SphereEx

# SphereEx-DBPlusEngine

# User Manual

V1.2.0

2022.11

# 目录

# 1 Product Introduction

- The differences between the open source and commercial editions

## 1.1 What is DBPlusEngine?

DBPlusEngine is based on the open source kernel ShardingSphere, which is packaged with the addition of enhanced enterprise-class features, which can provide enterprises with enhanced data service capabilities, including but not limited to data sharding and data security.

It consists of DBPlusEngine-Driver and DBPlusEngine-Proxy, two products that can be deployed independently or with hybrid deployments. They all provide standardized data level scaling, distributed transactions, and distributed governance capabilities, and can be applied to a variety of diverse application scenarios such as Java isomorphism, heterogeneous languages, and cloud native.

## 1.2 Core Concepts

- Connect

Flexible adaptation of database protocol, SQL dialect and database storage. It can quickly connect applications and heterogeneous databases.

- Enhance

Capture database access entry to transparently provide additional features, such as: redirect (sharding, read/write splitting and shadow), transform (data encryption and masking), authentication (security, audit and authority), governance (circuit breaker and access limitation and analyze, QoS and observability).

- Pluggable

Leveraging the micro kernel and 3 layers pluggable architecture, features and databases can be embedded flexibly. Developers can customize their ShardingSphere just like building with lego blocks.

- Cluster mode

Provides metadata sharing between multiple DBPlusEngine instances and state coordination in distributed scenarios. In a production environment where the real deployment goes live, you must use the cluster mode. It provides the capabilities necessary for distributed systems such as horizontal scaling of computing power and high availability. Clustered environments need to store metadata and coordinate node status through a separately deployed registry center.

## 1.3 Architecture

SphereEx-DBPlusEngine focuses on pluggable architecture, providing features that can flexibly be embedded into projects. Currently, features such as data sharding, read/write splitting, data encryption, shadow database, and SQL dialects / database protocols such as MySQL, PostgreSQL, SQLServer, Oracle are supported and are all weaved by plugins. Developers can customize their own ShardingSphere just like building with lego blocks.

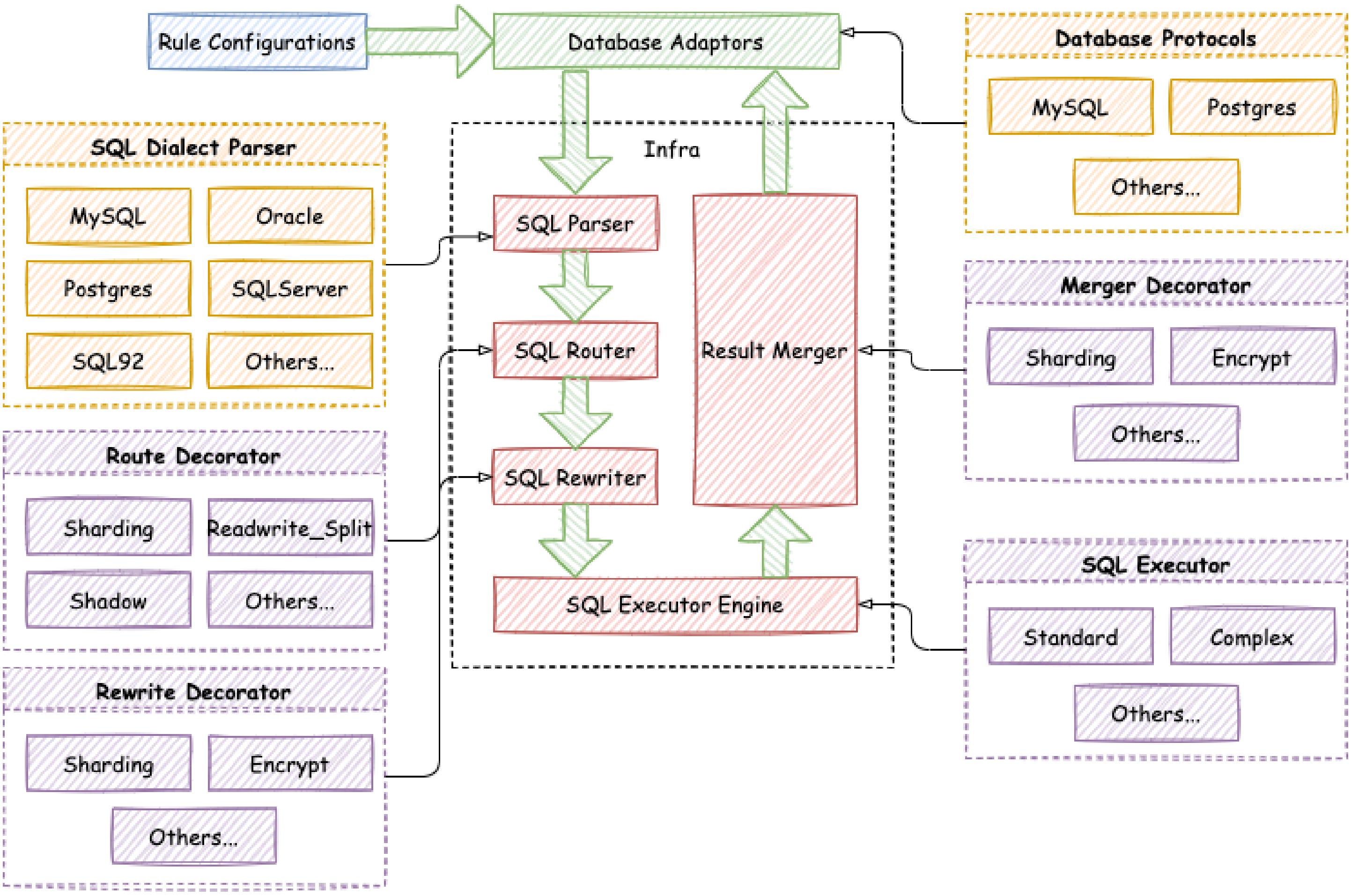


Fig. 1: Pluggable Platform

The pluggable architecture of DBPlusEngine are composed by L1 Kernel Layer, L2 Feature Layer and L3 Ecosystem Layer.

### 1.3.1 L1 Kernel Layer

An abstraction of the database’s basic capabilities. All components are required and the specific implementation can be replaced by pluggable way. It includes a query optimizer, distributed transaction engine, distributed execution engine, authority engine and scheduling engine.

### 1.3.2 L2 Feature Layer

Used to provide enhanced capability. All components are optional and can contain zero or multiple components. Components isolate each other and multiple components can be used together simultaneously. It includes data sharding, read/write splitting, database high availability, data encryption, shadow database and so on. User-defined features can be fully customized and extended for the top-level interface defined by Apache ShardingSphere without changing the kernel’s code.

### 1.3.3 L3 Ecosystem Layer

Used to integrate into the current database ecosystem. It includes database protocol, SQL parser and storage adapter.

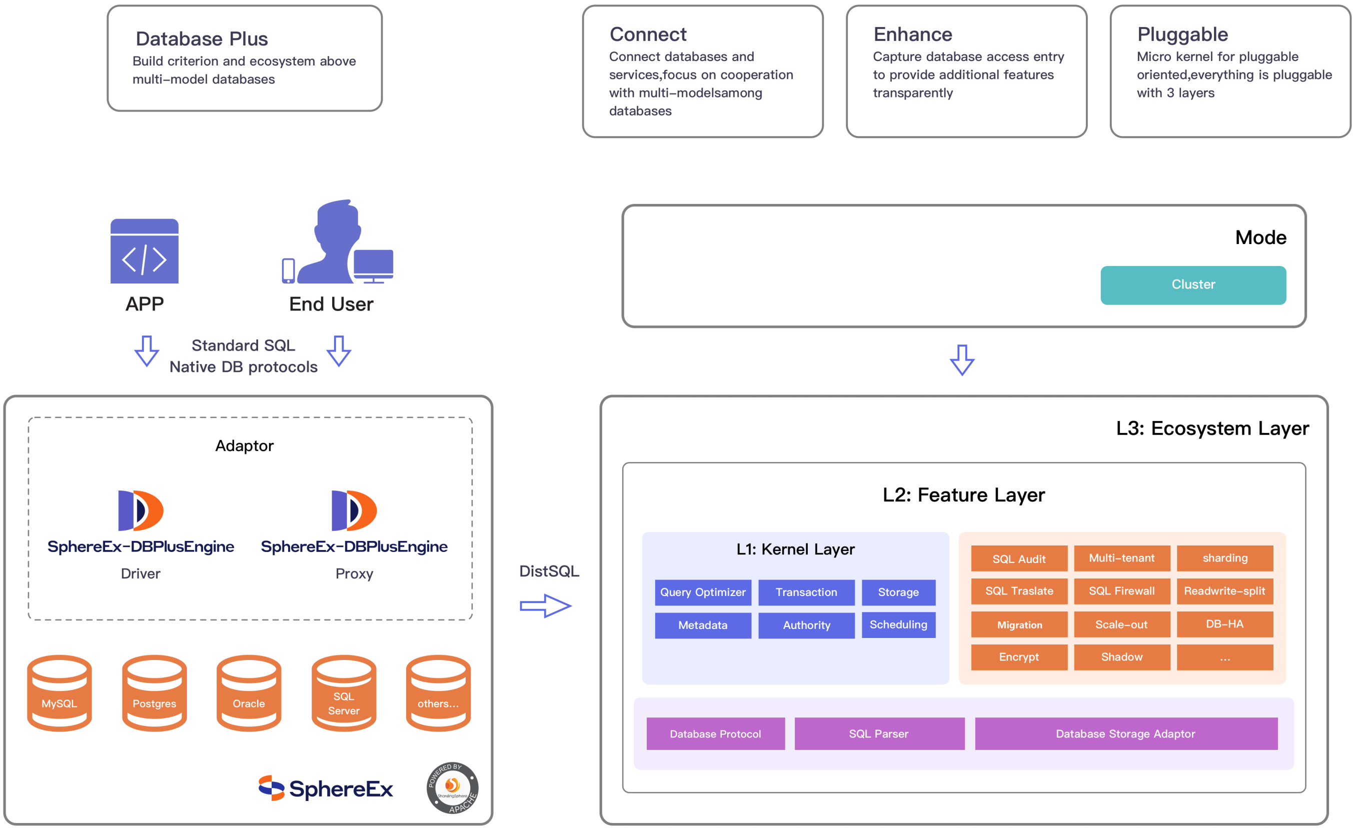


Fig. 2: Overview

## 1.4 Advantages

- **Build a standard layer and ecosystem above heterogeneous databases.**

DBPlusEngine is positioned as a Database Plus, and builds a standard layer and ecosystem above heterogeneous databases. It focuses on how to reuse existing databases and their respective upper layer, rather than creating a new database. DBPlusEngine stands at the upper level of the database and focuses on enhancing collaboration and compatibility between databases.

- **Provides extensions and enhancements to legacy relational databases.**

SphereEx-DBPlusEngine is designed to leverage the compute and storage power of existing relational databases in distributed scenarios. This is not an entirely new relational database. Relational databases still have a huge market share today, and are the cornerstone of enterprises core systems. Relational databases will be difficult to replace in the future, and thus we are more focused on providing incremental upgrades to the original databases, rather than replacing them.

- **DBPlusEngine supports multi-state access.**

DBPlusEngine-Driver uses a decentralized architecture that shares resources with applications.

DBPlusEngine-Proxy adopts independent application deployment.

DBPlusEngine is applicable to standard database access mode and provides support for multiple languages. It is suitable for OLTP and some OLAP scenarios.

