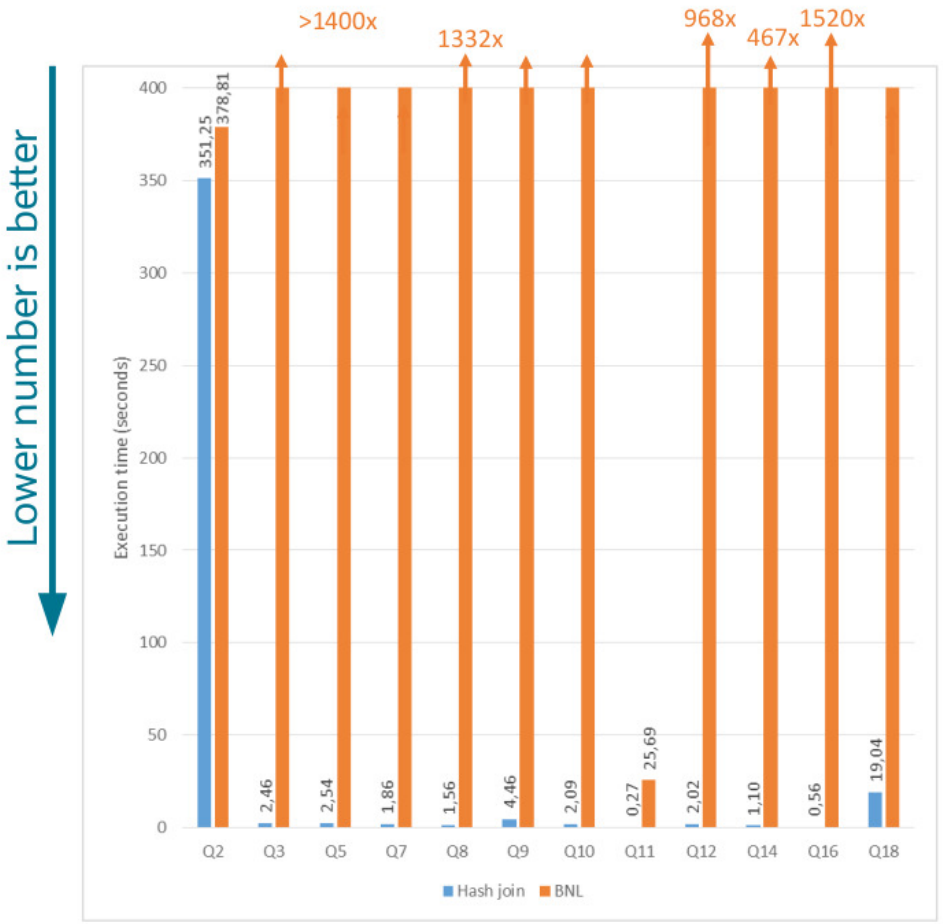
Actual number of loops

Hash Join

- Typically faster than nested loop for large result sets
- In-memory if possible
- Spill to disk if necessary
- Used in all types of joins (`inner, equi, outer, semi, anti`)
- Replaces BNL in query plans

Hash Join: performance

- **BNL compared to hash join**
- **Force BNL/hash join in DBT-3/TPC-H**
  - DBT-3/TPC-H without indexes
  - Optimizer selects BNL
  - Automatic conversion to hash join
- **Hash join is much faster than BNL**
- **Can't expect same improvement when indexes are available**



Performance & Scalability

InnoDB & Binary Logs - latest changes

@lefred

Redo logging can now be enabled and disable

Very useful during initial data load !

```
mysql>  ALTER INSTANCE DISABLE INNODB REDO_LOG;
```

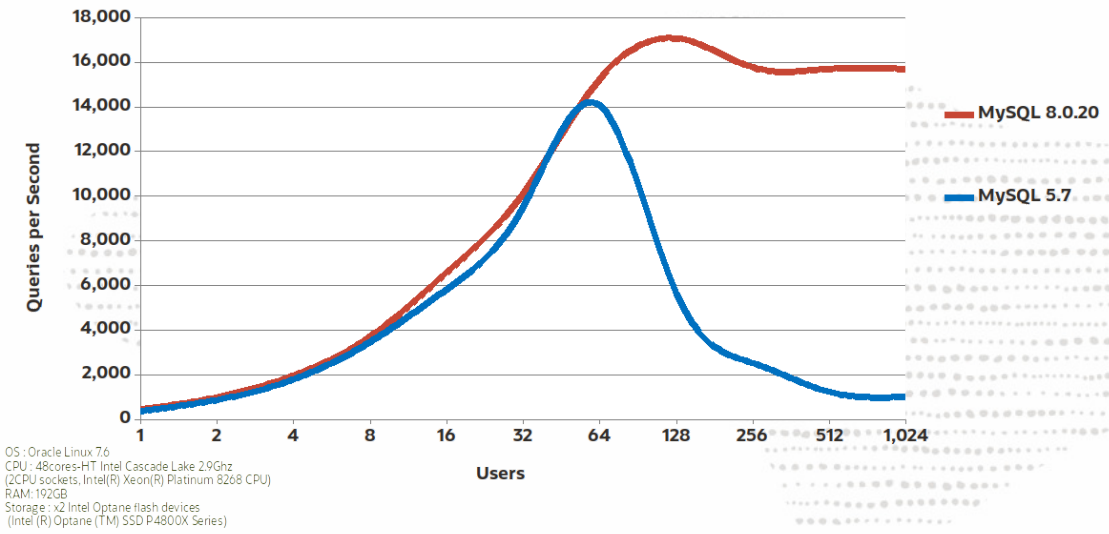
Redo logging can now be enabled and disable

Very useful during initial data load !

