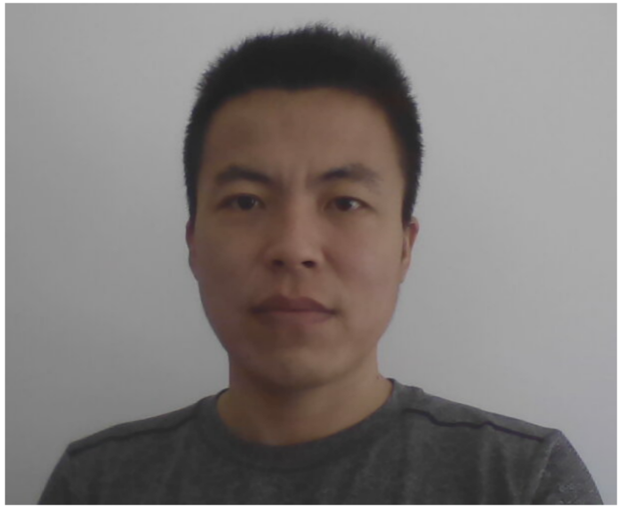
An Introduction to Kunlun Distributed DBMS

Zhao Wei <twitter/linkedin/wechat: david.zhao.cn@gmail.com>

About the author



- Zhao Wei (David Zhao) twitter/linkedin/wechat: david.zhao.cn@gmail.com
- Database kernel developer in Oracle
  - Berkeley DB
  - MySQL
- Database kernel developer in Tencent
  - TDSQL --- most popular distributed DBMS inside Tencent and Tencent Public/Private Cloud
    - Evolved TDSQL from a table-sharding solution to a distributed DBMS
- Started Kunlun project in Aug 2019
  - Goal
    - A distributed DBMS from day 1, with knowledge&lessons learned from TDSQL
    - Premium scalability, availability, fault-tolerance&crash safety
    - Premium performance, ease of use&administration, and autonomousity;
    - Cloud native & DBaaS
    - Cost effective in cloud: seperation of computing & storage paradigm
  - Finished over 60% kernel development

Agenda

- Why we need a distributed DBMS cluster
- Kunlun Architecture
- Kunlun Basics & Examples
- Kunlun Indepth Technologies
- Kunlun Project Progress & Plans

Why we need a distributed DBMS

- Ever growing Data Management Needs
- Commodity hardware infrastructure

Existing solutions

- Application developers pain points
- DBA pain points

Ever growing Data Management Needs

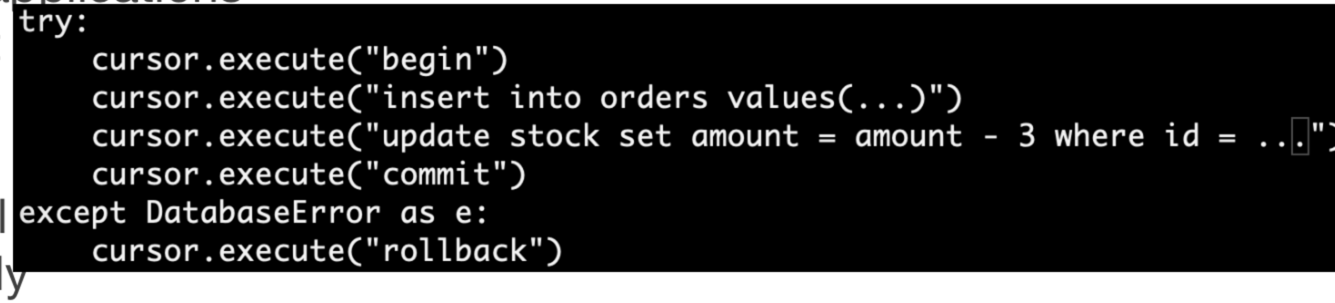
- New techs changing the world NOW!
  - 5G, IoT, sensors, robots, drones
  - auto-pilot & intellegient transportation
  - Intellegient city/manufacture/agriculture
- Data will be produced, accumulated and used much faster & much more extensive
  - by humans, animals, plants, smart devices/equipments, sensors, etc
  - on ground, in water, air and space, 24*7 non-stop
  - Largely relational data, but multi-model(graph, time-series, spatial, json, text)

Hardware infrastructure for computing

- Commodity hardware
  - low cost & moderate reliability
  - limited computing resources&capacity (CPU/memory/storage/network)
  - deployed massively in multiple data centers of multiple places
  - interconnected via tcp/ip network
- Problems of hardware infrastructure
  - server nodes stop working usually&randomly/unpredictably
    - hardware/software fault/failure
    - power outage: node, rack, data-center
    - planned hardware/software maintenance/upgrade
    - planned restart/renewal/retire
  - network issues are common&random/unpredictable
    - partition/congestion
    - break/slow-down
  - Resource bottlenecks easily reached
- Infrastructures Software Needs --- DBMS
  - High availability & crash safety & fault-tolerance
  - Scale out efficiently
    - Single point hotspot kills performance&scalability
    - share nothing(sharding) paradigm

DBMS User Needs

- DBA&Devops
  - least human intervention, esp. in case of random incidents
  - work autonomously
  - help diagnose/analyze/monitor performance and other issues
- Application developers
  - isolate data mgmt complexity from applications
    - agnostic to data physical layout
    - execute transactions&queries
      - handle DB exceptions
  - focus on business logic design&impl
    - adapt to changes&evolve quickly
    - base on simple logical data layer abstraction
  - Agile&predictable development & affordable cost
    - leave common data mgmt work to DBMS



```
try:
    cursor.execute("begin")
    cursor.execute("insert into orders values(...)")
    cursor.execute("update stock set amount = amount - 3 where id = ...")
    cursor.execute("commit")
except DatabaseError as e:
    cursor.execute("rollback")
```

Future DBMS Requirement

- Distributed DBMS Requirement
  - manage TBs to PBs of frequently accessed data
    - no single hotspot/single point of failure
    - multiple read&write nodes
  - run on many commodity hardwares collaborating efficiently
    - highly available
    - crash safe & fault tolerant
  - scale out elastically on demand continuously & automatically
    - zero impact to apps/users
  - run as DBaaS
    - public VS private cloud
    - one cloud VS multi-clouds
    - on premise deployment
  - work autonomously, require least human(DBA) intervention
    - diagnose/analyze/monitor

Existing solutions --- common

- DBMS HA Cluster
  - Achieve HA by replicating entire data set
    - each node stores all data storage: storage/computing capacity bottleneck
  - single primary node for write traffic
    - unscalable for writes
  - multiple replicas, can serve read requests
    - eventual consistency
  - propagate changes to replicas
    - as binlogs/WAL logs
    - MySQL async/semisync/group replication
    - PostgreSQL WAL replication
  - alternative: replicate data/log file blocks using shared storage
    - Aurora and its variants
  - challenges
    - scalability

Existing solutions

- DBMS HA clusters plus middleware/proxy/gateway instances
    - supports sharding
    - supports multiple write&read nodes
    - no support for global transaction/query processing
        - can write only one shard in a transaction
            - or risk inconsistent global transactions in node failures
        - can read only tablets of one shard in a SELECT stmt
            - implement specific multi-table joins in app code
        - some with limited multi tablets (of one table) queries, often aggregates
        - Application developers need to often write their SQL queries/transactions according to data physical layout
    - fixed tablet layout setting, no automatic scaling-out allowed
    - unable to adapt to DBMS node failures/alterations automatically
- DBMS HA clusters plus application level sharding implemented specifically
    - table sharding for each table
    - cross shard DML queries for each query
    - global transaction commit&recovery for each transaction

Existing solutions

- DBMS HA clusters plus micro-services
  - micor-services collaborating asyncly & loosely coupled via message queues
    - user data partitioned by micro-services, each has its own part of data
      - smaller amount data to manage for each service&its DB instance
    - handle inconsistency/eventual consistency between cooperating services
      - hard to impl correct business logic
      - hard to maintain consistent global data snapshot
    - handle service crash-safety
      - reliable message queue positions to resume consumption
  - more infrastructure facilities (message queue & DB cluster deployment)
    - more hardware costs
    - more maintenance/administration work for DBAs & Devops
  - service scale-out VS DBMS scale-out
    - services defined by business domain NOT data
      - can't scale-out a service's data
    - some services may still face huge amount of data
      - e.g. place-order service, money-transfer-out/in services
  - Conclusion: micro-service architecture can't meet/resolve the needs for a distributed DBMS