## 1.5 The differences between the open source edition and the commercial edition

| Module | Level One Function | Level Two Function | Open Source | Commercial |
|---|---|---|---|---|
| Kernel Function | Dialect Compatibility | MySQL | √ | √ |
| | | PostgreSQL | √ | √ |
| | | Oracle | √ | √ |
| | | SQL Server | √ | √ |
| | | openGauss | √ | √ |
| | | SQL 92/2003 standard | √ | √ |
| | SQL Extension | SQL Hint | √ | √ |
| | | SQL dialect conversion | √ | √ |
| | | DistSQL | √ | √ |
| | Distributed Transaction | XA transaction | √ | √ |
| | | Flexible transaction | √ | √ |
| | | Global transaction | × | √ |
| | Query Optimizer | Native Query Optimization | √ | √ |
| | | Federated query optimization | √ | √ |
| Plugin Function | Data Sharding | Basic sharding strategy | √ | √ |
| | | Customized sharding strategy | × | √ |
| | | Automated sharding management | × | √ |
| | | Pre sharding capability | × | √ |
| | | Automatic data redistribution | × | √ |
| | | Distributed object | √ | √ |
| | Data Security (decryption) | Data storage encryption | √ | √ |
| | | Open source encryption algorithm | √ | √ |
| | | Commercial encryption algorithm | × | √ |
| | | Simple dense state calculation | √ | √ |
| | | Complex dense state calculation | × | √ |
| | Data Security (authority) | User Management | √ | √ |
| | | Role Management | × | √ |

continues on next page

Table 1 – continued from previous page

| Module | Level One Function | Level Two Function | Open Source | Commercial |
|---|---|---|---|---|
| | | Database table level authority control | × | √ |
| | | Row and column level authority control | × | √ |
| | | LDAP docking | × | √ |
| | | Third party authority docking | × | √ |
| | Behavior Security (AUDIT) | User behavior audit (afterwards) | × | √ |
| | | Proactive audit (beforehand) | × | √ |
| | Flexible Scheduling | Basic migration | √ | √ |
| | | Distributed high performance migration | × | √ |
| | | Data consistency verification | √ | √ |
| | | Enhanced flow switching and control | × | √ |
| | Data Isolation | Test production data isolation | √ | √ |
| | | Test data traceability | √ | × |
| Executive Function | Metadata Management | Online strong consistency change | × | √ |
| | | DDL multi version control | × | √ |
| | High Availability | Compute node high availability | √ | √ |
| | | Storage node high availability | × | √ |
| | | Fault perception and self-healing | √ | √ |
| | Monitoring Diagnosis | Host level monitoring | × | √ |
| | | Component level monitoring | √ | √ |
| | | Statement level monitoring (slow query) | √ | √ |
| | | Enhance monitoring indicators | × | √ |
| | | Monitoring alarm docking | × | √ |
| | | Link tracking (APM) | √ | √ |
| Delivery Capability | Delivery of Products | ShardingSphere-JDBC | √ | √ |
| | | ShardingSphere-Proxy | √ | √ |
| | | SphereEx-Boot(command line tool) | √ | √ |
| | | SphereEx-Console(console) | × | √ |
| | | Compatibility assessment tool | × | √ |
| | Support mode | Community support | √ | √ |
| | | Business support | × | √ |
| | Service Content | Problem solving | √ | √ |
| | | Architecture consulting | × | √ |
| | | Training | × | √ |
| | | Remote fault support | × | √ |
| | | Field fault support | × | √ |
| | | Installation and upgrade services | × | √ |
| | | On-site support services | × | √ |

## 1.6 Application Scenarios

### 1.6.1 Independent DBPlus Engine-Driver products deployment

As the first product and the predecessor of DBPlusEngine, DBPlusEngine-Driver is a lightweight Java framework that provides extra services at Java JDBC layer.

With the client end connecting directly to the database, it provides services in the form of jar and requires no extra deployment and dependence. It can be considered as an enhanced JDBC driver, fully compatible with JDBC and all kinds of ORM frameworks.

- Applicable to any ORM framework based on JDBC, such as JPA, Hibernate, Mybatis, Spring JDBC Template or direct use of JDBC.
- Supports any third-party database connection pool, such as DBCP, C3P0, BoneCP, HikariC.

- Supports any kind of JDBC standard database: MySQL, PostgreSQL, Oracle, SQLServer and any JDBC adapted databases.

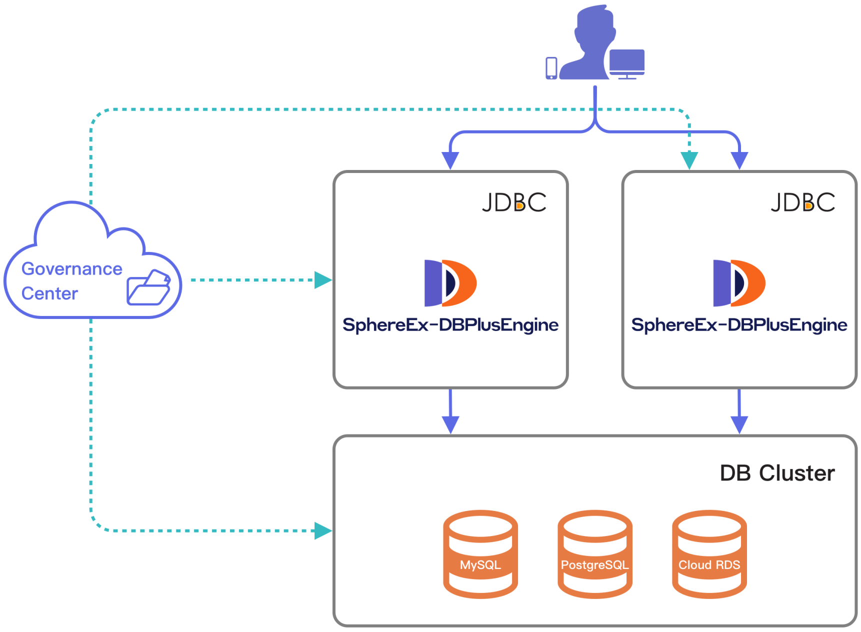


Fig. 3: ShardingSphere-JDBC Architecture

| | DBPlusEngine-Driver | DBPlusEngine-Proxy |
|---|---|---|
| Database | Any | MySQL/PostgreSQL |
| Connections Count Cost | More | Less |
| Supported Languages | Java Only | Any |
| Performance | Low loss | Relatively High loss |
| Decentralization | Yes | No |
| Static Entry | No | Yes |

DBPlusEngine-Driver is suitable for java application.

### 1.6.2 Independent DBPlusEngine-Proxy products deployment

DBPlusEngine-Proxy is the second product of DBPlusEngine. It is a transparent database proxy, providing a database server that encapsulates database binary protocol to support heterogeneous languages. Currently, MySQL and PostgreSQL (compatible with PostgreSQL-based databases, such as openGauss) versions are provided.

It can use any kind of terminal (such as MySQL Command Client, MySQL Workbench, etc.) that is compatible with MySQL or PostgreSQL protocols to operate data, making it friendlier to DBAs

- Totally transparent to applications, it can be used directly as MySQL/PostgreSQL.
- Applicable to any kind of client end that is compatible with MySQL/PostgreSQL protocol.

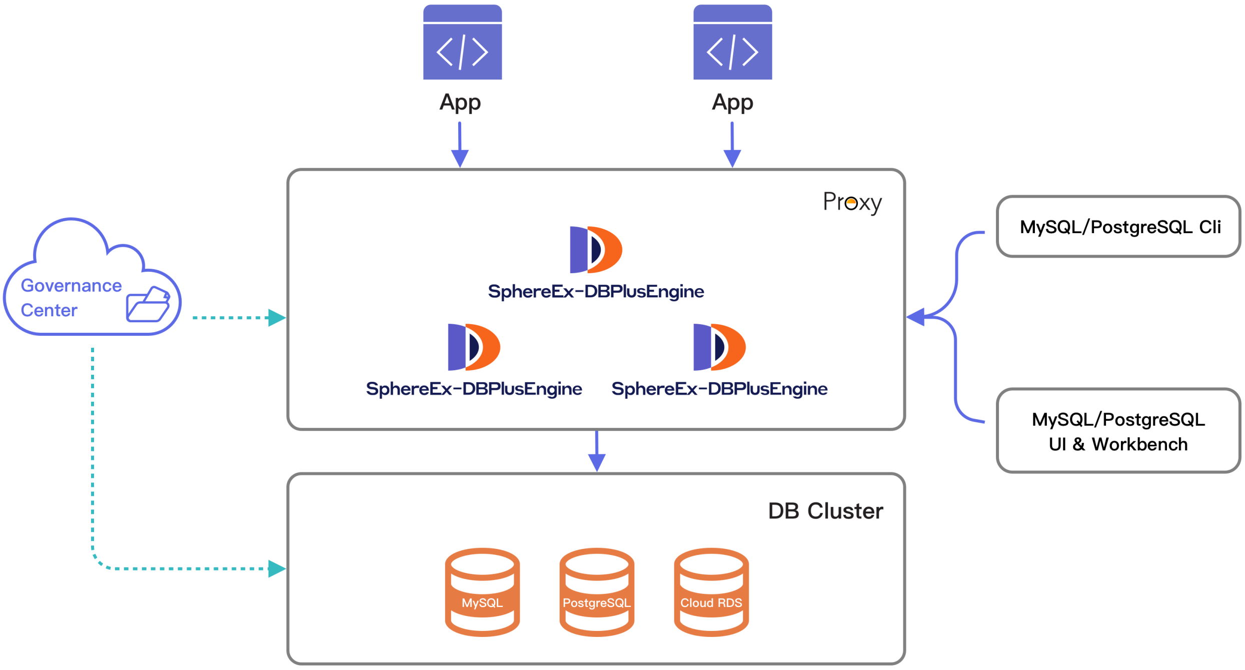


Fig. 4: ShardingSphere-Proxy Architecture

| | DBPlusEngine-Driver | DBPlusEngine-Proxy |
|---|---|---|
| Database | Any | MySQL/PostgreSQL |
| Connections Count Cost | High | Low |
| Supported Languages | Java Only | Any |
| Performance | Low loss | Relatively high loss |
| Decentralization | Yes | No |
| Static Entry | No | Yes |

The advantages of DBPlusEngine-Proxy lie in supporting heterogeneous languages and providing operational entries for DBAs.

### 1.6.3 Hybrid deployment with DBPlusEngine-Driver and DBPlusEngine-Proxy

DBPlusEngine is an ecosystem consisting of multiple endpoints together. Through a mixed use of DBPlusEngine-Driver and DBPlusEngine-Proxy, and a unified sharding strategy by the same registry center, DBPlusEngine can build an application system that is applicable to all kinds of scenarios. Architects can conveniently adjust the system architecture to suit their business requirements.

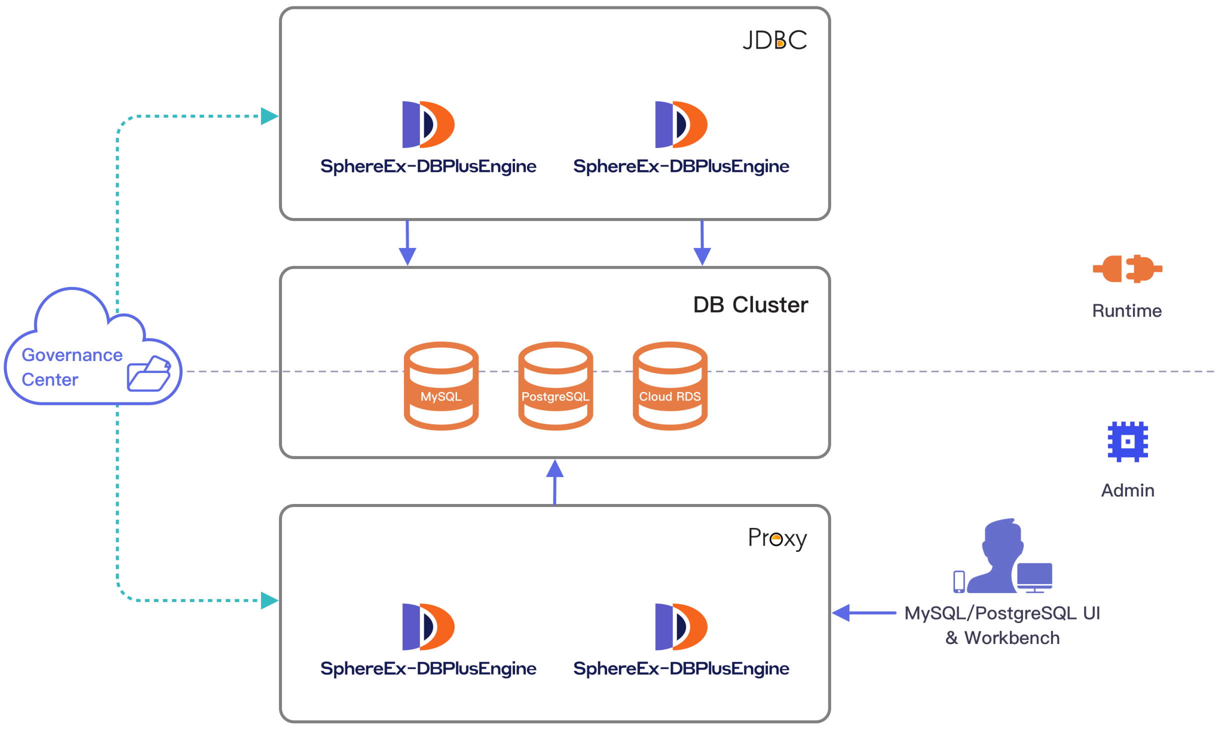


Fig. 5: ShardingSphere Hybrid Architecture

# 2 Features

DBPlusEngine provides a variety of features, from database kernel and database database solution, to applications closed features.