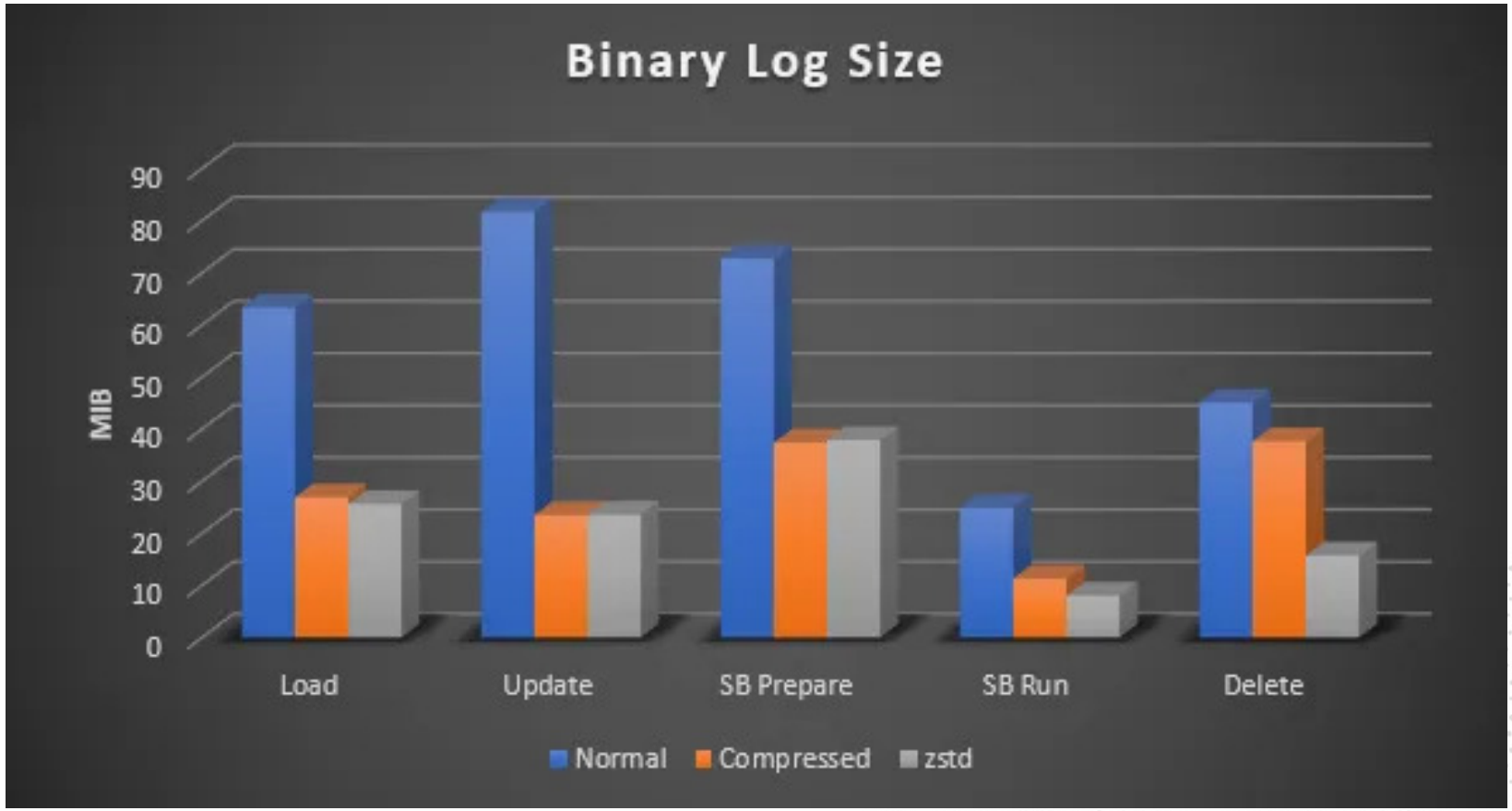
```
mysql>  ALTER INSTANCE DISABLE INNODB REDO_LOG;
```

New Double Write Buffer 15x Faster !!