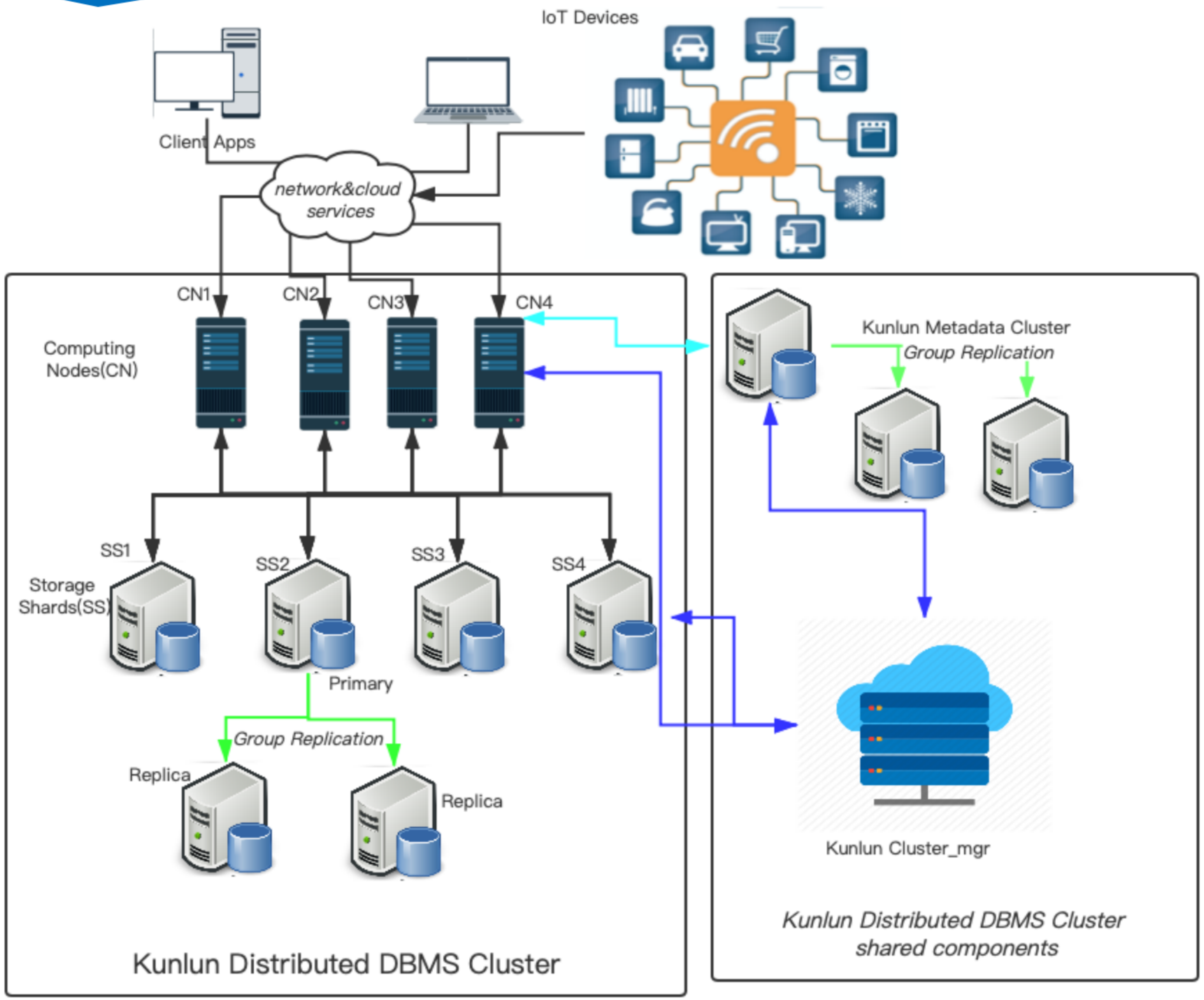
Pain points

- Application developers
  - write SQL queries/transactions according to data physical location
    - physical data layout dependency is nasty!
  - implement cross shard transaction ACID in application code
  - assemble user result using multiple queries in application code
  - overwhelmed by redundant & error prone data mgmt work
    - writing db functionality in application repeatedly for each specific task
- DBAs/devops
  - scale out manually
    - impacts apps/end users
  - handle db node failures manually
  - reconfigure proxy nodes in case of primary switch or when add/drop a table
  - A lot of unforeseeable chores and routines to do, any-time in any day, quite error-prone
- Business owners
  - high human cost and/or hardware/infrastructure cost
  - unpredictable development timespan and quality and slow response to business changes
  - service&revenue&user loss during db node failures

Best approach --- Distributed DBMS

- Manage huge amount of data using distributed DBMS
  - keep application independent from data mgmt work & work automously
  - architect: design with one integral/consistent data snapshot which scales-out on demand
    - natural&straightforward thinking
    - can still use micro-service paramdigms in application design&impl
  - app developers: focus on business logic impl, based on reliable DBMS functionality
    - simpy use SQL stmts, no message queues needed
    - assume transactions and ACID guarantees
  - app developers: all parts/services work with 'simple' application data
    - consistent data snapshot/view
    - always available
    - always crash-safe&fault-tolerant and resillient
    - always sufficient resources --- scales-out on demand
  - DBAs: more efficient & productive
    - automate almost everything, minimal manual maintenance work
    - focus on valuable work: data schema design, performance tuning, resource planning, etc
- Kunlun Distributed DBMS fully meet all such needs

Kunlun Architecture



Kunlun Basic

- Computing nodes
  - accept & validate user connections
  - accept & process user queries
    - parse -> optimize -> execute(send SQL -> receive & assemble)
    - executes DDLs and DMLs
  - can have one or more nodes in a cluster, independent from each other
  - doesn't store user data, only store metadata locally
    - takes trivial storage space
    - store user data in storage shards
  - based on PostgreSQL-11.5, supports pg client protocol
  - supports common PostgreSQL DDL grammar
  - supports most PostgreSQL DML grammar & native data types
  - will support mysql client protocol and common MySQL private DML grammar(pending)

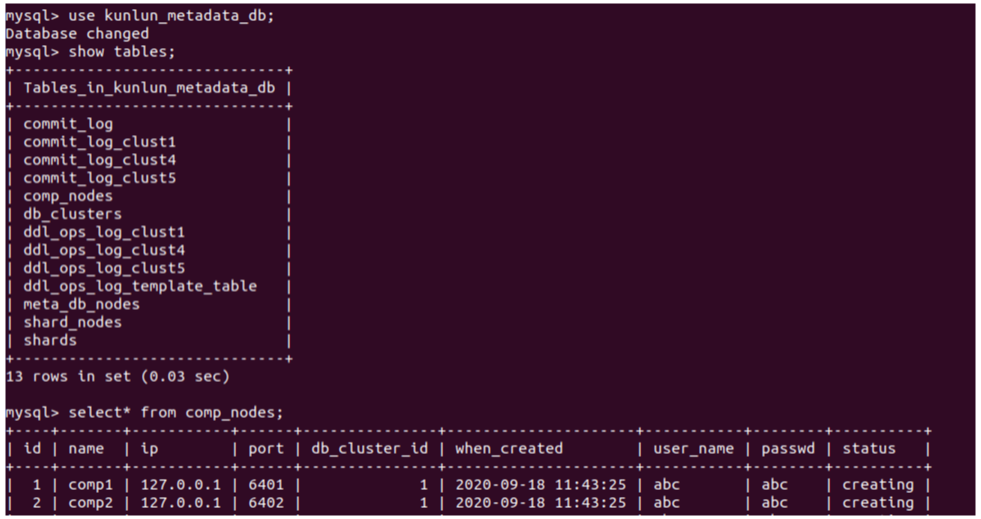
Kunlun Basic

- Storage shards
  - Uses MySQL group replication(MGR) single primary mode for shard HA
    - primary election
    - robust consistency guarantees
  - Require kunlun-percona-MySQL-8.0.18-9
    - developed based on percona-MySQL-8.0.18-9
    - contains critical bug fixes & supporting features
    - will advance versions with upper stream
  - Stores application(user) data in standalone tables
    - PG single tables
    - PG table partitions
  - execute mostly single table queries
    - in a global transaction's local transaction branch

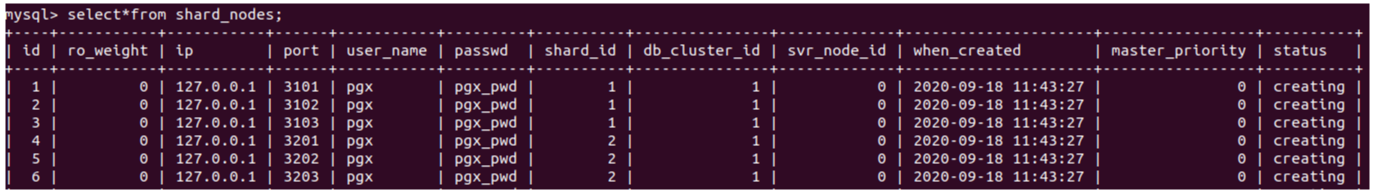
Kunlun Basic

- Metadata Cluster
    - kunlun-percona-mysql MGR cluster
    - Shared by one or more Kunlun distributed db clusters
    - stores metadata of Kunlun clusters
- Cluster_mgr
    - maintain MGR cluster&node online status
        - nodes come&go&rejoin
        - must join GR explicitly
        - primary&replicas join GR differently & primary first
    - startup entire MGR cluster
        - choose the right primary
    - work on all Kunlun clusters registered in a metadata cluster