There is no boundary for these features, and we warmly welcome more open source engineers to join the community and provide exciting ideas and features.

- Traffic Dual Routing (Commercial Edition)
- Authority Control (Commercial Edition)

## 2.1 DB Compatibility

### 2.1.1 Background

Thanks to technology innovation, an increasing number of applications are established in new fields. This is causing a rapid data increase, leading to fast innovation for data storage and computing methods.

Transaction, big data, association analysis, Internet of Things and other scenarios have emerged quickly, resulting in a single database not being applicable to all application scenarios anymore. At the same time, it has become normal for similar scenarios to use different databases.

The trend of database fragmentation is emerging.

### 2.1.2 Challenges

There is no unified database access protocol and SQL dialect, as well as the maintenance and monitoring methods differences by various databases, learning and maintenance cost of developers and DBAs are increasing rapidly. Improving the compatibility with the original database is the premise for providing incremental services on it.

The compatibility between SQL dialect and database protocol is the key point to improve database compatibility.

### 2.1.3 Goal

**The goal of database compatibility for Apache ShardingSphere is to minimize or eliminate the differences between various databases for the user.**

### 2.1.4 SQL Parser

SQL is the standard operation language between users and databases. SQL Parse engine parses SQL into an abstract syntax tree to provide to Apache ShardingSphere for understanding and implementing the add-on features.

It supports SQL dialect for MySQL, PostgreSQL, SQLServer, Oracle, openGauss and SQL that conform to the SQL92 specification. Due to the complexity of SQL syntax, there are still some SQL that are not yet supported.

This chapter lists unsupported SQLs as a reference for users.

We will try best to support the unavailable SQLs in future versions.

#### MySQL

The following list includes unsupported SQL for MySQL:

| SQL |
|---|
| CLONE LOCAL DATA DIRECTORY = ‘clone_dir’ |
| INSTALL COMPONENT ‘file://component1’, ‘file://component2’ |
| UNINSTALL COMPONENT ‘file://component1’, ‘file://component2’ |
| REPAIR TABLE t_order |
| OPTIMIZE TABLE t_order |
| CHECKSUM TABLE t_order |
| CHECK TABLE t_order |
| SET RESOURCE GROUP group_name |
| DROP RESOURCE GROUP group_name |
| CREATE RESOURCE GROUP group_name TYPE = SYSTEM |
| ALTER RESOURCE GROUP rg1 VCPU = 0-63 |

#### openGauss

The following list includes unsupported SQL for openGauss:

| SQL |
|---|
| CREATE type avg_state AS (total bigint, count bigint); |
| CREATE AGGREGATE my_avg(int4) (stype = avg_state, sfunc = avg_transfn, finalfunc = avg_finalfn) |
| CREATE TABLE agg_data_2k AS SELECT g FROM generate_series(0, 1999) g; |
| CREATE SCHEMA alt_nsp1; |
| ALTER AGGREGATE alt_agg3(int) OWNER TO regress_alter_generic_user2; |
| CREATE CONVERSION alt_conv1 FOR ‘LATIN1’ TO ‘UTF8’ FROM iso8859_1_to_utf8; |
| CREATE FOREIGN DATA WRAPPER alt_fdw1 |
| CREATE SERVER alt_fserv1 FOREIGN DATA WRAPPER alt_fdw1 |
| CREATE LANGUAGE alt_lang1 HANDLER plpgsql_call_handler |
| CREATE STATISTICS alt_stat1 ON a, b FROM alt_regress_1 |
| CREATE TEXT SEARCH DICTIONARY alt_ts_dict1 (template=simple) |
| CREATE RULE def_view_test_ins AS ON INSERT TO def_view_test DO INSTEAD INSERT INTO def_test SELECT new.* |
| ALTER TABLE alterlock SET (toast.autovacuum_enabled = off) |
| CREATE PUBLICATION pub1 FOR TABLE alter1.t1, ALL TABLES IN SCHEMA alter2 |

### PostgreSQL

The following list includes unsupported SQL for PostgreSQL:

| SQL |
|---|
| CREATE type avg_state AS (total bigint, count bigint); |
| CREATE AGGREGATE my_avg(int4) (stype = avg_state, sfunc = avg_transfn, finalfunc = avg_finalfn) |
| CREATE TABLE agg_data_2k AS SELECT g FROM generate_series(0, 1999) g; |
| CREATE SCHEMA alt_nsp1; |
| ALTER AGGREGATE alt_agg3(int) OWNER TO regress_alter_generic_user2; |
| CREATE CONVERSION alt_conv1 FOR ‘LATIN1’ TO ‘UTF8’ FROM iso8859_1_to_utf8; |
| CREATE FOREIGN DATA WRAPPER alt_fdw1 |
| CREATE SERVER alt_fserv1 FOREIGN DATA WRAPPER alt_fdw1 |
| CREATE LANGUAGE alt_lang1 HANDLER plpgsql_call_handler |
| CREATE STATISTICS alt_stat1 ON a, b FROM alt_regress_1 |
| CREATE TEXT SEARCH DICTIONARY alt_ts_dict1 (template=simple) |
| CREATE RULE def_view_test_ins AS ON INSERT TO def_view_test DO INSTEAD INSERT INTO def_test SELECT new.* |
| ALTER TABLE alterlock SET (toast.autovacuum_enabled = off) |
| CREATE PUBLICATION pub1 FOR TABLE alter1.t1, ALL TABLES IN SCHEMA alter2 |

### SQLServer

The following list includes unsupported SQL for SQLServer:

TODO

### Oracle

The following list includes unsupported SQL for Oracle:

TODO

### SQL92

The following list includes unsupported SQL for SQL92:

TODO

## 2.1.5 DB Protocol

Apache ShardingSphere implements the MySQL and PostgreSQL protocols.

## 2.1.6 Feature Support

Apache ShardingSphere provides the database distributed collaboration capability, and abstracts part of the database features to the upper layer for unified management to reduce the user difficulty.

Therefore, for the unified provided features, the native SQL will no longer be transferred to the database, and it will be prompted that the operation is not supported.

User can use the feature provided by ShardingSphere to replace it.

This chapter has listed unsupported database features and related SQLs reference for users.

There are some unsupported SQLs maybe missing, and the list is continuously updated.

## MySQL

The following is the list of unsupported SQL:

### User & Role

| SQL |
|---|
| CREATE USER ‘finley’ @ ‘localhost’ IDENTIFIED BY ‘password’ |
| ALTER USER ‘finley’ @ ‘localhost’ IDENTIFIED BY ‘new_password’ |
| DROP USER ‘finley’ @ ‘localhost’ ; |
| CREATE ROLE ‘app_read’ |
| DROP ROLE ‘app_read’ |
| SHOW CREATE USER finley |
| SET PASSWORD = ‘auth_string’ |
| SET ROLE DEFAULT; |

### Authorization

| SQL |
|---|
| GRANT ALL ON db1.* TO ‘jeffrey’ @ ‘localhost’ |
| GRANT SELECT ON world.* TO ‘role3’ ; |
| GRANT ‘role1’ , ‘role2’ TO ‘user1’ @ ‘localhost’ |
| REVOKE INSERT ON . FROM ‘jeffrey’ @ ‘localhost’ |
| REVOKE ‘role1’ , ‘role2’ FROM ‘user1’ @ ‘localhost’ |
| REVOKE ALL PRIVILEGES, GRANT OPTION FROM user_or_role |
| SHOW GRANTS FOR ‘jeffrey’ @ ‘localhost’ |
| SHOW GRANTS FOR CURRENT_USER |
| FLUSH PRIVILEGES |

## PostgreSQL

The following list includes unsupported SQL for PostgreSQL:

TODO

## SQLServer

The following list includes the unsupported SQL list for SQLServer:

TODO

#### Oracle

The following list includes unsupported SQL for Oracle:

TODO

#### SQL92

The following list includes unsupported SQL for SQL92:

TODO

## 2.2 Management

### 2.2.1 Background

Unified management capability of cluster perspective, and control capability of individual components are necessary in modern database system.

### 2.2.2 Challenges

The challenges are unified management of centralized management, and operation in case of single node in failure.

Centralized management uniformly manages the state of database storage nodes and middleware computing nodes, and can detect the latest updates in the distributed environment in real time, further provide information with control and scheduling.

In the overload traffic scenario, circuit breaker and request limiting for a node to ensure whole database cluster can run continuously is a challenge to control ability of a single node.

### 2.2.3 Goal

The goal of DBPlusEngine management module is to achieve the integrated management ability from database to computing node, and provide control ability for components in case of failure.

### 2.2.4 Core Concept

#### Circuit Breaker

Fuse connection between DBPlusEngine and the database. When a DBPlusEngine node exceeds the max load, stop the node’s access to the database, so that the database can ensure sufficient resources to provide services for other DBPlusEngine nodes.

### Request Limit

In case of overload requests, open request limiting to protect some requests can still respond quickly.

## 2.3 Sharding

### 2.3.1 Background

The traditional solution that stores all the data in one concentrated node has hardly satisfied the requirement of massive data scenario in three aspects - performance, availability and operation cost.

In performance, the relational database mostly uses B+ tree index. When the data amount exceeds the threshold, deeper index will increase the disk IO access number, and weaken the performance of query. At the same time, high concurrency requests also make the centralized database to be the greatest limitation of the system.

In availability, capacity can be expanded at a relatively low cost and any extent with stateless service, which can make all the pressure, at last, fall on the database. But the single data node or simple primary-replica structure has been harder and harder to take these pressures. Therefore, database availability has become the key to the whole system.

From the aspect of operation costs, when the data in a database instance has reached above the threshold, DBA’s operation pressure will also increase. The time cost of data backup and data recovery will be more uncontrollable with increasing amount of data. Generally, it is a relatively reasonable range for the data in single database case to be within 1TB.

Under the circumstance that traditional relational databases cannot satisfy the requirement of the Internet, there are more and more attempts to store the data in native distributed NoSQL. But its incompatibility with SQL and imperfection in ecosystem block it from defeating the relational database in the competition, so the relational database still holds an unshakable position.

Sharding refers to splitting the data in one database and storing them in multiple tables and databases according to some certain standard, so that the performance and availability can be improved. Both methods can effectively avoid the query limitation caused by data exceeding affordable threshold. What’s more, database sharding can also effectively disperse TPS. Table sharding, though cannot ease the database pressure, can provide possibilities to transfer distributed transactions to local transactions, since cross-database upgrades are once involved, distributed transactions can turn pretty tricky sometimes. The use of multiple primary-replica sharding method can effectively avoid the data concentrating on one node and increase the architecture availability.

Splitting data through database sharding and table sharding is an effective method to deal with high TPS and mass amount data system, because it can keep the data amount lower than the threshold and evacuate the traffic. Sharding method can be divided into vertical sharding and horizontal sharding.