MySQL 8.0: Sysbench **IO-bound** OLTP_RW, 8 tables, 50M rows each

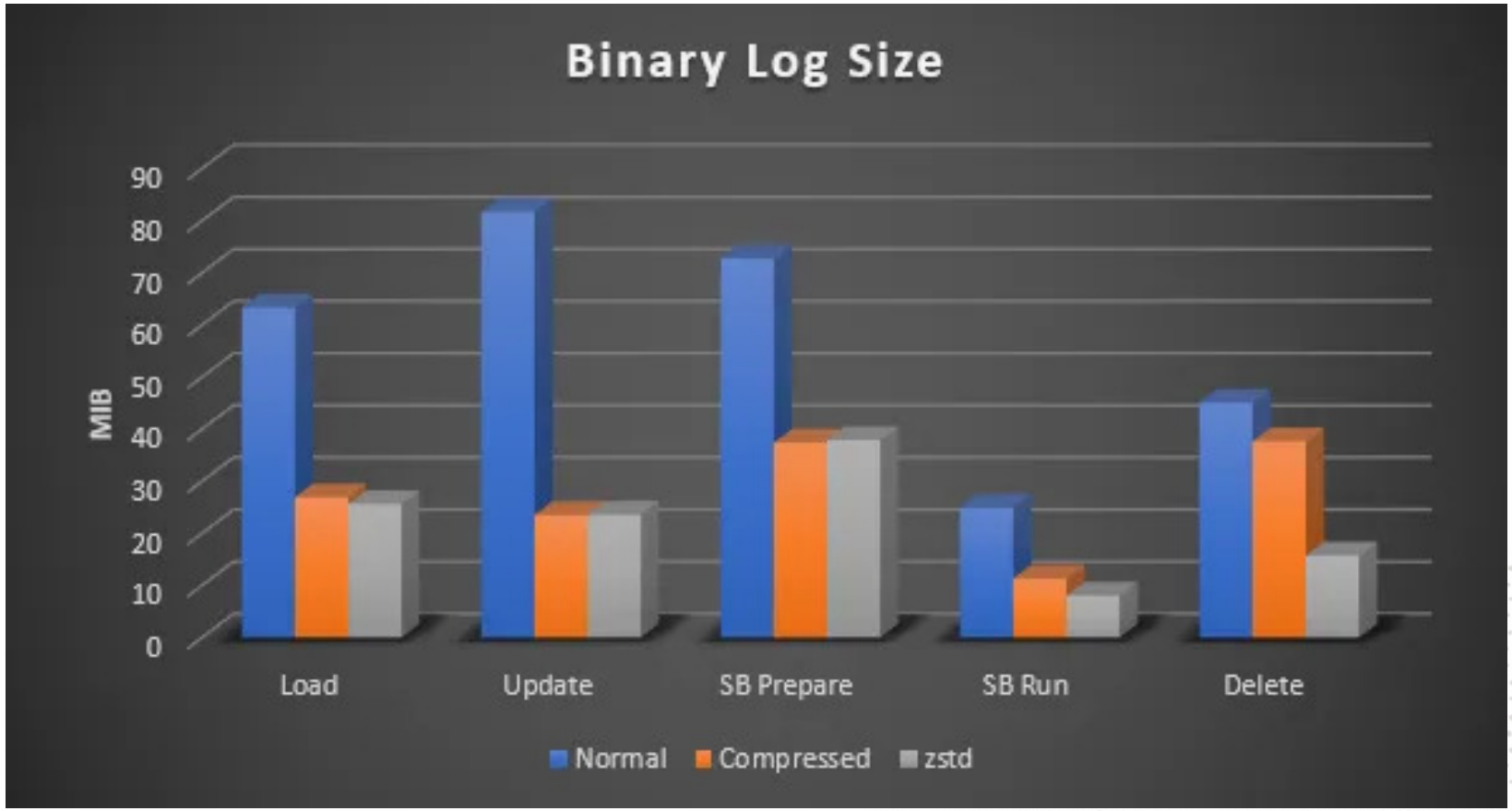


New Binlog Compression !



Credits: @JWKrogh - https://mysql.wisborg.dk/2020/05/07/mysql-compressed-binary-logs/

New Binlog Compression !



Credits: @JWKrogh - https://mysql.wisborg.dk/2020/05/07/mysql-compressed-binary-logs/

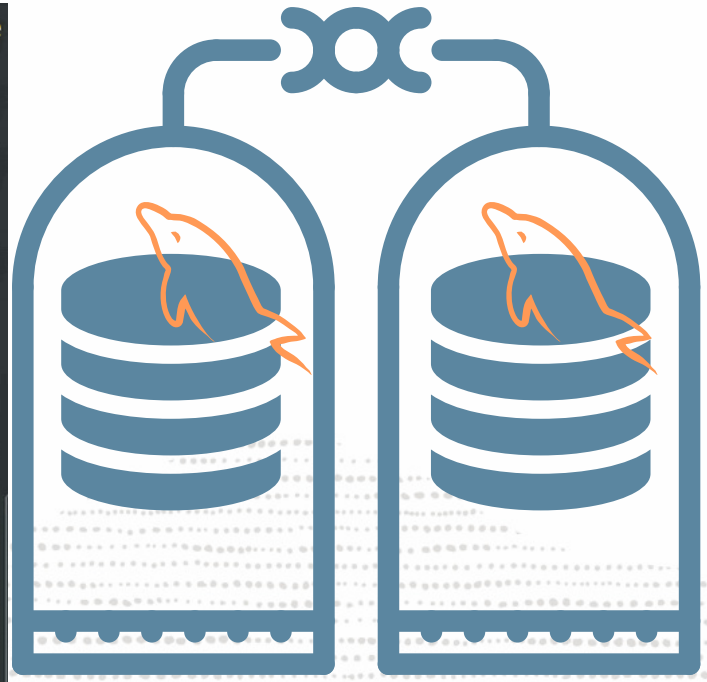
and of course MySQL Clone

```
Monitoring recovery process of the new cluster member. Press ^C to stop monitoring and let it continue
 in background.
Clone based state recovery is now in progress.

NOTE: A server restart is expected to happen as part of the clone process. If the
server does not support the RESTART command or does not come back after a
while, you may need to manually start it back.

* Waiting for clone to finish...
NOTE: mysql3:3306 is being cloned from mysql2:3306
** Stage DROP DATA: Completed
** Clone Transfer
    FILE COPY  ##################################################  100%  Completed
    PAGE COPY  ##################################################  100%  Completed
    REDO COPY  ##################################################  100%  Completed
** Stage RECOVERY: \
NOTE: mysql3:3306 is shutting down...

* Waiting for server restart... ready
* mysql3:3306 has restarted, waiting for clone to finish...
* Clone process has finished: 63.07 MB transferred in about 1 second (~inf TB/s)
```



Usability

always improving usability !

@lefred

Error log available in performance_schema !

```
mysql> select * from error_log where subsystem='InnoDB';
+----------------------------+-----------+--------+------------+-----------+-----------------------------------+
| LOGGED                     | THREAD_ID | PRIO   | ERROR_CODE | SUBSYSTEM | DATA                              |
+----------------------------+-----------+--------+------------+-----------+-----------------------------------+
| 2020-09-28 22:50:36.381707 |         1 | System | MY-013576  | InnoDB    | InnoDB initialization has started. |
| 2020-09-28 22:50:37.230615 |         1 | System | MY-013577  | InnoDB    | InnoDB initialization has ended.   |
| 2020-10-06 15:22:32.645623 |         1 | System | MY-013576  | InnoDB    | InnoDB initialization has started. |
| 2020-10-06 15:22:34.209989 |         1 | System | MY-013577  | InnoDB    | InnoDB initialization has ended.   |
| 2020-10-08 23:24:49.568095 |         1 | System | MY-013576  | InnoDB    | InnoDB initialization has started. |
| 2020-10-08 23:24:50.782234 |         1 | System | MY-013577  | InnoDB    | InnoDB initialization has ended.   |
+----------------------------+-----------+--------+------------+-----------+-----------------------------------+
6 rows in set (0.00 sec)
```

Error log available in performance_schema !

```
mysql> select * from error_log where subsystem='InnoDB';
+----------------------------+-----------+--------+------------+-----------+-----------------------------------+
| LOGGED                     | THREAD_ID | PRIO   | ERROR_CODE | SUBSYSTEM | DATA                              |
+----------------------------+-----------+--------+------------+-----------+-----------------------------------+
| 2020-09-28 22:50:36.381707 |         1 | System | MY-013576  | InnoDB    | InnoDB initialization has started. |
| 2020-09-28 22:50:37.230615 |         1 | System | MY-013577  | InnoDB    | InnoDB initialization has ended.   |
| 2020-10-06 15:22:32.645623 |         1 | System | MY-013576  | InnoDB    | InnoDB initialization has started. |
| 2020-10-06 15:22:34.209989 |         1 | System | MY-013577  | InnoDB    | InnoDB initialization has ended.   |
| 2020-10-08 23:24:49.568095 |         1 | System | MY-013576  | InnoDB    | InnoDB initialization has started. |
| 2020-10-08 23:24:50.782234 |         1 | System | MY-013577  | InnoDB    | InnoDB initialization has ended.   |
+----------------------------+-----------+--------+------------+-----------+-----------------------------------+
6 rows in set (0.00 sec)
```

```
mysql> select * from error_log where subsystem='server' order by logged desc limit 1\G
*************************** 1. row ***************************
    LOGGED: 2020-10-08 23:24:51.791041
 THREAD_ID: 0
      PRIO: System
ERROR_CODE: MY-010931
 SUBSYSTEM: Server
      DATA: /home/fred/opt/mysql/8.0.22/bin/mysqld: ready for connections. Version: '8.0.22'  socket: '/tmp/mysql_sandbox21223.sock'
  port: 21223  MySQL Community Server - GPL.
1 row in set (0.00 sec)
```