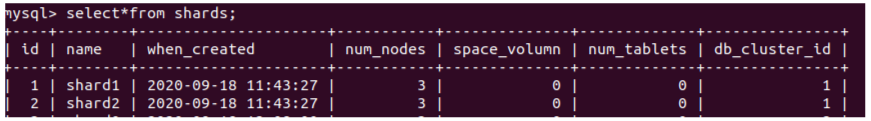
Kunlun Distributed DBMS cluster metadata stored in Metadata Cluster



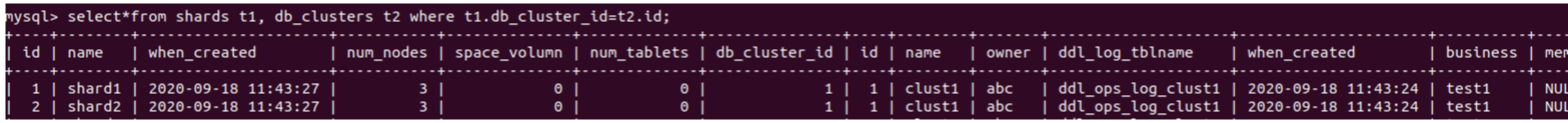
```
mysql> use kunlun_metadata_db;
Database changed
mysql> show tables;
+-----------------------------+
| Tables_in_kunlun_metadata_db |
+-----------------------------+
| commit_log                   |
| commit_log_clust1            |
| commit_log_clust4            |
| commit_log_clust5            |
| comp_nodes                   |
| db_clusters                  |
| ddl_ops_log_clust1           |
| ddl_ops_log_clust4           |
| ddl_ops_log_clust5           |
| ddl_ops_log_template_table   |
| meta_db_nodes                |
| shard_nodes                  |
| shards                       |
+-----------------------------+
13 rows in set (0.03 sec)

mysql> select* from comp_nodes;
+----+-------+-----------+------+---------------+---------------------+-----------+--------+----------+
| id | name  | ip        | port | db_cluster_id | when_created        | user_name | passwd | status   |
+----+-------+-----------+------+---------------+---------------------+-----------+--------+----------+
|  1 | comp1 | 127.0.0.1 | 6401 |             1 | 2020-09-18 11:43:25 | abc       | abc    | creating |
|  2 | comp2 | 127.0.0.1 | 6402 |             1 | 2020-09-18 11:43:25 | abc       | abc    | creating |
```



```
mysql> select*from shard_nodes;
+----+-----------+-----------+------+-----------+---------+----------+---------------+-------------+---------------------+-----------------+----------+
| id | ro_weight | ip        | port | user_name | passwd  | shard_id | db_cluster_id | svr_node_id | when_created        | master_priority | status   |
+----+-----------+-----------+------+-----------+---------+----------+---------------+-------------+---------------------+-----------------+----------+
|  1 |         0 | 127.0.0.1 | 3101 | pgx       | pgx_pwd |        1 |             1 |           0 | 2020-09-18 11:43:27 |               0 | creating |
|  2 |         0 | 127.0.0.1 | 3102 | pgx       | pgx_pwd |        1 |             1 |           0 | 2020-09-18 11:43:27 |               0 | creating |
|  3 |         0 | 127.0.0.1 | 3103 | pgx       | pgx_pwd |        1 |             1 |           0 | 2020-09-18 11:43:27 |               0 | creating |
|  4 |         0 | 127.0.0.1 | 3201 | pgx       | pgx_pwd |        2 |             1 |           0 | 2020-09-18 11:43:27 |               0 | creating |
|  5 |         0 | 127.0.0.1 | 3202 | pgx       | pgx_pwd |        2 |             1 |           0 | 2020-09-18 11:43:27 |               0 | creating |
|  6 |         0 | 127.0.0.1 | 3203 | pgx       | pgx_pwd |        2 |             1 |           0 | 2020-09-18 11:43:27 |               0 | creating |
```

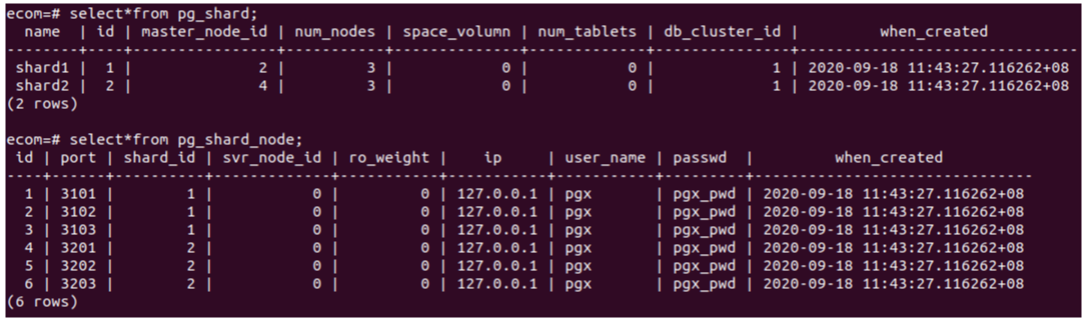


```
mysql> select*from shards;
+----+--------+---------------------+-----------+--------------+-------------+---------------+
| id | name   | when_created        | num_nodes | space_volumn | num_tablets | db_cluster_id |
+----+--------+---------------------+-----------+--------------+-------------+---------------+
|  1 | shard1 | 2020-09-18 11:43:27 |         3 |            0 |           0 |             1 |
|  2 | shard2 | 2020-09-18 11:43:27 |         3 |            0 |           0 |             1 |
```

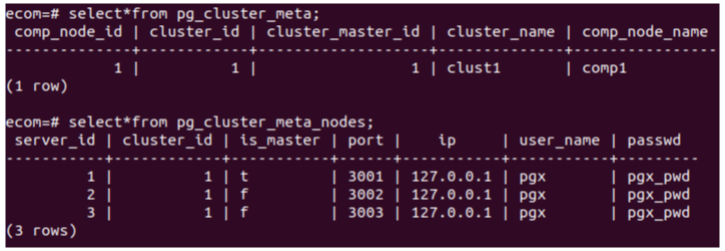


```
mysql> select*from shards t1, db_clusters t2 where t1.db_cluster_id=t2.id;
+----+--------+---------------------+-----------+--------------+-------------+---------------+----+--------+-------+--------------------+---------------------+----------+---
| id | name   | when_created        | num_nodes | space_volumn | num_tablets | db_cluster_id | id | name   | owner | ddl_log_tblname    | when_created        | business | mem
+----+--------+---------------------+-----------+--------------+-------------+---------------+----+--------+-------+--------------------+---------------------+----------+---
|  1 | shard1 | 2020-09-18 11:43:27 |         3 |            0 |           0 |             1 |  1 | clust1 | abc   | ddl_ops_log_clust1 | 2020-09-18 11:43:24 | test1    | NUL
|  2 | shard2 | 2020-09-18 11:43:27 |         3 |            0 |           0 |             1 |  1 | clust1 | abc   | ddl_ops_log_clust1 | 2020-09-18 11:43:24 | test1    | NUL
```