### Vertical Sharding

According to business sharding method, it is called vertical sharding, or longitudinal sharding, the core concept of which is to specialize databases for different uses. Before sharding, a database consists of many tables corresponding to different businesses. But after sharding, tables are categorized into different databases according to business, and the pressure is also separated into different databases. The diagram below has presented the solution to assign user tables and order tables to different databases by vertical sharding according to business need.

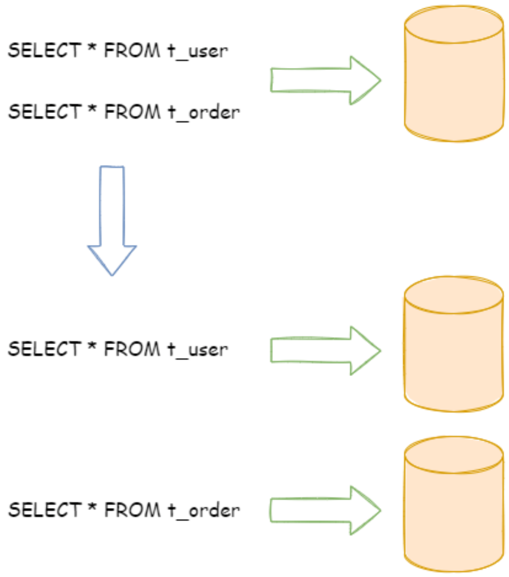


Fig. 1: Vertical Sharding

Vertical sharding requires to adjust the architecture and design from time to time. Generally speaking, it is not soon enough to deal with fast changing needs from Internet business and not able to really solve the single-node problem. it can ease problems brought by the high data amount and concurrency amount, but cannot solve them completely. After vertical sharding, if the data amount in the table still exceeds the single node threshold, it should be further processed by horizontal sharding.

### Horizontal Sharding

Horizontal sharding is also called transverse sharding. Compared with the categorization method according to business logic of vertical sharding, horizontal sharding categorizes data to multiple databases or tables according to some certain rules through certain fields, with each sharding containing only part of the data. For example, according to primary key sharding, even primary keys are put into the 0 database (or table) and odd primary keys are put into the 1 database (or table), which is illustrated as the following diagram.

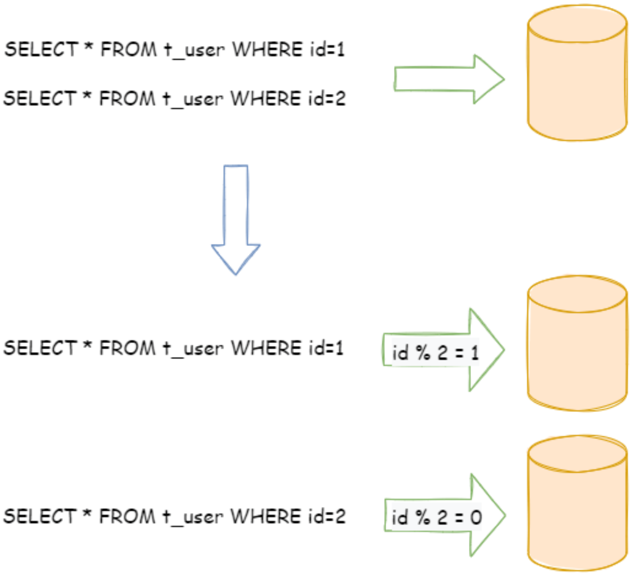

SELECT * FROM t_user WHERE id=1
SELECT * FROM t_user WHERE id=2
SELECT * FROM t_user WHERE id=1
id % 2 = 1
SELECT * FROM t_user WHERE id=2
id % 2 = 0

Fig. 2: Horizontal Sharding

Theoretically, horizontal sharding has overcome the limitation of data processing volume in single machine and can be extended relatively freely, so it can be taken as a standard solution to database sharding and table sharding.

### 2.3.2 Challenges

Though sharding has solved problems such as performance, availability and single-node backup and recovery, its distributed architecture has also introduced some new problems as acquiring profits.

One problem is that application development engineers and database administrators' operations become exceptionally laborious, when facing such scattered databases and tables. They should know exactly which database table is the one to acquire data from.

Another challenge is that, the SQL that runs rightly in single-node databases may not be right in the sharding database. The change of table name after sharding, or misconducts caused by operations such as pagination, order by or aggregated group by are just the case in point.

Cross-database transaction is also a tricky thing that distributed databases need to deal. Fair use of sharding tables can also lead to the full use of local transactions when single-table data amount decreases. Troubles brought by distributed transactions can be avoided by the wise use of different tables in the same database. When cross-database transactions cannot be avoided, some businesses still need to keep transactions consistent. Internet giants have not massively adopted XA based distributed transactions since they are not able to ensure its performance in high-concurrency situations. They usually replace strongly consistent transactions with eventually consistent soft state.

## 2.3.3 Goal

**The main design goal of Apache ShardingSphere’ s data sharding module is to try to reduce the influence of sharding, in order to let users use horizontal sharding database group like one database.**

## 2.3.4 Core Concept

### Overview

This chapter introduces core concepts of data sharding.

### Table

Table is the core concept of transparent data sharding. There are diversified tables provided for different data sharding requirements by Apache ShardingSphere.

### Logic Table

The logical name of the horizontal sharding databases (tables) with the same schema, it is the logical table identification in SQL. For instance, the data of order is divided into 10 tables according to the last number of the primary key, and they are from t_order_0 to t_order_9, whose logic name is t_order.

### Actual Table

The physical table that really exists in the horizontal sharding database, i.e., t_order_0 to t_order_9 in the instance above.

### Binding Table

It refers to the primary table and the joiner table with the same sharding rules. When using binding tables in multi-table correlating query, you must use the sharding key for correlation, otherwise Cartesian product correlation or cross-database correlation will appear, which will affect query efficiency. For example, t_order and t_order_item are both sharded by order_id, and use order_id to correlate, so they are binding tables with each other. Cartesian product correlation will not appear in the multi-tables correlating query, so the query efficiency will increase greatly. Take this one for example, if SQL is:

```sql
SELECT i.* FROM t_order o JOIN t_order_item i ON o.order_id=i.order_id WHERE o.order_id in (10, 11);
```

When binding table relations are not configured, suppose the sharding key order_id routes value 10 to sharding 0 and value 11 to sharding 1, there will be 4 SQLs in Cartesian product after routing:

```sql
SELECT i.* FROM t_order_0 o JOIN t_order_item_0 i ON o.order_id=i.order_id WHERE o.order_id in (10, 11);

SELECT i.* FROM t_order_0 o JOIN t_order_item_1 i ON o.order_id=i.order_id WHERE o.order_id in (10, 11);

SELECT i.* FROM t_order_1 o JOIN t_order_item_0 i ON o.order_id=i.order_id WHERE o.order_id in (10, 11);

SELECT i.* FROM t_order_1 o JOIN t_order_item_1 i ON o.order_id=i.order_id WHERE o.order_id in (10, 11);
```

With binding table configuration and use order_id to correlate, there should be 2 SQLs after routing:

```sql
SELECT i.* FROM t_order_0 o JOIN t_order_item_0 i ON o.order_id=i.order_id WHERE o.order_id in (10, 11);

SELECT i.* FROM t_order_1 o JOIN t_order_item_1 i ON o.order_id=i.order_id WHERE o.order_id in (10, 11);
```

In them, table t_order in the left end of FROM will be taken by ShardingSphere as the primary table of query. In a similar way, ShardingSphere will also take table t_order in the left end of FROM as the primary table of the whole binding table. All the route computations will only use the sharding strategy of the primary table, so sharding computation of t_order_item table will use the conditions of t_order. Due to this, sharding keys in binding tables should be totally identical.

### Broadcast Table

It refers to tables that exist in all sharding database sources. The schema and data must consist in each database. It can be applied to the small data volume that needs to correlate with big data tables to query, dictionary table for example.

### Single Table

It refers to only one table that exists in all sharding database sources. It is suitable for little data in table without sharding.

### Data Node

As the atomic unit of sharding, it consists of data source name and actual table name, e.g. ds_0.t_order_0.

Mapping relationships between logic tables and actual tables, it can be divided into two kinds: uniform topology and user-defined topology.

### Uniform topology

It means that tables are evenly distributed in each data source, for example:

```
db0
  ├──── t_order0
  └──── t_order1
db1
  ├──── t_order0
  └──── t_order1
```

The data node configurations will be as follows:

```
db0.t_order0, db0.t_order1, db1.t_order0, db1.t_order1
```

### User-defined topology

It means that tables are distributed with certain rules, for example:

```
db0
  ├──── t_order0
  └──── t_order1
db1
  ├──── t_order2
  ├──── t_order3
  └──── t_order4
```

The data node configurations will be as follows:

```
db0.t_order0, db0.t_order1, db1.t_order2, db1.t_order3, db1.t_order4
```

### Sharding

#### Sharding Key

Column used to determine database (table) sharding. For example, in last number modulo of order ID sharding, order ID is taken as the sharding key. The full route executed when there is no sharding column in SQL has a poor performance. Besides single sharding column, Apache ShardingSphere also supports multiple sharding columns.

#### Sharding Algorithm

Data sharding can be achieved by sharding algorithms through =, >=, <=, >, <, BETWEEN and IN. It can be implemented by developers themselves, or using built-in syntactic sugar of Apache ShardingSphere, with high flexibility.

#### Auto Sharding Algorithm

It provides syntactic sugar for sharding algorithm. It used to manage all data nodes automatically, users do not care about the topology of physical data nodes. It includes lots of implementation for Mod, Hash, Range and Time Interval etc.

#### User-Defined Sharding Algorithm

It provides interfaces for developers to implement the sharding algorithm related to business implementation, and allows users to manage the physical topology physical data nodes by themselves. It includes:

- Standard Sharding Algorithm

It processes the sharding case in which single sharding keys =, IN, BETWEEN AND, >, <, >=, <= are used.

- Complex Keys Sharding Algorithm

It processes the sharding case in which multiple sharding keys are used. It has a relatively complex logic that requires developers to deal by themselves.

- Hint Sharding Algorithm

It processes the sharding case in which Hint is used.

#### Sharding Strategy

It includes the sharding key and the sharding algorithm, and the latter one is extracted out for its independence. Only sharding key + sharding algorithm can be used in sharding operation.

#### SQL Hint

In the case that the sharding column is not decide by SQL but other external conditions, SQL hint can be used to inject sharding value. For example, databases are sharded according to the staff's ID, but column does not exist in the database. SQL Hint can be used by two ways, Java API and SQL comment (TODO). Please refer to Hint for more details.

## Inline Expression

### Motivation

Configuration simplicity and unity are two main problems that inline expression intends to solve.

In complex sharding rules, with more data nodes, a large number of configuration repetitions make configurations difficult to maintain. Inline expressions can simplify data node configuration work.

Java code is not helpful in the unified management of common configurations. By writing sharding algorithms with inline expressions, users can store rules together, making them easier to be browsed and stored.

### Syntax Explanation

The use of inline expressions is really direct. Users only need to configure ${ expression } or $->{ expression } to identify them. ShardingSphere currently supports the configurations of data nodes and sharding algorithms. Inline expressions use Groovy syntax, which can support all kinds of operations, including inline expressions. For example:

${begin..end} means range

${[unit1, unit2, unit_x]} means enumeration

If there are many continuous ${ expression } or $->{ expression } expressions, according to each sub-expression result, the ultimate result of the whole expression will be in cartesian combination.