SOURCE_CONNECTION_AUTO_FAILOVER

With MySQL 8.0.22 there is new asynchronous connection failover mechanism to automatically establish an asynchronous (source to replica) replication connection to a new source after the existing connection from a replica to its source fails.

```
mysql> SELECT asynchronous_connection_failover_add_source('', '127.0.0.1', 21223, '', 80);
+-------------------------------------------------------------------------------+
| asynchronous_connection_failover_add_source('', '127.0.0.1', 21223, '', 80)  |
+-------------------------------------------------------------------------------+
| The UDF asynchronous_connection_failover_add_source() executed successfully. |
+-------------------------------------------------------------------------------+
1 row in set (0.18 sec)

mysql> SELECT asynchronous_connection_failover_add_source('', '127.0.0.1', 21224, '', 20);
+-------------------------------------------------------------------------------+
| asynchronous_connection_failover_add_source('', '127.0.0.1', 21224, '', 20)  |
+-------------------------------------------------------------------------------+
| The UDF asynchronous_connection_failover_add_source() executed successfully. |
+-------------------------------------------------------------------------------+
1 row in set (0.06 sec)

mysql> select * from mysql.replication_asynchronous_connection_failover;
+--------------+-----------+-------+-------------------+--------+
| Channel_name | Host      | Port  | Network_namespace | Weight |
+--------------+-----------+-------+-------------------+--------+
|              | 127.0.0.1 | 21223 |                   |     80 |
|              | 127.0.0.1 | 21224 |                   |     20 |
+--------------+-----------+-------+-------------------+--------+
2 rows in set (0.00 sec)
```

Solutions Easy to Deploy

Database Architectures

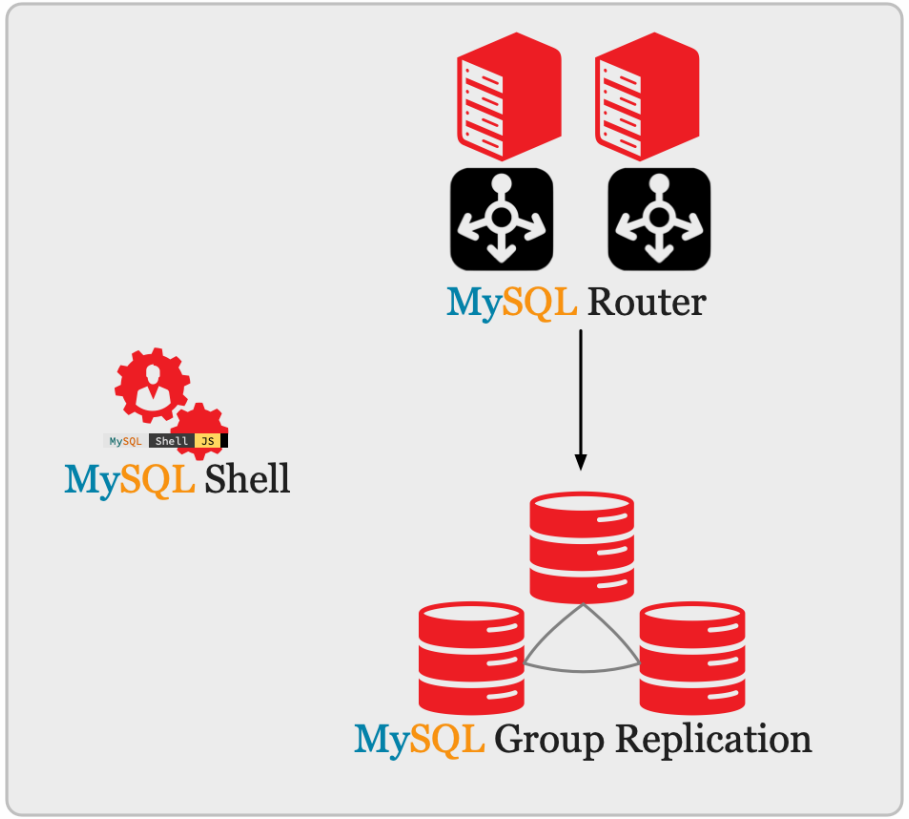
@lefred

MySQL InnoDB Cluster

"A single product — MySQL — with **high availability** and **scaling features** baked in; providing an **integrated end-to-end solution that is easy to use**."