Kunlun Distributed DBMS cluster metadata stored in a computing node

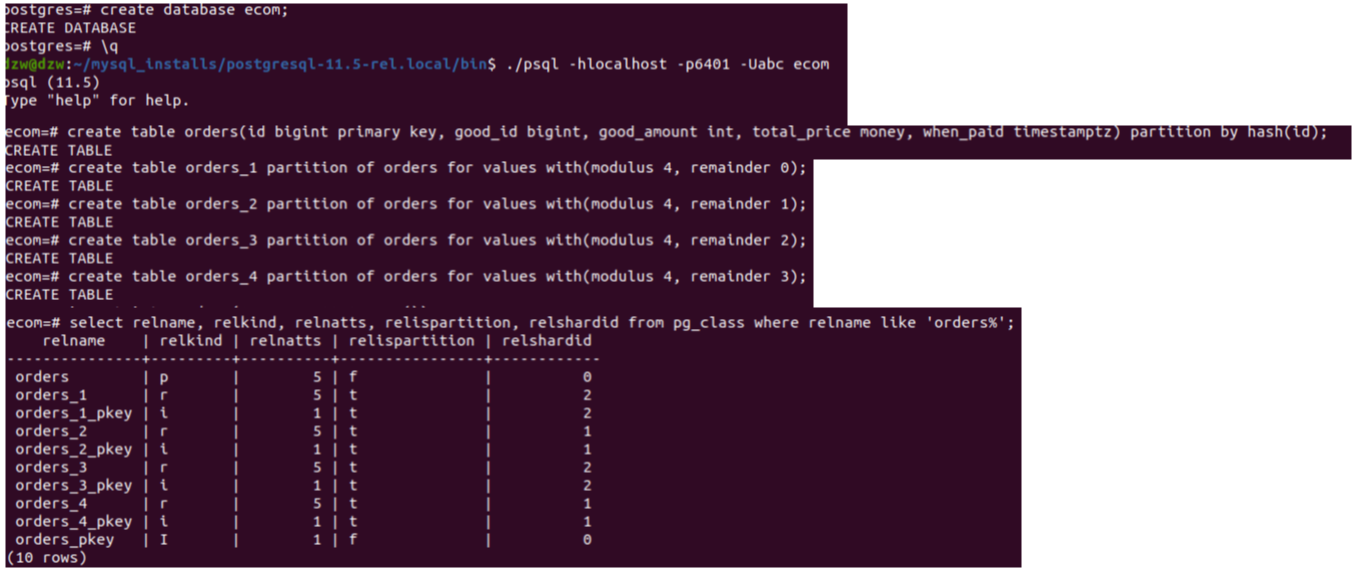


```
ecom=# select*from pg_shard;
  name  | id | master_node_id | num_nodes | space_volumn | num_tablets | db_cluster_id |         when_created
--------+----+----------------+-----------+--------------+-------------+---------------+-------------------------------
 shard1 |  1 |              2 |         3 |            0 |           0 |             1 | 2020-09-18 11:43:27.116262+08
 shard2 |  2 |              4 |         3 |            0 |           0 |             1 | 2020-09-18 11:43:27.116262+08
(2 rows)

ecom=# select*from pg_shard_node;
 id | port | shard_id | svr_node_id | ro_weight |    ip     | user_name | passwd  |         when_created
----+------+----------+-------------+-----------+-----------+-----------+---------+-------------------------------
  1 | 3101 |        1 |           0 |         0 | 127.0.0.1 | pgx       | pgx_pwd | 2020-09-18 11:43:27.116262+08
  2 | 3102 |        1 |           0 |         0 | 127.0.0.1 | pgx       | pgx_pwd | 2020-09-18 11:43:27.116262+08
  3 | 3103 |        1 |           0 |         0 | 127.0.0.1 | pgx       | pgx_pwd | 2020-09-18 11:43:27.116262+08
  4 | 3201 |        2 |           0 |         0 | 127.0.0.1 | pgx       | pgx_pwd | 2020-09-18 11:43:27.116262+08
  5 | 3202 |        2 |           0 |         0 | 127.0.0.1 | pgx       | pgx_pwd | 2020-09-18 11:43:27.116262+08
  6 | 3203 |        2 |           0 |         0 | 127.0.0.1 | pgx       | pgx_pwd | 2020-09-18 11:43:27.116262+08
(6 rows)
```



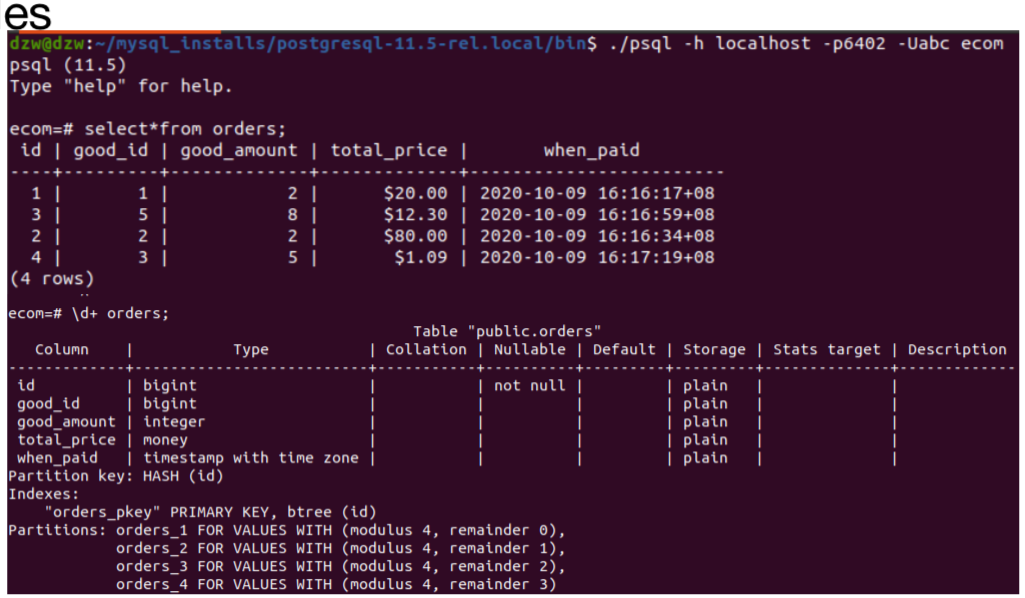
```
ecom=# select*from pg_cluster_meta;
 comp_node_id | cluster_id | cluster_master_id | cluster_name | comp_node_name
--------------+------------+-------------------+--------------+----------------
            1 |          1 |                 1 | clust1       | comp1
(1 row)

ecom=# select*from pg_cluster_meta_nodes;
 server_id | cluster_id | is_master | port |    ip     | user_name | passwd
-----------+------------+-----------+------+-----------+-----------+---------
         1 |          1 | t         | 3001 | 127.0.0.1 | pgx       | pgx_pwd
         2 |          1 | f         | 3002 | 127.0.0.1 | pgx       | pgx_pwd
         3 |          1 | f         | 3003 | 127.0.0.1 | pgx       | pgx_pwd
(3 rows)
```

DDL & Table sharding in Kunlun computing node



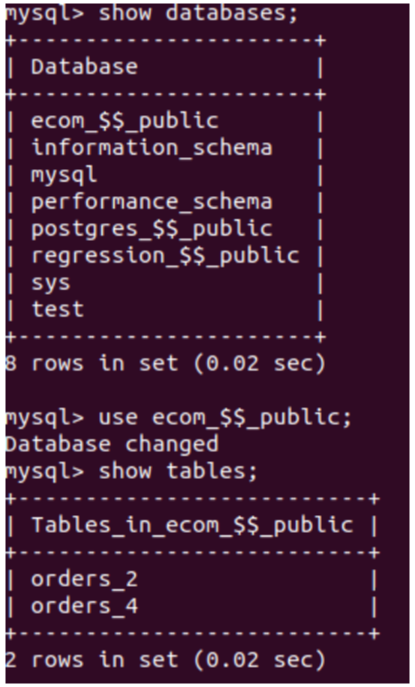
```
postgres=# create database ecom;
CREATE DATABASE
postgres=# \q
dzw@dzw:~/mysql_installs/postgresql-11.5-rel.local/bin$ ./psql -hlocalhost -p6401 -Uabc ecom
psql (11.5)
Type "help" for help.

ecom=# create table orders(id bigint primary key, good_id bigint, good_amount int, total_price money, when_paid timestamptz) partition by hash(id);
CREATE TABLE
ecom=# create table orders_1 partition of orders for values with(modulus 4, remainder 0);
CREATE TABLE
ecom=# create table orders_2 partition of orders for values with(modulus 4, remainder 1);
CREATE TABLE
ecom=# create table orders_3 partition of orders for values with(modulus 4, remainder 2);
CREATE TABLE
ecom=# create table orders_4 partition of orders for values with(modulus 4, remainder 3);
CREATE TABLE
ecom=# select relname, relkind, relnatts, relispartition, relshardid from pg_class where relname like 'orders%';
    relname    | relkind | relnatts | relispartition | relshardid
---------------+---------+----------+----------------+------------
 orders        | p       |        5 | f              |          0
 orders_1      | r       |        5 | t              |          2
 orders_1_pkey | i       |        1 | t              |          2
 orders_2      | r       |        5 | t              |          1
 orders_2_pkey | i       |        1 | t              |          1
 orders_3      | r       |        5 | t              |          2
 orders_3_pkey | i       |        1 | t              |          2
 orders_4      | r       |        5 | t              |          1
 orders_4_pkey | i       |        1 | t              |          1
 orders_pkey   | I       |        1 | f              |          0
(10 rows)
```