For example, the following inline expression:

```
${['online', 'offline']}_table${1..3}
```

Will be parsed as:

```
online_table1, online_table2, online_table3, offline_table1, offline_table2, offline_table3
```

### Configuration

### Data Node

For evenly distributed data nodes, if the data structure is as follow:

```
db0
  ├──── t_order0
  └──── t_order1
db1
  ├──── t_order0
  └──── t_order1
```

It can be simplified by inline expression as:

```
db${0..1}.t_order${0..1}
```

Or

```
db$->{0..1}.t_order$->{0..1}
```

For self-defined data nodes, if the data structure is:

```
db0
  ├──── t_order0
  └──── t_order1
db1
  ├──── t_order2
  ├── t_order3
  └── t_order4
```

It can be simplified by inline expression as:

```
db0.t_order${0..1},db1.t_order${2..4}
```

Or

```
db0.t_order$->{0..1},db1.t_order$->{2..4}
```

For data nodes with prefixes, inline expression can also be used to configure them flexibly, if the data structure is:

```
db0
  ├── t_order_00
  ├── t_order_01
  ├── t_order_02
  ├── t_order_03
  ├── t_order_04
  ├── t_order_05
  ├── t_order_06
  ├── t_order_07
  ├── t_order_08
  ├── t_order_09
  ├── t_order_10
  ├── t_order_11
  ├── t_order_12
  ├── t_order_13
  ├── t_order_14
  ├── t_order_15
  ├── t_order_16
  ├── t_order_17
  ├── t_order_18
  ├── t_order_19
  └── t_order_20
db1
  ├── t_order_00
  ├── t_order_01
  ├── t_order_02
  ├── t_order_03
  ├── t_order_04
  ├── t_order_05
  ├── t_order_06
  ├── t_order_07
  ├── t_order_08
  ├── t_order_09
  ├── t_order_10
  ├── t_order_11
  ├── t_order_12
  ├── t_order_13
  ├── t_order_14
  ├── t_order_15
  ├── t_order_16
  ├── t_order_17
  ├── t_order_18
  ├── t_order_19
  └── t_order_20
```

Users can configure separately, data nodes with prefixes first, those without prefixes later, and automatically combine them with the cartesian product feature of inline expressions. The example above can be simplified by inline expression as:

```
db${0..1}.t_order_0${0..9}, db${0..1}.t_order_${10..20}
```

Or

```
db$->{0..1}.t_order_0$->{0..9}, db$->{0..1}.t_order_$->{10..20}
```

### Sharding Algorithm

For single sharding SQL that uses = and IN, inline expression can replace codes in configuration.

Inline expression is a piece of Groovy code in essence, which can return the corresponding real data source or table name according to the computation method of sharding keys.

For example, sharding keys with the last number 0 are routed to the data source with the suffix of 0, those with the last number 1 are routed to the data source with the suffix of 1, the rest goes on in the same way. The inline expression used to indicate sharding algorithm is:

```
ds${id % 10}
```

Or

```
ds$->{id % 10}
```

### Distributed Primary Key

#### Motivation

In the development of traditional database software, the automatic sequence generation technology is a basic requirement. All kinds of databases have provided corresponding support for this requirement, such as MySQL auto-increment key, Oracle auto-increment sequence and so on. It is a tricky problem that there is only one sequence generated by different data nodes after sharding. Auto-increment keys in different physical tables in the same logic table cannot perceive each other and thereby generate repeated sequences. It is possible to avoid clashes by restricting the initiative value and increasing the step of auto-increment key. But introducing extra operation rules can make the solution lack integrity and scalability.

Currently, there are many third-party solutions that can solve this problem perfectly, (such as UUID and others) relying on some particular algorithms to generate unrepeated keys or introducing sequence generation services. We have provided several built-in key generators, such as UUID, SNOWFLAKE. Besides, we have also extracted a key generator interface to make users implement self-defined key generator.

### Built-In Key Generator

#### UUID

Use UUID.randomUUID() to generate the distributed key.

### SNOWFLAKE

Users can configure the strategy of each table in sharding rule configuration module, with default snowflake algorithm generating 64bit long integral data.

As the distributed sequence generation algorithm published by Twitter, snowflake algorithm can ensure sequences of different processes do not repeat and those of the same process are ordered.

#### Principle

In the same process, it makes sure that IDs do not repeat through time, or through order if the time is identical. At the same time, with monotonously increasing time, if servers are generally synchronized, generated sequences are generally assumed to be ordered in a distributed environment. This can guarantee the effectiveness in index field insertion, like the sequence of MySQL Innodb storage engine.

In the sequence generated with the snowflake algorithm, binary form has 4 parts, 1 bit sign, 41bit timestamp, 10bit work ID and 12bit sequence number from high to low.

- sign bit (1bit)

Reserved sign bit, constantly to be zero.

- timestamp bit (41bit)

41bit timestamp can contain 2 to the power of 41 milliseconds. One year can use 365 * 24 * 60 * 60 * 1000 milliseconds. We can see from the calculation:

```
Math.pow(2, 41) / (365 * 24 * 60 * 60 * 1000L);
```

The result is approximately equal to 69.73 years. Apache ShardingSphere snowflake algorithm starts from November 1st, 2016, and can be used until 2086, which we believe can satisfy the requirement of most systems.

- work ID bit (10bit)

The sign is the only one in Java process. If applied in distributed deployment, each work ID should be different. The default value is 0 and can be set through properties.

- sequence number bit (12bit)

The sequence number is used to generate different IDs in a millisecond. If the number generated in that millisecond exceeds 4,096 (2 to the power of 12), the generator will wait till the next millisecond to continue.

Please refer to the following picture for the detailed structure of snowflake algorithm sequence.

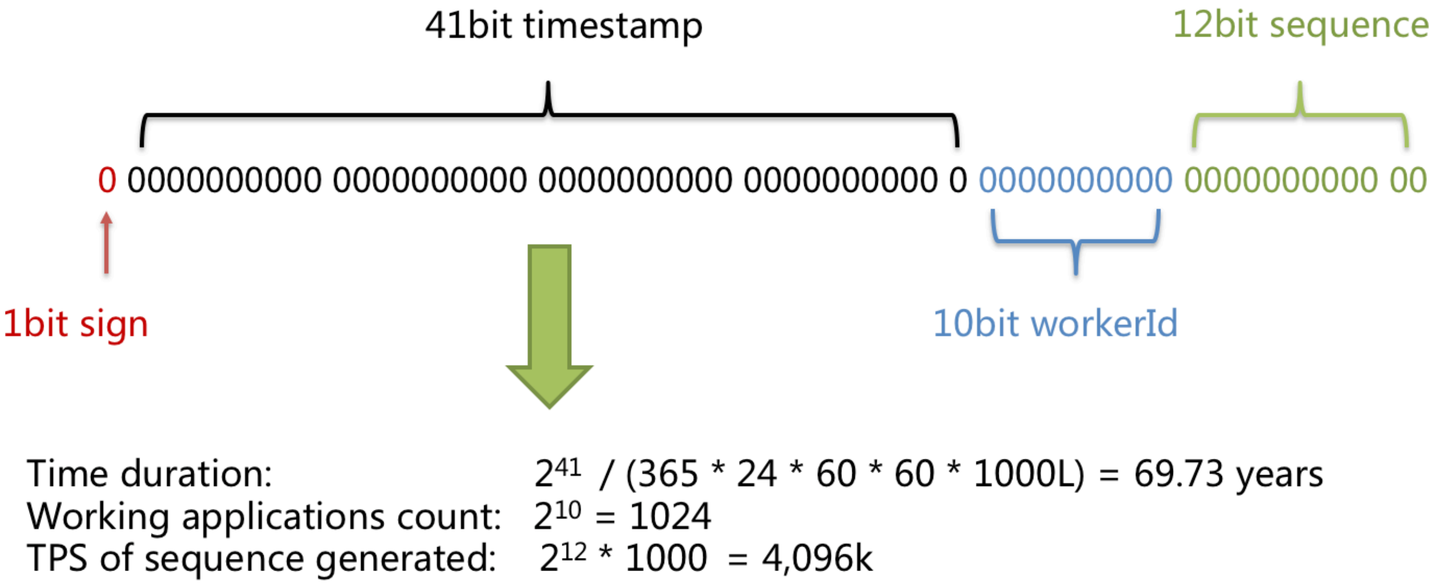

41bit timestamp
12bit sequence
0 0000000000 0000000000 0000000000 0000000000 0 0000000000 0000000000 00
1bit sign
10bit workerId
Time duration: $2^{41}$ / (365 * 24 * 60 * 60 * 1000L) = 69.73 years
Working applications count: $2^{10}$ = 1024
TPS of sequence generated: $2^{12}$ * 1000 = 4,096k

Fig. 3: Snowflake

### Clock-Back

The clock-back of server can generate repeated sequence, so the default distributed sequence generator has provided a maximum clock-back millisecond. If the clock-back time has exceeded it, the program will report error. If it is within the tolerance range, the generator will wait till after the last generation time and then continue to work. The default maximum clock-back millisecond is 0 and can be set through properties.

## Hint Sharding Route

### Motivation

Apache ShardingSphere can be compatible with SQL by parsing SQL statements and extracting columns and values to shard. If SQL does not have sharding conditions, it is impossible to shard without full data node route.

In some applications, sharding conditions are not in SQL but in the external business logic. So it requires to designate sharding result externally, which is referred to as Hint in ShardingSphere.

### Mechanism

Apache ShardingSphere uses ThreadLocal to manage sharding key values. Users can program to add sharding conditions to HintManager, but the condition is only effective within the current thread.

In addition to the programming method, Apache ShardingSphere is able to cite Hint through special notation in SQL, so that users can use that function in a more transparent way.

The SQL designated with sharding hint will ignore the former sharding logic but directly route to the designated node.

### 2.3.5 Use Norms

#### Background

Although SphereEx-DBPlusEngine intends to be compatible with all the SQLs and standalone databases, the distributed scenario has created even more complex situations for the database. SphereEx-DBPlusEngine wants to solve massive data OLTP problem first, and eventually complete relevant OLAP support.

#### SQL

#### SQL Support Status

Compatible with all regular SQL when **routing to single data node**; **The SQL routing to multiple data nodes** is pretty complex, it divides the scenarios as totally supported, experimental supported and unsupported.

#### Totally Supported

Fully support DML, DDL, DCL, TCL and most regular DAL. Support complex query with pagination, DISTINCT, ORDER BY, GROUP BY, aggregation and table JOIN.

#### Regular Query

- SELECT Clause

```
SELECT select_expr [, select_expr ...] FROM table_reference [, table_reference ...]
[WHERE predicates]
[GROUP BY {col_name | position} [ASC | DESC], ...]
[ORDER BY {col_name | position} [ASC | DESC], ...]
[LIMIT {[offset,] row_count | row_count OFFSET offset}]
```

- select_expr

```
*
| [DISTINCT] COLUMN_NAME [AS] [alias]
| (MAX | MIN | SUM | AVG)(COLUMN_NAME | alias) [AS] [alias]
| COUNT(* | COLUMN_NAME | alias) [AS] [alias]
```

- table_reference

```
tbl_name [AS] alias] [index_hint_list]
| table_reference ([INNER] | {LEFT|RIGHT} [OUTER]) JOIN table_factor [JOIN ON conditional_expr | USING (column_list)]
```

#### Subquery

Stable supported when sharding keys are using in both subquery and outer query, and values of sharding keys are the same.

For example:

```
SELECT * FROM (SELECT * FROM t_order WHERE order_id = 1) o WHERE o.order_id = 1;
```

Stable supported for subquery with pagination.

For example:

```sql
SELECT * FROM (SELECT row_.*, rownum rownum_ FROM (SELECT * FROM t_order) row_ WHERE rownum <= ?) WHERE
rownum > ?;
```

### Sharding value in expression

Sharding value in calculated expressions will lead to full routing.

For example, if create_time is sharding value:

```sql
SELECT * FROM t_order WHERE to_date(create_time, 'yyyy-mm-dd') = '2019-01-01';
```

### Experimental Supported

Experimental support specifically refers to use of Federation execution engine. The engine still in rapid development, basically available to users, but it still needs lots of optimization. It is an experimental product.

### Subquery

Experimental supported when sharding keys are not using for both subquery and outer query, or values of sharding keys are not the same.