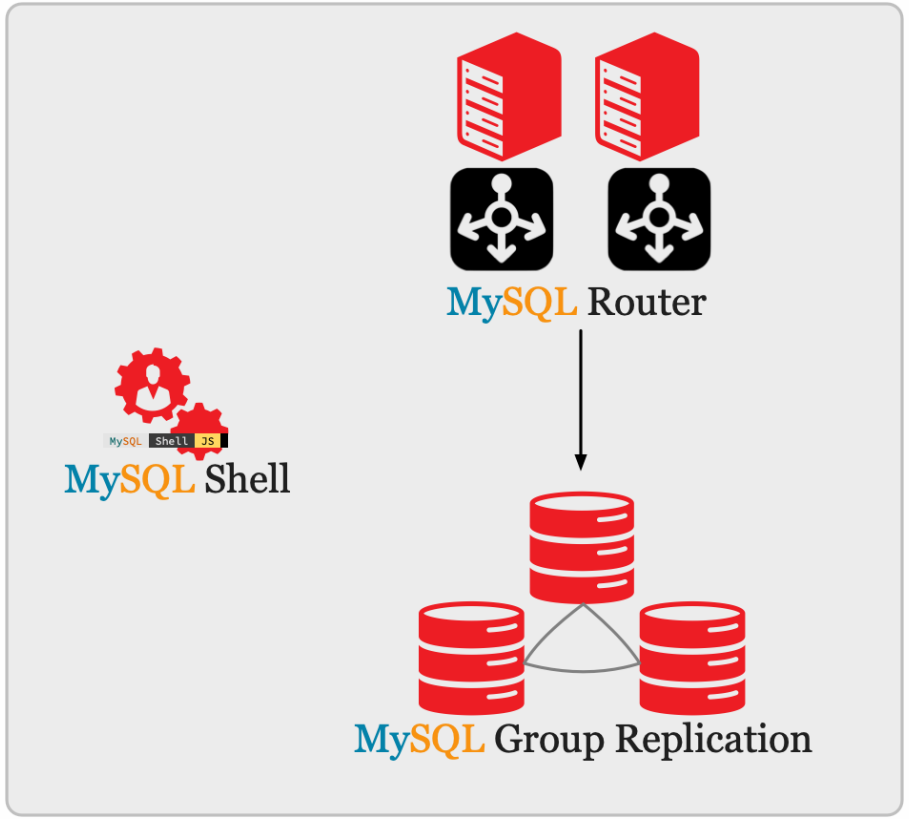
MySQL InnoDB Cluster

"A single product — MySQL — with **high availability** and **scaling features** baked in; providing an **integrated end-to-end solution that is easy to use**."



MySQL InnoDB Cluster

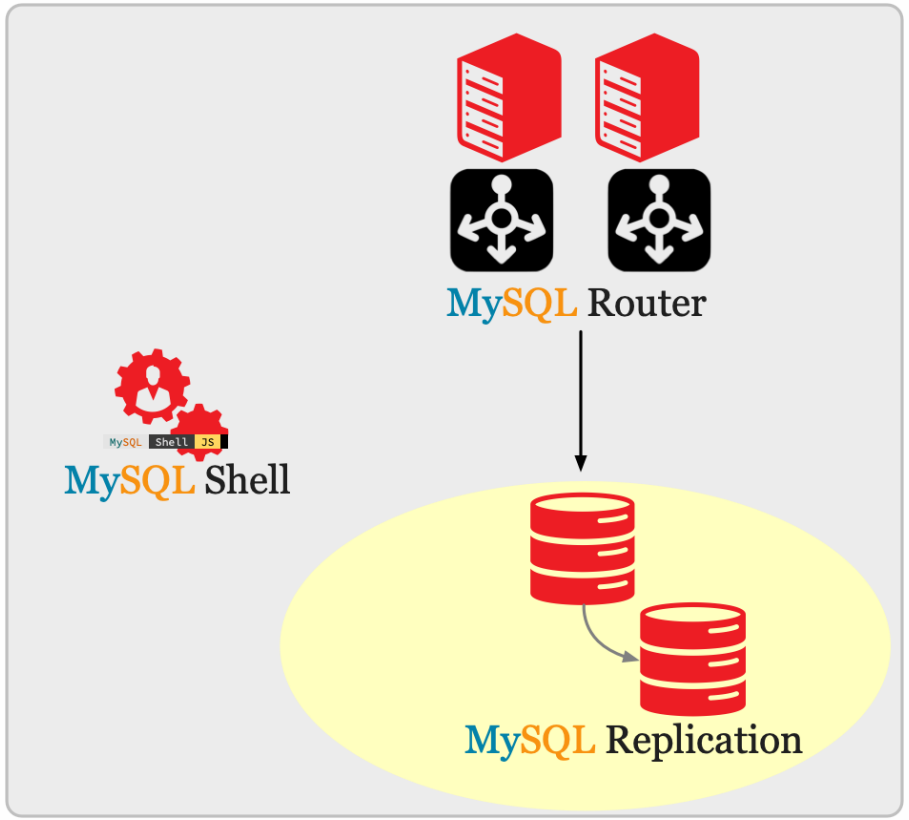
"A single product — MySQL — with **high availability** and **scaling features** baked in; providing an **integrated end-to-end solution that is easy to use**."



High Available Distributed MySQL DB

- Fault tolerance
- Automatic failover
- Active/Active update anywhere (limits apply)
- Automatic membership management
  - Adding/removing members
  - Network partitions, failures
- Conflict detection and resolution
- Prevents data loss
- GPL

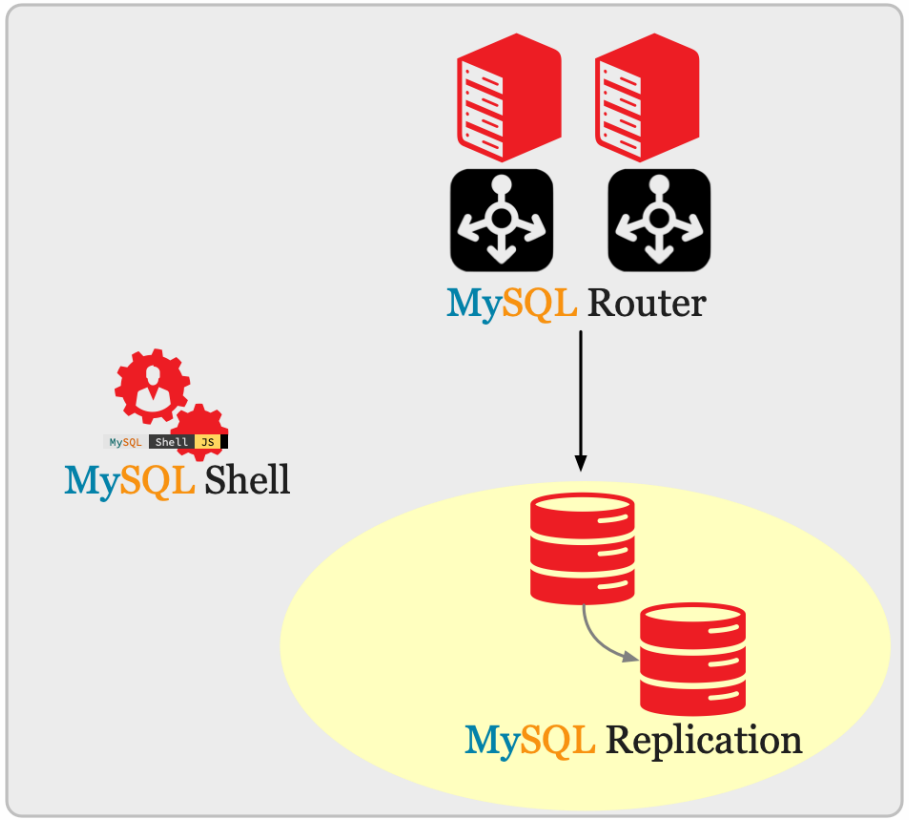
MySQL InnoDB Replicaset



Introducing MySQL InnoDB ReplicaSet!