ecom database and orders table are accessible in other computing nodes

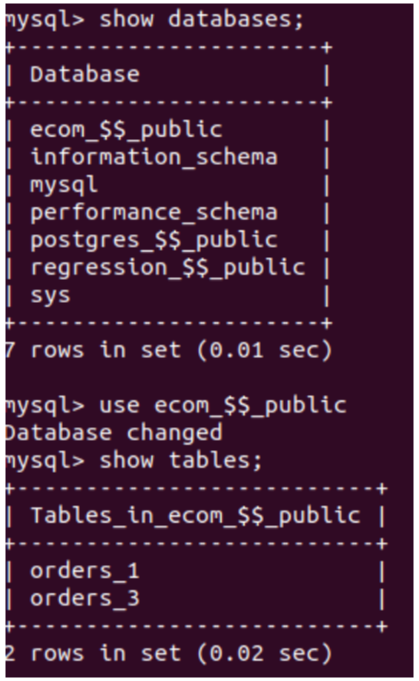


```
dzw@dzw:~/mysql_installs/postgresql-11.5-rel.local/bin$ ./psql -h localhost -p6402 -Uabc ecom
psql (11.5)
Type "help" for help.

ecom=# select*from orders;
 id | good_id | good_amount | total_price |       when_paid
----+---------+-------------+-------------+------------------------
  1 |       1 |           2 |      $20.00 | 2020-10-09 16:16:17+08
  3 |       5 |           8 |      $12.30 | 2020-10-09 16:16:59+08
  2 |       2 |           2 |      $80.00 | 2020-10-09 16:16:34+08
  4 |       3 |           5 |       $1.09 | 2020-10-09 16:17:19+08
(4 rows)

ecom=# \d+ orders;
                                      Table "public.orders"
   Column    |           Type           | Collation | Nullable | Default | Storage | Stats target | Description
-------------+--------------------------+-----------+----------+---------+---------+--------------+-------------
 id          | bigint                   |           | not null |         | plain   |              |
 good_id     | bigint                   |           |          |         | plain   |              |
 good_amount | integer                  |           |          |         | plain   |              |
 total_price | money                    |           |          |         | plain   |              |
 when_paid   | timestamp with time zone |           |          |         | plain   |              |
Partition key: HASH (id)
Indexes:
    "orders_pkey" PRIMARY KEY, btree (id)
Partitions: orders_1 FOR VALUES WITH (modulus 4, remainder 0),
            orders_2 FOR VALUES WITH (modulus 4, remainder 1),
            orders_3 FOR VALUES WITH (modulus 4, remainder 2),
            orders_4 FOR VALUES WITH (modulus 4, remainder 3)
```

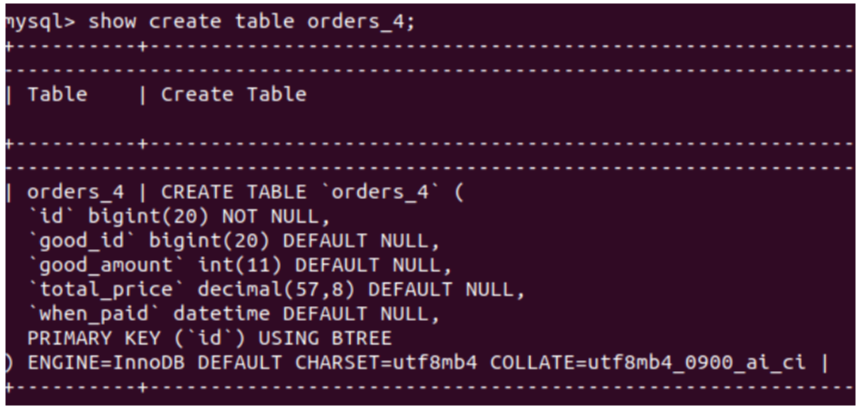
DDL & Table sharding in Kunlun's storage shards



```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| ecom_$$_public     |
| information_schema |
| mysql              |
| performance_schema |
| postgres_$$_public |
| regression_$$_public |
| sys                |
| test               |
+--------------------+
8 rows in set (0.02 sec)

mysql> use ecom_$$_public;
Database changed
mysql> show tables;
+--------------------------+
| Tables_in_ecom_$$_public |
+--------------------------+
| orders_2                 |
| orders_4                 |
+--------------------------+
2 rows in set (0.02 sec)
```

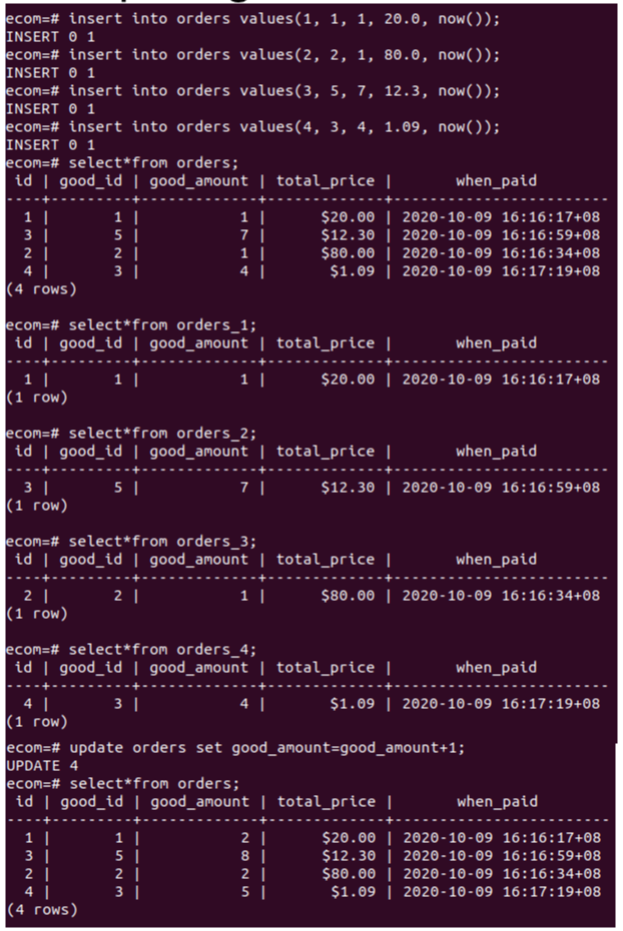


```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| ecom_$$_public     |
| information_schema |
| mysql              |
| performance_schema |
| postgres_$$_public |
| regression_$$_public |
| sys                |
+--------------------+
7 rows in set (0.01 sec)

mysql> use ecom_$$_public
Database changed
mysql> show tables;
+--------------------------+
| Tables_in_ecom_$$_public |
+--------------------------+
| orders_1                 |
| orders_3                 |
+--------------------------+
2 rows in set (0.02 sec)
```



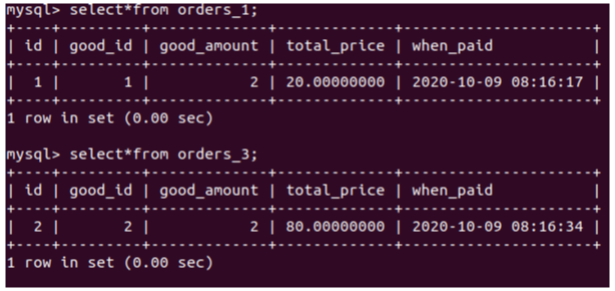
```
mysql> show create table orders_4;
+----------+--------------------------------------------------------
--------------------------------------------------------------------
| Table    | Create Table

+----------+--------------------------------------------------------
--------------------------------------------------------------------
| orders_4 | CREATE TABLE `orders_4` (
  `id` bigint(20) NOT NULL,
  `good_id` bigint(20) DEFAULT NULL,
  `good_amount` int(11) DEFAULT NULL,
  `total_price` decimal(57,8) DEFAULT NULL,
  `when_paid` datetime DEFAULT NULL,
  PRIMARY KEY (`id`) USING BTREE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci |
+----------+--------------------------------------------------------
```