For example:

```sql
SELECT * FROM (SELECT * FROM t_order) o;

SELECT * FROM (SELECT * FROM t_order) o WHERE o.order_id = 1;

SELECT * FROM (SELECT * FROM t_order WHERE order_id = 1) o;

SELECT * FROM (SELECT * FROM t_order WHERE order_id = 1) o WHERE o.order_id = 2;
```

### Join with cross databases

When tables in a join query are distributed on different database instances, sql statement will be supported by Federation execution engine. Assuming that t_order and t_order_item are sharding tables with multiple data nodes, and no binding table rules are configured, t_user and t_user_role are single tables that distributed on different database instances. Federation execution engine can support the following commonly used join query:

```sql
SELECT * FROM t_order o INNER JOIN t_order_item i ON o.order_id = i.order_id WHERE o.order_id = 1;

SELECT * FROM t_order o INNER JOIN t_user u ON o.user_id = u.user_id WHERE o.user_id = 1;

SELECT * FROM t_order o LEFT JOIN t_user_role r ON o.user_id = r.user_id WHERE o.user_id = 1;

SELECT * FROM t_order_item i LEFT JOIN t_user u ON i.user_id = u.user_id WHERE i.user_id = 1;

SELECT * FROM t_order_item i RIGHT JOIN t_user_role r ON i.user_id = r.user_id WHERE i.user_id = 1;

SELECT * FROM t_user u RIGHT JOIN t_user_role r ON u.user_id = r.user_id WHERE u.user_id = 1;
```

### Unsupported

CASE WHEN can not support as following:

- CASE WHEN containing sub-query
- CASE WHEN containing logical-table (instead of table alias)

### SQL Example

| Stable supported SQL | Necessary conditions |
|---|---|
| SELECT * FROM tbl_name | |
| SELECT * FROM tbl_name WHERE (col1 = ? or col2 = ?) and col3 = ? | |
| SELECT * FROM tbl_name WHERE col1 = ? ORDER BY col2 DESC LIMIT ? | |
| SELECT COUNT(*), SUM(col1), MIN(col1), MAX(col1), AVG(col1) FROM tbl_name WHERE col1 = ? | |
| SELECT COUNT(col1) FROM tbl_name WHERE col2 = ? GROUP BY col1 ORDER BY col3 DESC LIMIT ?, ? | |
| SELECT DISTINCT * FROM tbl_name WHERE col1 = ? | |
| SELECT COUNT(DISTINCT col1), SUM(DISTINCT col1) FROM tbl_name | |
| (SELECT * FROM tbl_name) | |
| SELECT * FROM (SELECT * FROM tbl_name WHERE col1 = ?) o WHERE o.col1 = ? | Subquery and outer query in same sharded data node after route |
| INSERT INTO tbl_name (col1, col2,⋯) VALUES (?, ?, ⋯.) | |
| INSERT INTO tbl_name VALUES (?, ?,⋯.) | |
| INSERT INTO tbl_name (col1, col2, ⋯) VALUES(1 + 2, ?, ⋯) | |
| INSERT INTO tbl_name (col1, col2, ⋯) VALUES (?, ?, ⋯.), (?, ?, ⋯.) | |
| INSERT INTO tbl_name (col1, col2, ⋯) SELECT col1, col2, ⋯FROM tbl_name WHERE col3 = ? | Inserted and selected table must be the same or binding tables |
| REPLACE INTO tbl_name (col1, col2, ⋯) SELECT col1, col2, ⋯FROM tbl_name WHERE col3 = ? | Replaced and selected table must be the same or binding tables |
| UPDATE tbl_name SET col1 = ? WHERE col2 = ? | |
| DELETE FROM tbl_name WHERE col1 = ? | |
| CREATE TABLE tbl_name (col1 int, ⋯) | |
| ALTER TABLE tbl_name ADD col1 varchar(10) | |
| DROP TABLE tbl_name | |
| TRUNCATE TABLE tbl_name | |
| CREATE INDEX idx_name ON tbl_name | |
| DROP INDEX idx_name ON tbl_name | |
| DROP INDEX idx_name | |

| Experimental supported SQL | Necessary conditions |
|---|---|
| SELECT * FROM (SELECT * FROM tbl_name) o | |
| SELECT * FROM (SELECT * FROM tbl_name) o WHERE o.col1 = ? | |
| SELECT * FROM (SELECT * FROM tbl_name WHERE col1 = ?) o | |
| SELECT * FROM (SELECT * FROM tbl_name WHERE col1 = ?) o WHERE o.col1 = ? | Subquery and outer query in different sharded data node after route |
| SELECT (SELECT MAX(col1) FROM tbl_name) a, col2 from tbl_name | |
| SELECT SUM(DISTINCT col1), SUM(col1) FROM tbl_name | |
| SELECT col1, SUM(col2) FROM tbl_name GROUP BY col1 HAVING SUM(col2) > ? | |
| SELECT col1, col2 FROM tbl_name UNION SELECT col1, col2 FROM tbl_name | |
| SELECT col1, col2 FROM tbl_name UNION ALL SELECT col1, col2 FROM tbl_name | |

| Slow SQL | Reason |
|---|---|
| SELECT * FROM tbl_name WHERE to_date(create_time, ‘yyyy-mm-dd’ ) = ? | Full route because of sharding value in calculate expression |

| Unsupported SQL | Reason | So lution |
|---|---|---|
| INSERT INTO tbl_name (col1, col2, …) SELECT * FROM tbl_name WHERE col3 = ? | SELECT clause does not support *-shorthand and built-in key generator | - |
| REPLACE INTO tbl_name (col1, col2, …) SELECT * FROM tbl_name WHERE col3 = ? | SELECT clause does not support *-shorthand and built-in key generator | - |
| SELECT MAX(tbl_name.col1) FROM tbl_name | Use table name as column owner in function | I nstead of table alias |

### Pagination

Totally support pagination queries of MySQL, PostgreSQL and Oracle; partially support SQLServer pagination query due to its complexity.

### Pagination Performance

### Performance Bottleneck

Pagination with query offset too high can lead to a low data accessibility, take MySQL as an example:

```
SELECT * FROM t_order ORDER BY id LIMIT 1000000, 10
```

This SQL will make MySQL acquire another 10 records after skipping 1,000,000 records when it is not able to use indexes. Its performance can thus be deduced. In sharding databases and sharding tables (suppose there are two databases), to ensure the data correctness, the SQL will be rewritten as this:

```
SELECT * FROM t_order ORDER BY id LIMIT 0, 1000010
```

It also means taking out all the records prior to the offset and only acquire the last 10 records after ordering. It will further aggravate the performance bottleneck effect when the database is already slow in execution. The reason for that is the former SQL only needs to transmit 10 records to the user end, but now it will transmit 1,000,010 * 2 records after the rewrite.

### Optimization of ShardingSphere

ShardingSphere was optimized in two ways.

Firstly, it adopts stream process + merger ordering to avoid excessive memory occupation. SQL rewrite unavoidably occupies extra bandwidth, but it will not lead to sharp increase of memory occupation. Most people may assume that ShardingSphere would upload all the 1,000,010 * 2 records to the memory and occupy a large amount of it, which can lead to memory overflow. But each ShardingSphere comparison only acquires current result set record of each shard, since result set records have their own order. The record stored in the memory is only the current position pointed by the cursor in the result set of the shard routed to. For the item to be sorted which has its own order, merger ordering only has the time complexity of O(mn(log m)), and the number of shard m is generally small enough to be considered as O(n), with a very low performance consumption.

Secondly, ShardingSphere further optimizes the query that only falls into single shards. Requests of this kind can guarantee the correctness of records without rewriting SQLs. Under this kind of situation, ShardingSphere will not do that in order to save the bandwidth.

### Pagination Solution Optimization

For LIMIT cannot search for data through indexes, if the ID continuity can be guaranteed, pagination by ID is a better solution:

```sql
SELECT * FROM t_order WHERE id > 100000 AND id <= 100010 ORDER BY id
```

Or use the ID of last record of the former query result to query the next page:

```sql
SELECT * FROM t_order WHERE id > 100000 LIMIT 10
```

### Pagination Sub-query

Both Oracle and SQLServer pagination need to be processed by sub-query, ShardingSphere supports pagination related sub-query.

- Oracle

Support rownum pagination:

```sql
SELECT * FROM (SELECT row_.*, rownum rownum_ FROM (SELECT o.order_id as order_id FROM t_order o JOIN t_order_item i
ON o.order_id = i.order_id) row_ WHERE rownum <= ?) WHERE rownum > ?
```

Do not support rownum + BETWEEN pagination for now.

- SQLServer

Support TOP + ROW_NUMBER() OVER pagination:

```sql
SELECT * FROM (SELECT TOP (?) ROW_NUMBER() OVER (ORDER BY o.order_id DESC) AS rownum, * FROM t_order o) AS temp
WHERE temp.rownum > ? ORDER BY temp.order_id
```

Support OFFSET FETCH pagination after SQLServer 2012:

```sql
SELECT * FROM t_order o ORDER BY id OFFSET ? ROW FETCH NEXT ? ROWS ONLY
```

Do not support WITH xxx AS (SELECT …) pagination. Because SQLServer automatically generated by Hibernate uses WITH statements, Hibernate SQLServer pagination or two TOP + sub-query pagination is not available now.

- MySQL, PostgreSQL

Both MySQL and PostgreSQL support LIMIT pagination, no need for sub-query:

```
SELECT * FROM t_order o ORDER BY id LIMIT ? OFFSET ?
```

## 2.4 Distributed Transaction

### 2.4.1 Background

Database transactions should satisfy ACID (atomicity, consistency, isolation and durability) features.

- Atomicity guarantees that each transaction is treated as a single unit, which either succeeds completely, or fails completely.
- Consistency ensures that a transaction can only bring the database from one valid state to another, maintaining database invariants.
- Isolation ensures that concurrent execution of transactions leaves the database in the same state that would have been obtained if the transactions were executed sequentially.
- Durability guarantees that once a transaction has been committed, it will remain committed even in the case of a system failure (e.g., power outage or crash).

In single data node, transactions are only restricted to the access and control of single database resources, called local transactions. Almost all the mature relational databases have provided native support for local transactions. But in distributed application situations based on micro-services, more and more of them require to include multiple accesses to services and the corresponding database resources in the same transaction. As a result, distributed transactions appear.

Though the relational database has provided perfect native ACID support, it can become an obstacle to the system performance under distributed situations. How to make databases satisfy ACID features under distributed situations or find a corresponding substitute solution, is the priority work of distributed transactions.

#### Local Transaction

It means let each data node to manage their own transactions on the premise that any distributed transaction manager is not on. They do not have any coordination and communication ability, or know other data nodes have succeeded or not. Though without any consumption in performance, local transactions are not capable enough in high consistency and eventual consistency.

#### 2PC Transaction

The earliest distributed transaction model of XA standard is X/Open Distributed Transaction Processing (DTP) model brought up by X/Open, XA for short.

Distributed transaction based on XA standard has little intrusion to businesses. Its biggest advantage is the transparency to users, who can use distributed transactions based on XA standard just as local transactions. XA standard can strictly guarantee ACID features of transactions.

That guarantee can be a double-edged sword. It is more proper in the implementation of short transactions with fixed time, because it will lock all the resources needed during the implementation process. For long transactions, data monopolization during its implementation will lead to an obvious concurrency performance recession for business systems depend on hot spot data. Therefore, in high concurrency situations that take performance as the highest, distributed transaction based on XA standard is not the best choice.

### BASE Transaction

If we call transactions that satisfy ACID features as hard transactions, then transactions based on BASE features are called soft transactions. BASE is the abbreviation of basically available, soft state and eventually consistent those there factors.

- Basically available feature means not all the participants of distributed transactions have to be online at the same time.
- Soft state feature permits some time delay in system renewal, which may not be noticed by users.
- Eventually consistent feature of systems is usually guaranteed by message availability.

There is a high requirement for isolation in ACID transactions: all the resources must be locked during the transaction implementation process. The concept of BASE transactions is uplifting mutex operation from resource level to business level through business logic. Broaden the requirement for high consistency to exchange the rise in system throughput.