- 8.0.19 Feature!
- Fully integrated MySQL Router
  - Automatic Routing
- Ease of use with MySQL Shell
  - Configuring, Adding, Removing members
  - Automatic Member Provisioning (`CLONE`)

MySQL InnoDB Replicaset



- Replication Architecture:
  - (manual) Switchover & Failover
  - (asynchronous) Read Scaleout
  - 'Simple' Replication architecture:
    - no network/hardware requirements
      - Providing Availability on PRIMARY when issues with secondaries or network

MySQL InnoDB Cluster & ReplicaSet - Goals

One Product: MySQL

- All components developed together
- Integration of all components
- Full stack testing

MySQL InnoDB Cluster & ReplicaSet - Goals

One Product: MySQL

- All components developed together
- Integration of all components
- Full stack testing

Easy to Use

- One client: MySQL Shell
- Integrated orchestration
- Homogenous servers

MySQL InnoDB Cluster & ReplicaSet - Goals

One Product: MySQL

- All components developed together
- Integration of all components
- Full stack testing

Easy to Use

- One client: MySQL Shell
- Integrated orchestration
- Homogenous servers

Support DNS-SRV since 8.0.19 with our connectors

Solutions Easy to Deploy

Solutions Easy to Deploy

MySQL InnoDB Cluster

```
js> \c admin@mysql1
js> cluster = dba.createCluster('cluster')
```

MySQL InnoDB ReplicaSet

```
js> \c admin@mysql1
js> rs = dba.createReplicaSet('replicaset')
```

Solutions Easy to Deploy

MySQL InnoDB Cluster

```
js> \c admin@mysql1
js> cluster = dba.createCluster('cluster')
```

Configure server to add later

```
js> dba.configureInstance('admin@mysql2')
```

MySQL InnoDB ReplicaSet

```
js> \c admin@mysql1
js> rs = dba.createReplicaSet('replicaset')
```

```
js> dba.configureReplicaSetInstance('admin@mysql2')
```

Solutions Easy to Deploy

MySQL InnoDB Cluster

```
js> \c admin@mysql1
js> cluster = dba.createCluster('cluster')
```

Configure server to add later

```
js> dba.configureInstance('admin@mysql2')
```

Add server to the Cluster

```
js> cluster.addInstance('admin@mysql2')
```

MySQL InnoDB ReplicaSet

```
js> \c admin@mysql1
js> rs = dba.createReplicaSet('replicaset')
```

```
js> dba.configureReplicaSetInstance('admin@mysql2')
```

```
js> rs.addInstance('admin@mysql2')
```

Solutions Easy to Deploy

MySQL InnoDB Cluster

```
js> \c admin@mysql1
js> cluster = dba.createCluster('cluster')
```

Configure server to add later

```
js> dba.configureInstance('admin@mysql2')
```

Add server to the Cluster

```
js> cluster.addInstance('admin@mysql2')
```

Bootstrap MySQL Router

```
$ sudo mysqlrouter --user=mysqlrouter --bootstrap
$ sudo systemctl start mysqlrouter
```

MySQL InnoDB ReplicaSet

```
js> \c admin@mysql1
js> rs = dba.createReplicaSet('replicaset')
```

```
js> dba.configureReplicaSetInstance('admin@mysql2')
```

```
js> rs.addInstance('admin@mysql2')
```

```
$ sudo mysqlrouter --user=mysqlrouter --bootstrap
$ sudo systemctl start mysqlrouter
```

MySQL Database Architecture Summary

Single Region

Requirement	Solution
RTO = hours, RPO = minutes	MySQL Server w. Backups & Binary Log Sync
RTO = hours, RPO = less than a second	MySQL Server w. Backups & Binary Log Stream
RTO = minutes, RPO = less than a second	MySQL InnoDB ReplicaSet
RTO = seconds, RPO = 0	MySQL InnoDB Cluster

Multi Regions

Requirement	Solution
RTO = minutes, RPO = seconds	MySQL InnoDB Cluster w. asynchronous replica
RTO = seconds and/or RPO = 0	Multi Region MySQL InnoDB Cluster w: - 2 regions: consistency level AFTER - 3 regions deployment

MySQL Database Architecture Summary (2)

Other Requirements

Requirement	Solution
Fully Consistent Reads	MySQL InnoDB Cluster w. custom consistency levels
Multi Primary Single Region	MySQL InnoDB Cluster
Multi Primary Multi Regions	Multi Regions MySQL InnoDB Cluster

MySQL Shell

the best tool for DBAs

@lefred

MySQL Shell - Extensible

```
MySQL  127.0.0.1:33060+  Py  mydba.getPasswordExpiration()
+-------------------------------+-------------------------------+-----------------+
| User                          | Password last change          | Expires in      |
+-------------------------------+-------------------------------+-----------------+
| `fred`@`%`                    | 2018-11-12 21:59:56           | expired         |
| `test`@`%`                    | 2018-12-17 09:58:32           | 20 days         |
| `test2`@`%`                   | 2018-11-10 13:16:44           | expired         |
| `test3`@`%`                   | 2018-10-10 13:16:44           | expired         |
| `root`@`localhost`            | 2018-11-16 23:10:41           | expired         |
+-------------------------------+-------------------------------+-----------------+

MySQL  127.0.0.1:33060+  Py  mydba.getPasswordExpiration(False)
+-------------------------------+-------------------------------+-----------------+
| User                          | Password last change          | Expires in      |
+-------------------------------+-------------------------------+-----------------+
| `test`@`%`                    | 2018-12-17 09:58:32           | 20 days         |
+-------------------------------+-------------------------------+-----------------+
```