DML & transactions in computing node

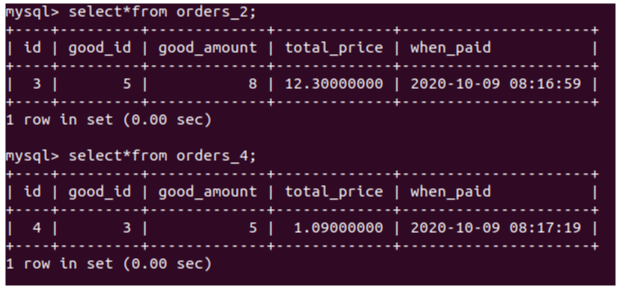


```
ecom=# insert into orders values(1, 1, 1, 20.0, now());
INSERT 0 1
ecom=# insert into orders values(2, 2, 1, 80.0, now());
INSERT 0 1
ecom=# insert into orders values(3, 5, 7, 12.3, now());
INSERT 0 1
ecom=# insert into orders values(4, 3, 4, 1.09, now());
INSERT 0 1
ecom=# select*from orders;
 id | good_id | good_amount | total_price |       when_paid
----+---------+-------------+-------------+------------------------
  1 |       1 |           1 |      $20.00 | 2020-10-09 16:16:17+08
  3 |       5 |           7 |      $12.30 | 2020-10-09 16:16:59+08
  2 |       2 |           1 |      $80.00 | 2020-10-09 16:16:34+08
  4 |       3 |           4 |       $1.09 | 2020-10-09 16:17:19+08
(4 rows)

ecom=# select*from orders_1;
 id | good_id | good_amount | total_price |       when_paid
----+---------+-------------+-------------+------------------------
  1 |       1 |           1 |      $20.00 | 2020-10-09 16:16:17+08
(1 row)

ecom=# select*from orders_2;
 id | good_id | good_amount | total_price |       when_paid
----+---------+-------------+-------------+------------------------
  3 |       5 |           7 |      $12.30 | 2020-10-09 16:16:59+08
(1 row)

ecom=# select*from orders_3;
 id | good_id | good_amount | total_price |       when_paid
----+---------+-------------+-------------+------------------------
  2 |       2 |           1 |      $80.00 | 2020-10-09 16:16:34+08
(1 row)

ecom=# select*from orders_4;
 id | good_id | good_amount | total_price |       when_paid
----+---------+-------------+-------------+------------------------
  4 |       3 |           4 |       $1.09 | 2020-10-09 16:17:19+08
(1 row)
ecom=# update orders set good_amount=good_amount+1;
UPDATE 4
ecom=# select*from orders;
 id | good_id | good_amount | total_price |       when_paid
----+---------+-------------+-------------+------------------------
  1 |       1 |           2 |      $20.00 | 2020-10-09 16:16:17+08
  3 |       5 |           8 |      $12.30 | 2020-10-09 16:16:59+08
  2 |       2 |           2 |      $80.00 | 2020-10-09 16:16:34+08
  4 |       3 |           5 |       $1.09 | 2020-10-09 16:17:19+08
(4 rows)
```

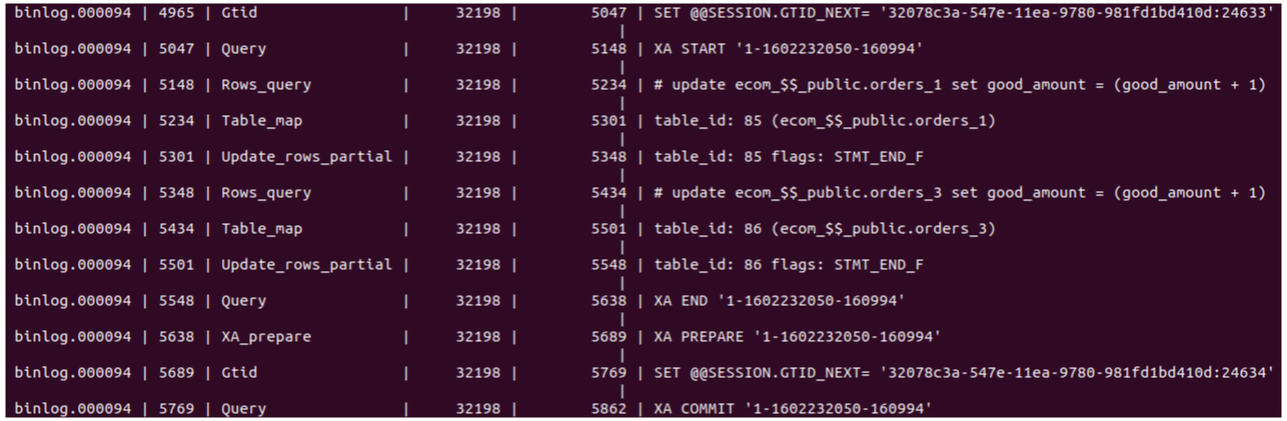
DML & transactions in storage shards



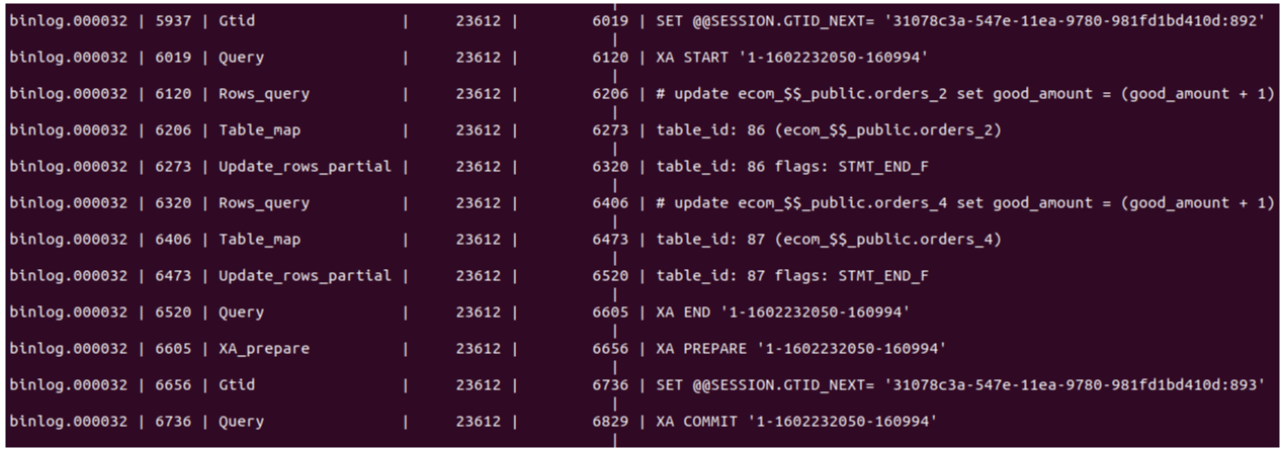
```
mysql> select*from orders_1;
+----+---------+-------------+-------------+---------------------+
| id | good_id | good_amount | total_price | when_paid           |
+----+---------+-------------+-------------+---------------------+
|  1 |       1 |           2 | 20.00000000 | 2020-10-09 08:16:17 |
+----+---------+-------------+-------------+---------------------+
1 row in set (0.00 sec)

mysql> select*from orders_3;
+----+---------+-------------+-------------+---------------------+
| id | good_id | good_amount | total_price | when_paid           |
+----+---------+-------------+-------------+---------------------+
|  2 |       2 |           2 | 80.00000000 | 2020-10-09 08:16:34 |
+----+---------+-------------+-------------+---------------------+
1 row in set (0.00 sec)
```



```
mysql> select*from orders_2;
+----+---------+-------------+-------------+---------------------+
| id | good_id | good_amount | total_price | when_paid           |
+----+---------+-------------+-------------+---------------------+
|  3 |       5 |           8 | 12.30000000 | 2020-10-09 08:16:59 |
+----+---------+-------------+-------------+---------------------+
1 row in set (0.00 sec)

mysql> select*from orders_4;
+----+---------+-------------+-------------+---------------------+
| id | good_id | good_amount | total_price | when_paid           |
+----+---------+-------------+-------------+---------------------+
|  4 |       3 |           5 |  1.09000000 | 2020-10-09 08:17:19 |
+----+---------+-------------+-------------+---------------------+
1 row in set (0.00 sec)
```



```
 binlog.000094 | 4965 | Gtid                |     32198 |        5047 | SET @@SESSION.GTID_NEXT= '32078c3a-547e-11ea-9780-981fd1bd410d:24633'
                                                                     |
 binlog.000094 | 5047 | Query               |     32198 |        5148 | XA START '1-1602232050-160994'
                                                                     |
 binlog.000094 | 5148 | Rows_query          |     32198 |        5234 | # update ecom_$$_public.orders_1 set good_amount = (good_amount + 1)
                                                                     |
 binlog.000094 | 5234 | Table_map           |     32198 |        5301 | table_id: 85 (ecom_$$_public.orders_1)
                                                                     |
 binlog.000094 | 5301 | Update_rows_partial |     32198 |        5348 | table_id: 85 flags: STMT_END_F
                                                                     |
 binlog.000094 | 5348 | Rows_query          |     32198 |        5434 | # update ecom_$$_public.orders_3 set good_amount = (good_amount + 1)
                                                                     |
 binlog.000094 | 5434 | Table_map           |     32198 |        5501 | table_id: 86 (ecom_$$_public.orders_3)
                                                                     |
 binlog.000094 | 5501 | Update_rows_partial |     32198 |        5548 | table_id: 86 flags: STMT_END_F
                                                                     |
 binlog.000094 | 5548 | Query               |     32198 |        5638 | XA END '1-1602232050-160994'
                                                                     |
 binlog.000094 | 5638 | XA_prepare          |     32198 |        5689 | XA PREPARE '1-1602232050-160994'
                                                                     |
 binlog.000094 | 5689 | Gtid                |     32198 |        5769 | SET @@SESSION.GTID_NEXT= '32078c3a-547e-11ea-9780-981fd1bd410d:24634'
                                                                     |
 binlog.000094 | 5769 | Query               |     32198 |        5862 | XA COMMIT '1-1602232050-160994'
```



```
binlog.000032 | 5937 | Gtid                |     23612 |        6019 | SET @@SESSION.GTID_NEXT= '31078c3a-547e-11ea-9780-981fd1bd410d:892'
                                                                    |
binlog.000032 | 6019 | Query               |     23612 |        6120 | XA START '1-1602232050-160994'
                                                                    |
binlog.000032 | 6120 | Rows_query          |     23612 |        6206 | # update ecom_$$_public.orders_2 set good_amount = (good_amount + 1)
                                                                    |
binlog.000032 | 6206 | Table_map           |     23612 |        6273 | table_id: 86 (ecom_$$_public.orders_2)
                                                                    |
binlog.000032 | 6273 | Update_rows_partial |     23612 |        6320 | table_id: 86 flags: STMT_END_F
                                                                    |
binlog.000032 | 6320 | Rows_query          |     23612 |        6406 | # update ecom_$$_public.orders_4 set good_amount = (good_amount + 1)
                                                                    |
binlog.000032 | 6406 | Table_map           |     23612 |        6473 | table_id: 87 (ecom_$$_public.orders_4)
                                                                    |
binlog.000032 | 6473 | Update_rows_partial |     23612 |        6520 | table_id: 87 flags: STMT_END_F
                                                                    |
binlog.000032 | 6520 | Query               |     23612 |        6605 | XA END '1-1602232050-160994'
                                                                    |
binlog.000032 | 6605 | XA_prepare          |     23612 |        6656 | XA PREPARE '1-1602232050-160994'
                                                                    |
binlog.000032 | 6656 | Gtid                |     23612 |        6736 | SET @@SESSION.GTID_NEXT= '31078c3a-547e-11ea-9780-981fd1bd410d:893'
                                                                    |
binlog.000032 | 6736 | Query               |     23612 |        6829 | XA COMMIT '1-1602232050-160994'
                                                                    |
```