Highly consistent transactions based on ACID and eventually consistent transactions based on BASE are not silver bullets, and they can only take the most effect in the most appropriate situations. The detailed distinctions between them are illustrated in the following table to help developers to choose technically:

| | Local transaction | 2PC (3PC) transaction | BASE transaction |
|---|---|---|---|
| Business trans formation | None | None | Relevant interface |
| Co nsistency | Not support | Support | Eventual consistency |
| Isolation | Not support | Support | Business-side guarantee |
| Co ncurrency pe rformance | No influence | Serious recession | Minor recession |
| Situation | Inconsistent operation at business side | Short transaction & low concurrency | Long transaction & high concurrency |

## 2.4.2 Challenge

For different application situations, developers need to reasonably weight the performance and the function between all kinds of distributed transactions.

Highly consistent transactions do not have totally the same API and functions as soft transactions, and they cannot switch between each other freely and invisibly. The choice between highly consistent transactions and soft transactions as early as development decision-making phase has sharply increased the design and development cost.

Highly consistent transactions based on XA is relatively easy to use, but is not good at dealing with long transaction and high concurrency situation of the Internet. With a high access cost, soft transactions require developers to transform the application and realize resources lock and backward compensation.

## 2.4.3 Goal

**The main design goal of Apache ShardingSphere's distributed transaction module is to integrate existing mature transaction cases to provide an unified distributed transaction interface for local transactions, 2PC transactions and soft transactions; and compensate for the deficiencies of current solutions to provide a one-stop distributed transaction solution.**

### 2.4.4 Core Concept

#### Navigation

This chapter mainly introduces the core concepts of distributed transactions, including:

- XA transaction
- BASE transaction

#### XA

2PC transaction submit uses the DTP Model defined by X/OPEN, in which created AP (Application Program), TM (Transaction Manager) and RM (Resource Manager) can guarantee a high transaction consistency. TM and RM use XA protocol for bidirectional streaming. Compared with traditional local transactions, XA transactions have a prepared phase, where the database cannot only passively receive commands, but also notify the submitter whether the transaction can be accepted. TM can collect all the prepared results of branch transactions before submitting all of them together, which has guaranteed the distributed consistency.

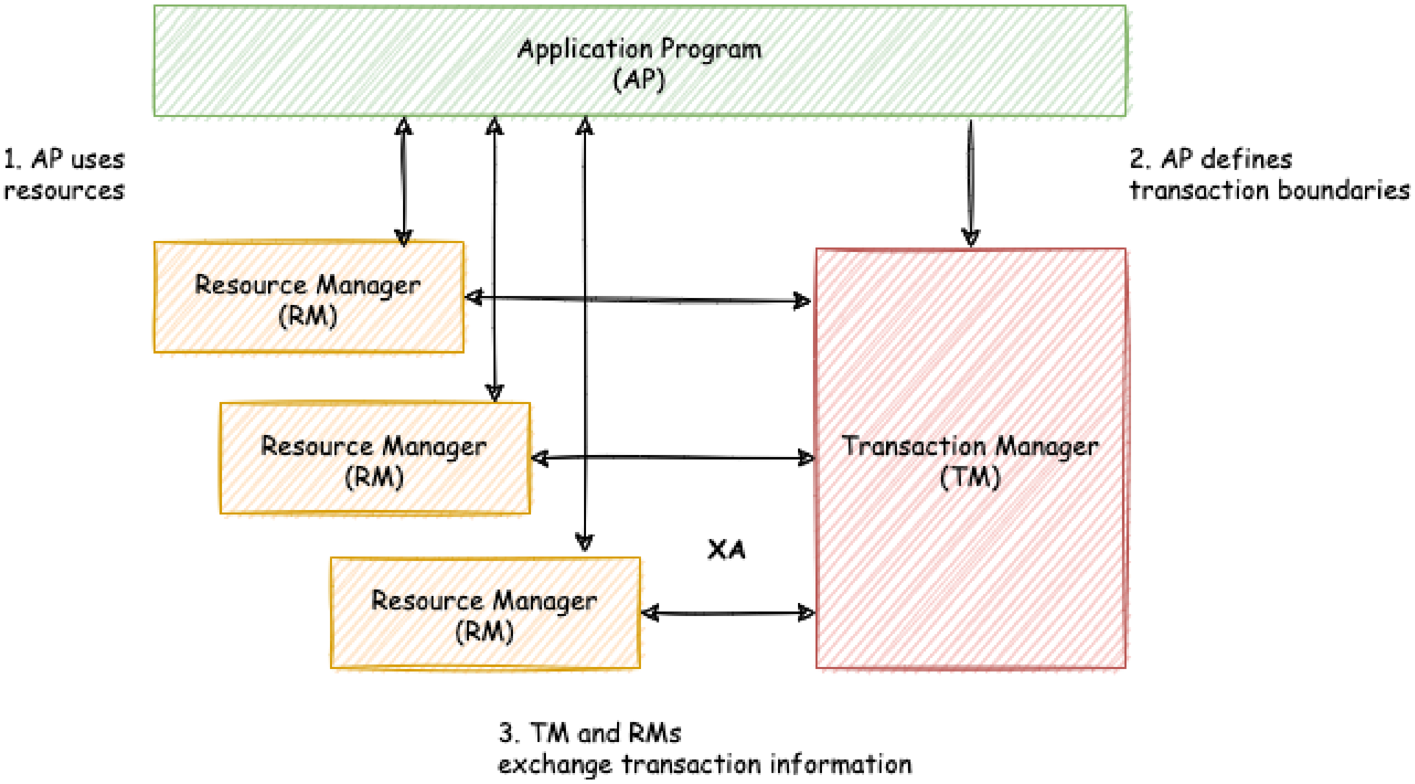


Fig. 4: 2PC XA model

Java implements the XA model through defining a JTA interface, in which ResourceManager requires an XA driver provided by database manufacturers and TransactionManager is provided by transaction manager manufacturers. Traditional transaction managers need to be bound with application server, which creates a high use cost. Built-in transaction managers have already been able to provide services through jar packages. Integrated with Apache ShardingSphere, it can guarantee the high consistency in cross-database transactions after sharding.

Usually, to use XA transaction, users must use its connection pool provided by transaction manager manufacturers. However, when Apache ShardingSphere integrates XA transactions, it has separated the management of XA transaction and its connection pool, so XA will not invade the applications.

### BASE

A paper published in 2008 first mentioned that BASE transaction advocates the use of eventual consistency instead of consistency to improve concurrency of transaction processing.

TCC and Saga are two regular implementations. They use reverse operation implemented by developers themselves to ensure the eventual consistency when data rollback. SEATA implements SQL reverse operation automatically, so that BASE transaction can be used without the intervention of developers.

Apache ShardingSphere integrates SEATA as solution of BASE transaction.

## 2.4.5 Use Norms

### Background

Although SphereEx-DBPlusEngine intends to be compatible with all distributed scenario and best performance, under CAP theorem guidance, there is no sliver bullet with distributed transaction solutions.

Apache ShardingSphere wants to give the users the choice of distributed transaction type and use the most suitable solution in different scenarios.

### Local Transaction

### Supported

- Support none-cross-database transactions. For example, sharding table or sharding database with its route result in same database.
- Support cross-database transactions caused by logic exceptions. For example, update two databases in transaction with exception thrown, data can rollback in both databases.

### Unsupported

- Do not support the cross-database transactions caused by network or hardware crash. For example, when update two databases in transaction, if one database crashes before commit, then only the data of the other database can commit.

### XA

### Supported

- Support cross-database transactions after sharding.
- Operation atomicity and high data consistency in 2PC transactions.
- When service is down and restarted, commit and rollback transactions can be recovered automatically.
- Support use XA and non-XA connection pool together.

### Unsupported

- Recover committing and rolling back in other machines after the service is down.

### XA Transaction managed by XA Statement

- When using XA START to open a XA Transaction, DBPlusEngine will pass it to backend database directly, and you have to manage this transaction by yourself.
- When recovering from a crush, you have to call XA RECOVER to check unfinished transaction, and choose to commit or rollback using xid. Or you can use ONE PHASE commit without PREPARE.

```
MySQL [(none)]> use test1                                          | MySQL [(none)]> use test2
Reading table information for completion of table and column names                       | Reading table information for
completion of table and column names
You can turn off this feature to get a quicker startup with -A                   | You can turn off this feature to get a quicker
startup with -A
                                                   |
Database changed                                            | Database changed
MySQL [test1]> XA START '61c052438d3eb';                                   | MySQL [test2]> XA START '61c0524390927';
Query OK, 0 rows affected (0.030 sec)                                 | Query OK, 0 rows affected (0.009 sec)
                                                   |
MySQL [test1]> update test set val = 'xatest1' where id = 1;                       | MySQL [test2]> update test set val = 'xatest2'
where id = 1;
Query OK, 1 row affected (0.077 sec)                                  | Query OK, 1 row affected (0.010 sec)
                                                   |
MySQL [test1]> XA END '61c052438d3eb';                                    | MySQL [test2]> XA END '61c0524390927';
Query OK, 0 rows affected (0.006 sec)                                 | Query OK, 0 rows affected (0.008 sec)
                                                   |
MySQL [test1]> XA PREPARE '61c052438d3eb';                                  | MySQL [test2]> XA PREPARE '61c0524390927';
Query OK, 0 rows affected (0.018 sec)                                 | Query OK, 0 rows affected (0.011 sec)
                                                   |
MySQL [test1]> XA COMMIT '61c052438d3eb';                                  | MySQL [test2]> XA COMMIT '61c0524390927';
Query OK, 0 rows affected (0.011 sec)                                 | Query OK, 0 rows affected (0.018 sec)
                                                   |
MySQL [test1]> select * from test where id = 1;                             | MySQL [test2]> select * from test where id = 1;
+----+---------+                                         | +----+---------+
| id | val    |                                         | | id | val    |
+----+---------+                                         | +----+---------+
|  1 | xatest1 |                                        | |  1 | xatest2 |
+----+---------+                                         | +----+---------+
1 row in set (0.016 sec)                                      | 1 row in set (0.129 sec)

MySQL [test1]> XA START '61c05243994c3';                                   | MySQL [test2]> XA START '61c052439bd7b';
Query OK, 0 rows affected (0.047 sec)                                 | Query OK, 0 rows affected (0.006 sec)
                                                   |
MySQL [test1]> update test set val = 'xarollback' where id = 1;                     | MySQL [test2]> update test set val = 'xarollback
' where id = 1;
Query OK, 1 row affected (0.175 sec)                                  | Query OK, 1 row affected (0.008 sec)
                                                   |
MySQL [test1]> XA END '61c05243994c3';                                    | MySQL [test2]> XA END '61c052439bd7b';
Query OK, 0 rows affected (0.007 sec)                                 | Query OK, 0 rows affected (0.014 sec)
                                                   |
MySQL [test1]> XA PREPARE '61c05243994c3';                                  | MySQL [test2]> XA PREPARE '61c052439bd7b';
Query OK, 0 rows affected (0.013 sec)                                 | Query OK, 0 rows affected (0.019 sec)
                                                   |
MySQL [test1]> XA ROLLBACK '61c05243994c3';                                 | MySQL [test2]> XA ROLLBACK '61c052439bd7b';
Query OK, 0 rows affected (0.010 sec)                                 | Query OK, 0 rows affected (0.010 sec)
                                                   |
MySQL [test1]> select * from test where id = 1;                             | MySQL [test2]> select * from test where id = 1;
+----+---------+                                         | +----+---------+
| id | val     |                                          | | id | val     |
+----+---------+                                            | +----+---------+
|  1 | xatest1 |                                          | |  1 | xatest2 |
+----+---------+                                            | +----+---------+
1 row in set (0.009 sec)                                        | 1 row in set (0.083 sec)

MySQL [test1]>  XA START '61c052438d3eb';
Query OK, 0 rows affected (0.030 sec)

MySQL [test1]> update test set val = 'recover' where id = 1;
Query OK, 1 row affected (0.072 sec)

MySQL [test1]> select * from test where id = 1;
+----+---------+
| id | val     |
+----+---------+
|  1 | recover |
+----+---------+
1 row in set (0.039 sec)

MySQL [test1]>  XA END '61c052438d3eb';
Query OK, 0 rows affected (0.005 sec)

MySQL [test1]> XA PREPARE '61c052438d3eb';
Query OK, 0 rows affected (0.020 sec)

MySQL [test1]> XA RECOVER;
+----------+--------------+--------------+---------------+
| formatID | gtrid_length | bqual_length | data          |
+----------+--------------+--------------+---------------+
|        1 |           13 |            0 | 61c052438d3eb |
+----------+--------------+--------------+---------------+
1 row in set (0.010 sec)

MySQL [test1]> XA RECOVER CONVERT XID;
+----------+--------------+--------------+------------------------------+
| formatID | gtrid_length | bqual_length | data                         |
+----------+--------------+--------------+------------------------------+
|        1 |           13 |            0 | 0x36316330353234333864336562 |
+----------+--------------+--------------+------------------------------+
1 row in set (0.011 sec)

MySQL [test1]> XA COMMIT 0x36316330353234333864336562;
Query OK, 0 rows affected (0.029 sec)

MySQL [test1]> XA RECOVER;
Empty set (0.011 sec)
```

## BASE

### Supported

- Support cross-database transactions after sharding.
- Support RC isolation level.
- Rollback transaction according to undo log.
- Support recovery committing transaction automatically after the service is down.