MySQL Shell - Extensible

```
MySQL  localhost:33060+  performance_schema  2020-04-09 23:40:35
JS  ext.check.getLocks()
+-----------------+--------------+----------------+-------------------+-------------------+-----------------------
| mysql_thread_id | trx_duration | row_locks_held | row_locks_pending | tables_with_locks | current_statement
+-----------------+--------------+----------------+-------------------+-------------------+-----------------------
|              55 | 37.94 s      |              1 |                 0 | test.t            | update t set v='bourry'
|              16 | 31.14 s      |              4 |                 0 | test.t            | update t set v='helolo'
|              62 | 255.91 us    |              0 |                 0 | NULL              | SELECT thr.processlist_
+-----------------+--------------+----------------+-------------------+-------------------+-----------------------
For which thread_id do you want to see locks ? (55) 16
Metadata Locks:
---------------
GRANTED SHARED_READ on test.t
GRANTED SHARED_WRITE on test.t

Data Locks:
-----------
GRANTED TABLE (IX) LOCK on test.t
GRANTED RECORD (X) LOCK on test.t
GRANTED RECORD (X,REC_NOT_GAP) LOCK on test.t
GRANTED RECORD (X,GAP) LOCK on test.t
```

https://github.com/lefred/mysqlshell-plugins

MySQL Database Service

MDS in OCI

@lefred

MySQL in the Cloud

Various options but many questions

- How to get the latest features, security fixes?
- How to provide security & regulatory compliance?
- How to provide compatibility with on-premises?
- How to integrate with Oracle technologies?
- How to get expert MySQL technical support?

Only Oracle provides a MySQL Database Service

- 100% developed and managed by the MySQL team
- 100% built on MySQL Enterprise Edition
- 100% compatible with on-premises MySQL
- 100% compatible with Oracle technologies
- 100% supported by the MySQL Team

MySQL Database Service

100% developed, managed, and supported by the MySQL team

Easy

- Fully Managed Database Service
- Instant Provisioning
- Latest Features

Secure

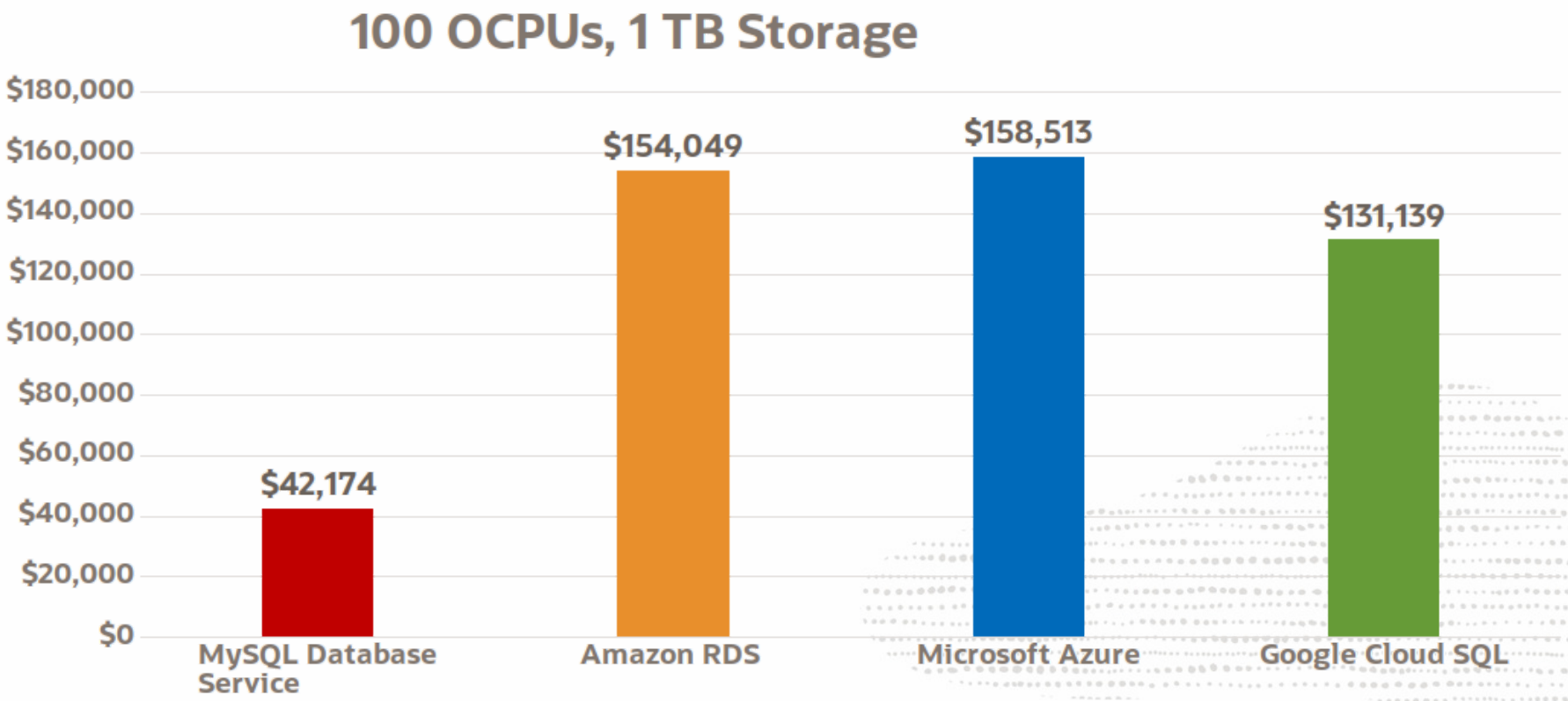
- Data Protection
- Advanced security
- Latest Security Updates

Enterprise-Ready

- Built on MySQL Enterprise Edition
- On Premises Compatibility
- Built on Gen 2 Cloud Infrastructure