Kunlun Indepth --- Computing nodes

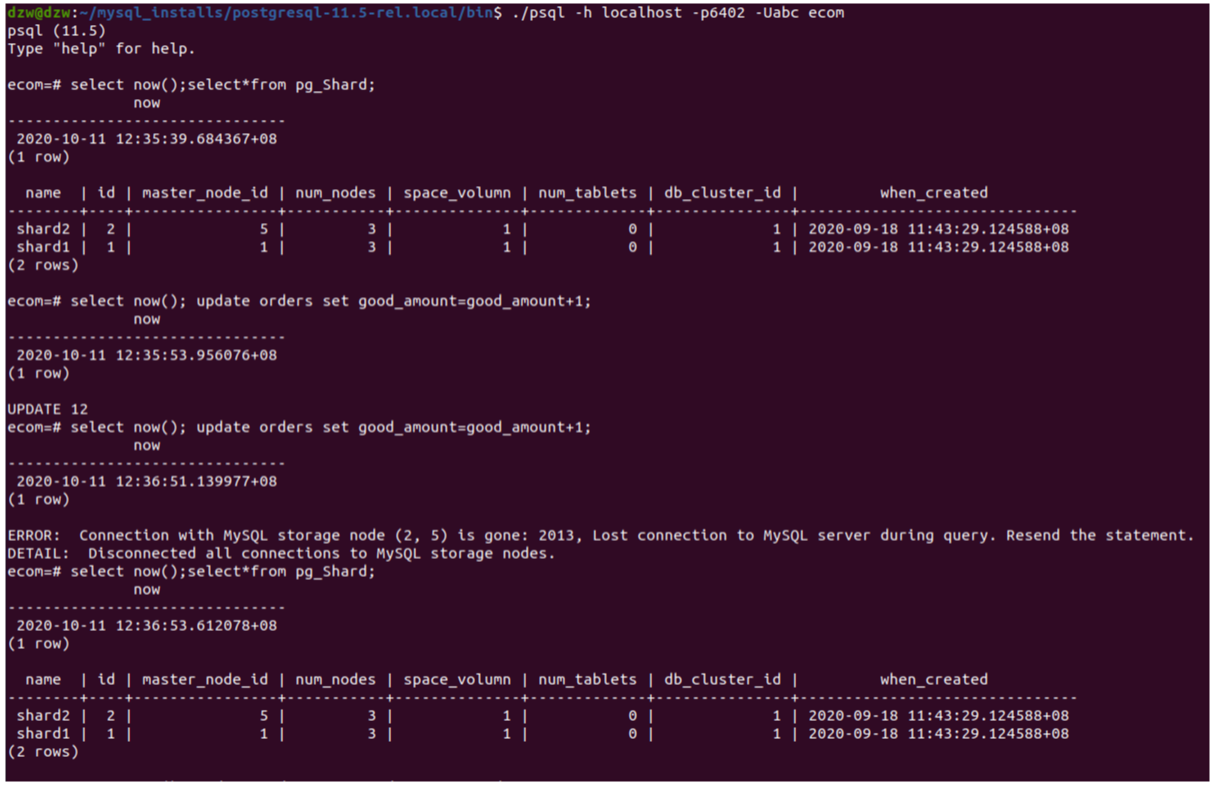
- Table sharding
  - Table mapping between computing nodes&storage shards
    - single table -> single table
    - table partition -> single table
    - use tables OR partitioned tables ?
  - Specifiy shard keys in 'create table' stmt
    - any (group of) columns
    - enable precise control of table data distribution for best performance
    - suggested simple default: use primary key
    - must be included in pk/unique keys
  - Table sharding methods: PostgreSQL table partitioning methods
    - hash
    - range
    - list
  - map rows of table partitions to target on storage shards
    - automatic&transparent

Kunlun Indepth --- Computing nodes

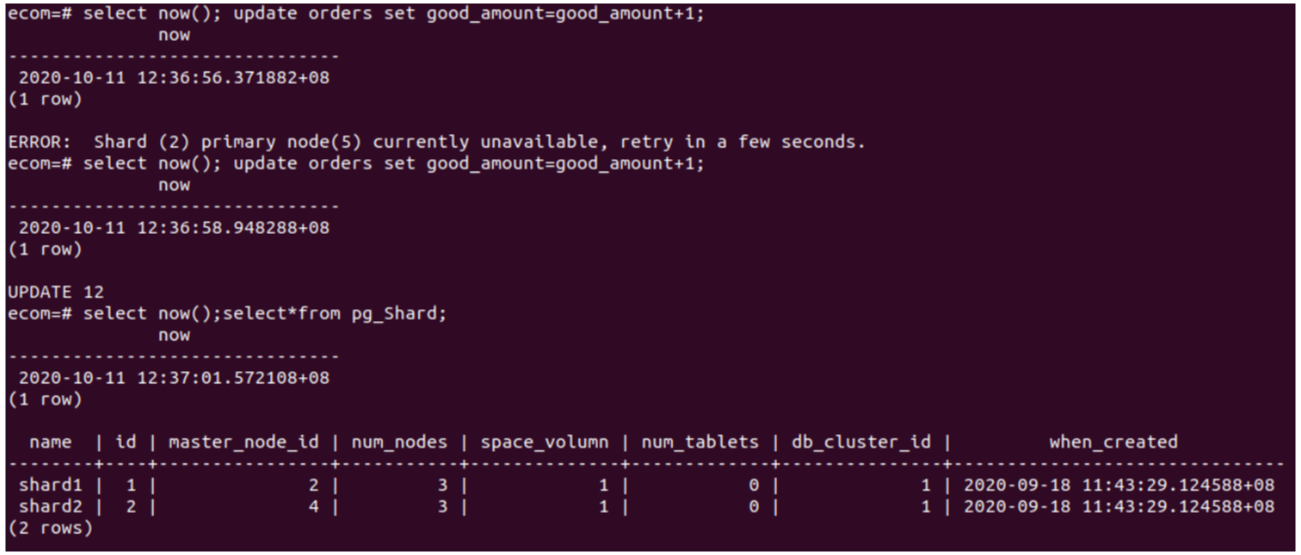
- Global transaction coordinator
  - two phase commit for transactions writing to multiple shards
    - one phase commit for 0/1 written shards & readonly shards
  - can resist node failures/network issues during commit
- App developers can
  - use transactions as if using standalone db
  - write to multiple shards in a transaction

Kunlun Indepth --- Computing nodes

- Storage resillience & auto failover
  - adapt to primary node failures of storage shards/metadata cluster automatically
  - Always use latest primary node for write
  - check against potential issues of MGR



```
dzw@dzw:~/mysql_installs/postgresql-11.5-rel.local/bin$ ./psql -h localhost -p6402 -Uabc ecom
psql (11.5)
Type "help" for help.

ecom=# select now();select*from pg_Shard;
              now
-------------------------------
 2020-10-11 12:35:39.684367+08
(1 row)

  name  | id | master_node_id | num_nodes | space_volumn | num_tablets | db_cluster_id |         when_created
--------+----+----------------+-----------+--------------+-------------+---------------+-------------------------------
 shard2 |  2 |              5 |         3 |            1 |           0 |             1 | 2020-09-18 11:43:29.124588+08
 shard1 |  1 |              1 |         3 |            1 |           0 |             1 | 2020-09-18 11:43:29.124588+08
(2 rows)

ecom=# select now(); update orders set good_amount=good_amount+1;
              now
-------------------------------
 2020-10-11 12:35:53.956076+08
(1 row)

UPDATE 12
ecom=# select now(); update orders set good_amount=good_amount+1;
              now
-------------------------------
 2020-10-11 12:36:51.139977+08
(1 row)

ERROR:  Connection with MySQL storage node (2, 5) is gone: 2013, Lost connection to MySQL server during query. Resend the statement.
DETAIL:  Disconnected all connections to MySQL storage nodes.
ecom=# select now();select*from pg_Shard;
              now
-------------------------------
 2020-10-11 12:36:53.612078+08
(1 row)

  name  | id | master_node_id | num_nodes | space_volumn | num_tablets | db_cluster_id |         when_created
--------+----+----------------+-----------+--------------+-------------+---------------+-------------------------------
 shard2 |  2 |              5 |         3 |            1 |           0 |             1 | 2020-09-18 11:43:29.124588+08
 shard1 |  1 |              1 |         3 |            1 |           0 |             1 | 2020-09-18 11:43:29.124588+08
(2 rows)
```

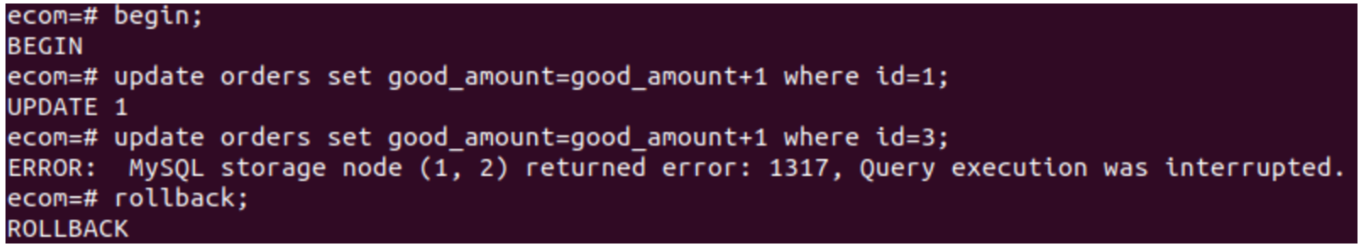


```
ecom=# select now(); update orders set good_amount=good_amount+1;
              now
-------------------------------
 2020-10-11 12:36:56.371882+08
(1 row)

ERROR:  Shard (2) primary node(5) currently unavailable, retry in a few seconds.
ecom=# select now(); update orders set good_amount=good_amount+1;
              now
-------------------------------
 2020-10-11 12:36:58.948288+08
(1 row)

UPDATE 12
ecom=# select now();select*from pg_Shard;
              now
-------------------------------
 2020-10-11 12:37:01.572108+08
(1 row)

  name  | id | master_node_id | num_nodes | space_volumn | num_tablets | db_cluster_id |         when_created
--------+----+----------------+-----------+--------------+-------------+---------------+-------------------------------
 shard1 |  1 |              2 |         3 |            1 |           0 |             1 | 2020-09-18 11:43:29.124588+08
 shard2 |  2 |              4 |         3 |            1 |           0 |             1 | 2020-09-18 11:43:29.124588+08
(2 rows)
```

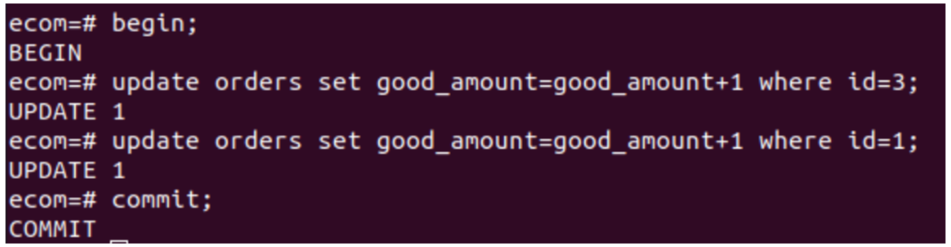
primary election & auto-failover completed within 8 seconds

Kunlun Indepth --- Computing nodes

- Global deadlock detector(GDD)
  - local wait-for edges form global cyclic wait-for graph
  - undetected in single innodb
  - detected&resolved periodically&actively
  - when writing to multiple DB HA cluster in app code, still need GDD
    - Can not without supporting features
    - alternative: lock/stmt timeout: resolves slowly
  - e.g. On shard1, GT1.LT1 -> GT2.LT1 ==> GT1 -> GT2
    On shard2, GT2.LT2 -> GT1.LT2 ==> GT2 -> GT1



```
ecom=# begin;
BEGIN
ecom=# update orders set good_amount=good_amount+1 where id=1;
UPDATE 1
ecom=# update orders set good_amount=good_amount+1 where id=3;
ERROR:  MySQL storage node (1, 2) returned error: 1317, Query execution was interrupted.
ecom=# rollback;
ROLLBACK
```



```
ecom=# begin;
BEGIN
ecom=# update orders set good_amount=good_amount+1 where id=3;
UPDATE 1
ecom=# update orders set good_amount=good_amount+1 where id=1;
UPDATE 1
ecom=# commit;
COMMIT
```

Kunlun Indepth --- Computing nodes

- DDL synchronization
  - no human intervention needed
  - executed in storage shards automatically
  - replicated by all other computing nodes
    - All computing nodes share consistent metadata
  - DDL executed as an autocommit transaction
    - crash safe & fault-tolerant
    - MySQL 8.0 atomic DDL: can't be aborted
    - concurrency control: allow consistent concurrent execution

Kunlun Indepth --- Computing nodes

- Scalability
  - replica reads (under development)
    - chooses the right one at the right time
    - eventual consistent, inconsistency tolerant
  - async data read/write with target storage shards
    - parallel query execution
    - high performance
  - balanced tablet (re)distribution on new storage shards (pending)
    - elastically & continuously & on demand & automatically
    - done at background
    - undetected in application or by end users
  - Typical cost effective usage
    - start with one or a few shards
    - scale out on demand
  - Single shard usage benefit
    - storage resillience&auto failover
    - replica read
    - cluster&node GR state maintenenace cluster_mgr
    - scale out on demand, no capacity planning/restriction

Kunlun Indepth -- Storage shards

- Storage shards
  - Crash safety&fault tolerance challenges in MySQL XA transaction processing
    - keep XA transactions in innodb and binlog identical
    - keep XA transactions in primary and replicas identical
    - keep XA transaction gtids in innodb undo log and binlog identical (8.0)
  - Performance enhancement in XA transaction processing
    - 50%+ QPS increase & 50%+ latency decrease in sysbench write cases
      - https://zhuanlan.zhihu.com/p/151664455
    - Less frequent jitter in QPS if any
  - Supporting features needed by computing nodes
    - transaction status for global deadlock detector
  - MGR issues
    - bug#101114

Kunlun Development Progress

- Completed functionality (latest release version: 0.7)
  - Table sharding
  - Global transaction processing
  - single table DML queries
  - crash safety&fault tolerance and auto failover
  - Common DDLs(create/drop db/schema/table/index)
  - DDL synchronization
  - Basic cluster mgmt
  - POC ready
- On-going development (version 0.8)
  - advanced cross shard multi-table query processing
  - advanced query supporting features
    - sequences
    - prepared stmts
    - query cache
- Project access
  - https://github.com/david-zhao/Kunlun
  - A lot of tech articles around kunlun & db in general: https://www.zhihu.com/column/dbtech
  - Latest released binary download: https://share.weiyun.com/PCIfvwFF

Kunlun Development Plan

- Planned work (version 0.9 and higher)
  - k8s & containerization
  - DBaaS & Cloud native
  - elastic scale-out
  - mysql client protocol & popular private DML stmts
  - More DDL stmts
    - alter table
    - views
  - Cluster Backup & Restore
  - Advanced data types
    - json/spatial
  - Stored procedure & more advanced query processing
  - GUI for DBA&Devops and end users
    - administration/mgmt
    - diagnosis/analysis/monitor
  - Improve db internal security
  - User Requested features

Q&A

Thank You

Zhao Wei

2020-10-17