MySQL Database Service

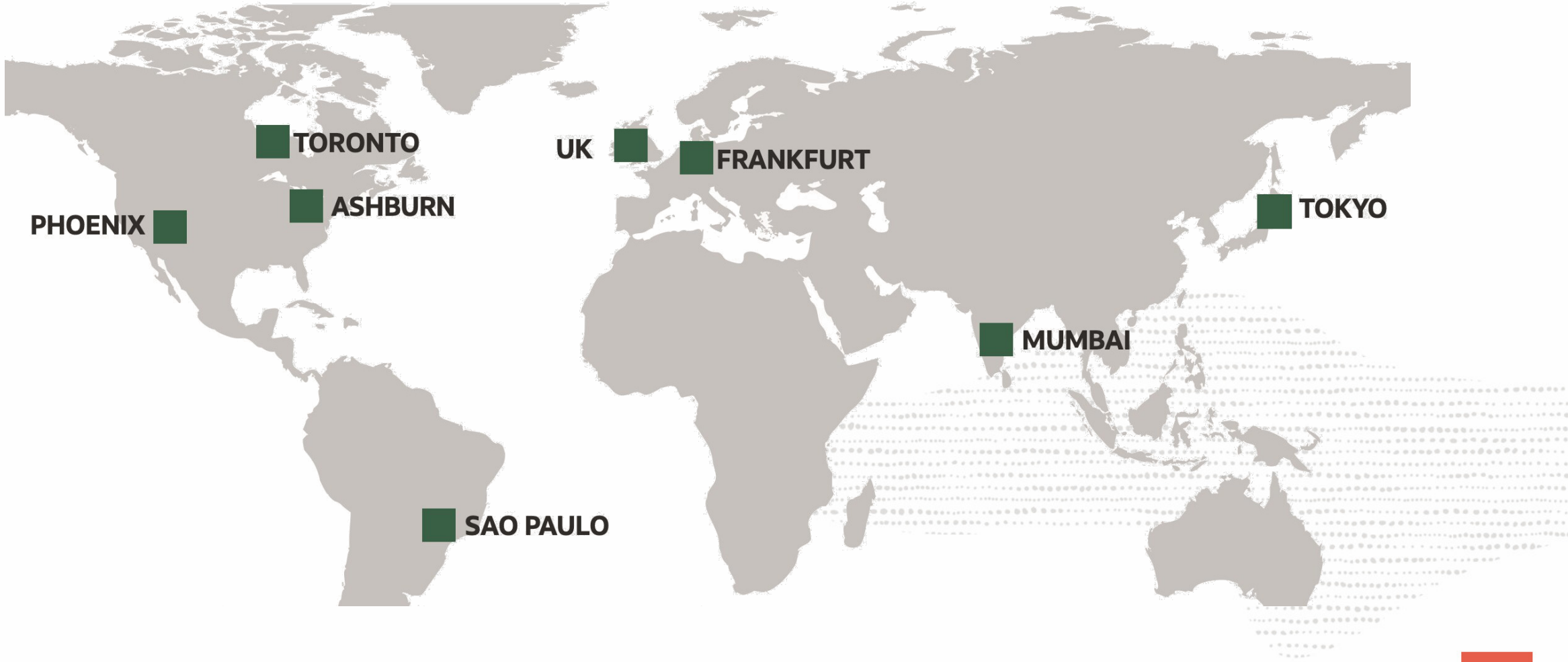
1 Year TCO



MySQL: Standard E2 AMD 8GB/Core
RDS: Intel M5 8GB/Core, AWS US East,
Azure: General Purpose Intel 10GB/Core, MS Azure US-East
Google: N1 Standard Intel 7.5GB/Core, GCP Northern Virginia.
Configuration: 100 OCPUs, 1 TB Storage

Cloud Regions

Situation as Oct 1st 2020



Black Lives Matter

#blacklivesmatter

@lefred

Terminology Changes

We decided to change terminoly in our products. From commands, output, manuals, source code and more !

Terminology Changes

We decided to change terminoly in our products. From commands, output, manuals, source code and more !

Of course this is a huge amount of work and will be distributed in several releases.

Terminology Changes

We decided to change terminoly in our products. From commands, output, manuals, source code and more !

Of course this is a huge amount of work and will be distributed in several releases.

Please check Kenny's announcement for more info: https://mysqlhighavailability.com/mysql-terminology-updates/

Terminology Changes (2)

You can already see some changes in 8.0.22:

```
mysql> show replica status\G
 *************************** 1. row ***************************
             Replica_IO_State: Waiting for master to send event
                  Source_Host: 127.0.0.1
                  Source_User: rsandbox
                  Source_Port: 21223
                Connect_Retry: 60
              Source_Log_File: mysql-bin.000002
          Read_Source_Log_Pos: 156
               Relay_Log_File: mysql-relay.000006
                Relay_Log_Pos: 371
        Relay_Source_Log_File: mysql-bin.000002
           Replica_IO_Running: Yes
          Replica_SQL_Running: Yes
                          ...
           Source_SSL_Allowed: No
                          ...
       Seconds_Behind_Source: 0
Source_SSL_Verify_Server_Cert: No
```

Open Source

MySQL is GPL

@lefred

MySQL Server : GPL
MySQL Router : GPL
MySQL Shell : GPL
MySQL Clone : GPL
MySQL Workbench : GPL

MySQL Server : GPL
MySQL Router : GPL
MySQL Shell : GPL
MySQL Clone : GPL
MySQL Workbench : GPL

And we accept contributions !

MySQL 8.0 is also

- 293 contributions

- 68 contributors (employees not counted 😉)

Top 3 contributors:

- *Facebook: 59*
- *Daniël van Eeden: 41*
- *Laurynas Biveinis: 17*

Thank you !

Join us on MySQL Comunity Slack

https://lefred.be/mysql-community-on-slack

MySQL 8.0 DBA Certification now available

MySQL 8.0

MySQL 8.0 Database Administrator

Exam Number: 1Z0-908

MySQL 8.0 Database Administrator | 1Z0-908

Take your exam online from your home.

Exam Details

Exam Title:	MySQL 8.0 Database Administrator	**Duration:**	140 Minutes
Exam Number:	1Z0-908	**Number of Questions:**	85
		Passing Score:	62%
		Validated Against:	Exam has been validated against MySQL 8.0

MySQL 8.0 Developer Certification now available

MySQL 8.0

MySQL 8.0 Database Developer

Exam Number: 1Z0-909

MySQL 8.0 Database Developer | 1Z0-909

Exam Details

Exam Title:	MySQL 8.0 Database Developer	**Duration:**	90 Minutes
Exam Number:	1Z0-909	**Number of Questions:**	65
Exam Price:	€220 More on exam pricing	**Passing Score:**	62%
Format:	Multiple Choice	**Validated Against:**	This exam has been validated against the version 8.0

Thank you !

@lefred

Soon MySQL 5.6 will be EOL !

< than 4 months !