#### Unsupported

- Does not support other isolation level except RC.

#### To Be Optimized

- SQL parsed twice by SphereEx-DBPlusEngine and SEATA.

## 2.5 Read/write splitting

### 2.5.1 Background

Database throughput is facing bottlenecks with increasing TPS. For applications with massive concurrence read but less write at the same time, we can divide the database into a primary database and a replica database. The primary database is responsible for the insert, delete and update of transactions, while the replica database is responsible for queries. It can significantly improve the query performance of the whole system by effectively avoiding row locks.

One primary database with multiple replica databases can further enhance processing capacity by distributing queries evenly into multiple data replicas. Multiple primary databases with multiple replica databases can enhance not only throughput but also availability. Therefore, the system can still run normally, even though any database is down or physical disk destroyed.

Different from the sharding that separates data to all nodes according to sharding keys, read/write splitting routes read and write separately to primary database and replica databases according SQL analysis.

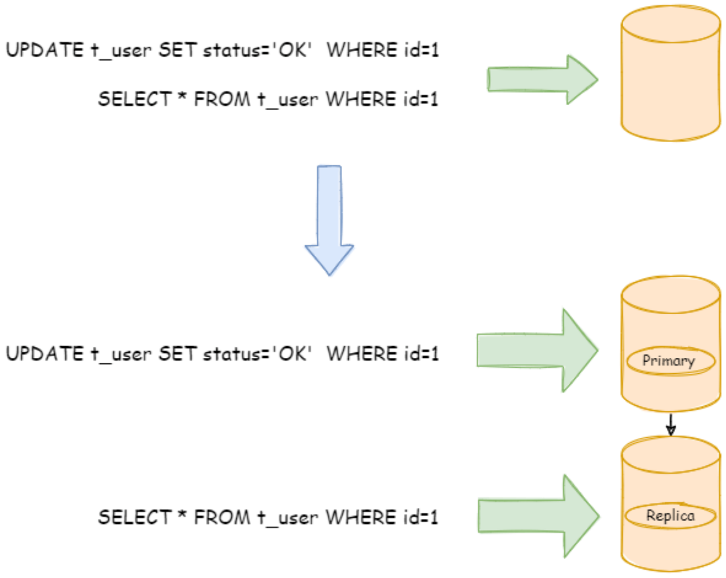

UPDATE t_user SET status='OK' WHERE id=1
SELECT * FROM t_user WHERE id=1
UPDATE t_user SET status='OK' WHERE id=1
Primary
SELECT * FROM t_user WHERE id=1
Replica

Fig. 5: Background

Data in read/write splitting nodes are consistent, whereas that in shards is not. The combined use of sharding and read/write splitting will effectively enhance the system performance.

### 2.5.2 Challenges

Though read/write splitting can enhance system throughput and availability, it also brings inconsistent data, including that among multiple primary databases and among primary databases and replica databases. What' s more, it also brings the same problem as data sharding, complicating developer and operator' s maintenance and operation. The following diagram shows the complex topological relationship between applications and database groups when sharding used together with read/write splitting.

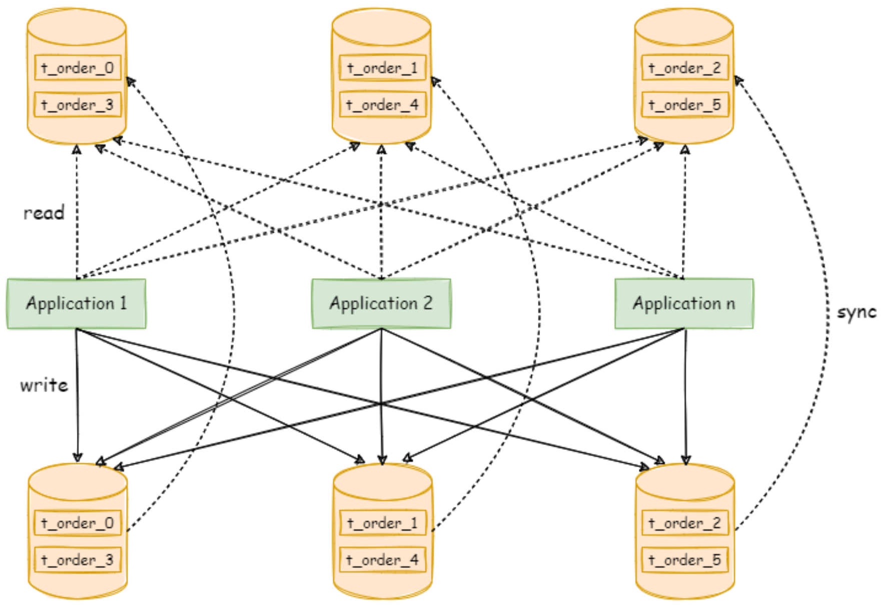


Fig. 6: Challenges

### 2.5.3 Goal

The main design goal of read/write splitting of DBPlusEngine is to try to reduce the influence of read/write splitting, in order to let users use primary-replica database group like one database.

### 2.5.4 Core Concept

**Primary Database**

It refers to the database used in data insertion, update and deletion. It only supports single primary database for now.

**Replica Database**

It refers to the database used in data query. It supports multiple replica databases.

**Primary Replica Replication**

It refers to the operation to asynchronously replicate data from the primary database to the replica database. Because of the asynchrony of primary-replica synchronization, there may be short-time data inconsistency between them.

**Load Balance Strategy**

Through this strategy, queries separated to different replica databases.

### 2.5.5 Use Norms

**Supported**

- Provide the read/write splitting configuration of one primary database with multiple replica databases, which can be used alone or with sharding table and database;
- Primary nodes need to be used for both reading and writing in the transaction;
- Forcible primary database route based on SQL Hint;
- The primary database provides read traffic when all secondary databases are down.

**Unsupported**

- Data replication between the primary and the replica databases;
- Data inconsistency caused by replication delay between databases;
- Double or multiple primary databases to provide write operation;
- The data for transaction across primary and replica nodes are inconsistent; In the read/write splitting model, primary nodes need to be used for both reading and writing in the transaction.

## 2.6 HA

### 2.6.1 Background

High availability is the most basic requirement of modern systems. As the cornerstone of the system, the database is also essential for high availability.

In the distributed database system with storage-compute splitting, the high availability solution of storage node and compute node are different. The stateful storage nodes need to pay attention to data consistency, health detection, primary node election and so on. The stateless compute nodes need to detect the changes of storage nodes, they also need to set up an independent load balancer and have the ability of service discovery and request distribution.

DBPlusEngine provides compute nodes and reuse database as storage nodes. Therefore, the high availability solution it adopts is to use the high availability solution of the database itself as the high availability of the storage node, and detect the changes automatically.

## 2.6.2 Challenges

DBPlusEngine needs to detect high availability solution of diversified storage nodes automatically, and can also integrate the readwrite splitting dynamically, which is the main challenge of implementation.

## 2.6.3 Goal

The main goal of DBPlusEngine' s high availability module is to ensure 24/7 uninterrupted database service as much as possible.

## 2.6.4 Core Concept

### High Availability Type

DBPlusEngine does not provide a high availability database solution, it reuses 3rd party high availability solution and auto-detect switch of primary and replica databases.

Specifically, the ability of DBPlusEngine provided is database discovery, detect the primary and replica databases automatically, and updates the connection of compute nodes to the databases.

### Dynamic Readwrite-Splitting

When high availability and read/write0-splitting are used together, there is unnecessary to configure specific primary and replica databases for readwrite-splitting.

When high availability and read/write splitting are used together, it supports detecting the delay time of the secondary database during semi-synchronous replication and asynchronous replication, and dynamically routes the secondary database with low delay to provide read traffic.

Highly available data sources will update the primary and replica databases of read/write splitting dynamically, and route the query and update SQL correctly.

High availability also provides that all secondary databases are down, and the read traffic is automatically routed to the main databased, to ensure the availability of the business system.

## 2.6.5 Use Norms

### Supported

- MySQL MGR single-primary mode.
- MySQL primary-secondary replication mode.
- openGauss primary-secondary replication mode.

#### Unsupported

- MySQL MGR multi-primary mode.

## 2.7 Data Migration

### 2.7.1 Background

In a scenario where the business continues to develop and the amount of data and concurrency reaches a certain extent, the traditional single database may face problems in terms of performance, scalability and availability.

Although NoSQL solutions can solve the above problems through data sharding and horizontal scale-out, NoSQL databases generally do not support transactions and SQL.

DBPlusEngine can also solve the above problems and supports data sharding and horizontal scale-out, while at the same time, also supporting distributed transactions and SQL.

The data migration scheme provided by DBPlusEngine can help the traditional single database smoothly switch to DBPlusEngine.

### 2.7.2 Challenges

The data migration process should not affect the running services. So the first challenge is to minimize the time window during which data is not available.

Next, data migration should not affect existing data. So the second challenge is to ensure the data correctness.

### 2.7.3 Goal

The major goal of SphereEx-DBPlusEngine in performing data migration is to reduce the impact of data migration on services and provide a one-stop universal data migration solution.

### 2.7.4 Core Concept

#### Nodes

Instances for running compute or storage tier component processes. These can either be physical machines, virtual machines, or containers, etc.

#### Cluster

Multiple nodes that are assembled together to provide a specified service.

#### Source

The storage cluster where the original data resides.

#### Target

The target storage cluster to which the original data is to be migrated.

#### Data Migration Process

The entire process of replicating data from one storage cluster to another.

#### Stock Data

The data that was already in the data node before the data migration operation started.

#### Incremental Data

New data generated by operational systems during the execution of data migration operations.

### 2.7.5 Limitations

#### Supported Procedures

- Migration of peripheral data to databases managed by SphereEx-DBPlusEngine.
- Migration of integer or string primary key tables.

#### Procedures not supported

- Migrate tables without primary keys or unique keys.
- Migrate tables with composite primary keys or composite unique keys.
- Migration on top of the current storage node is not supported, so a brand new database cluster needs to be prepared as the migration target cluster.

## 2.8 Scaling

### 2.8.1 Definition

Scaling refers to the system that dynamically expands or shrinks the capacity according to the status of the storage layer nodes, 、to ensure that the resource consumption is reduced as much as possible while meeting the business needs of the upper layer.

### 2.8.2 Related Concepts

#### Node

An instance running a compute or storage layer component process, which can be a physical machine, a virtual machine, a container, and so on.

#### Cluster

Multiple nodes that are grouped together to provide a specific service.

### 2.8.3 Limitations

#### Supported Items

- Integer or string primary key table migration.

#### Unsupported Items

- No primary key table migration.
- Composite primary key table migration.

### 2.8.4 How it works

If the sharding algorithm and the use method meet the conditions, it can be efficiently scaled and contracted, without the need to move data or only a small amount of data.

## Range sharding/scaling-out

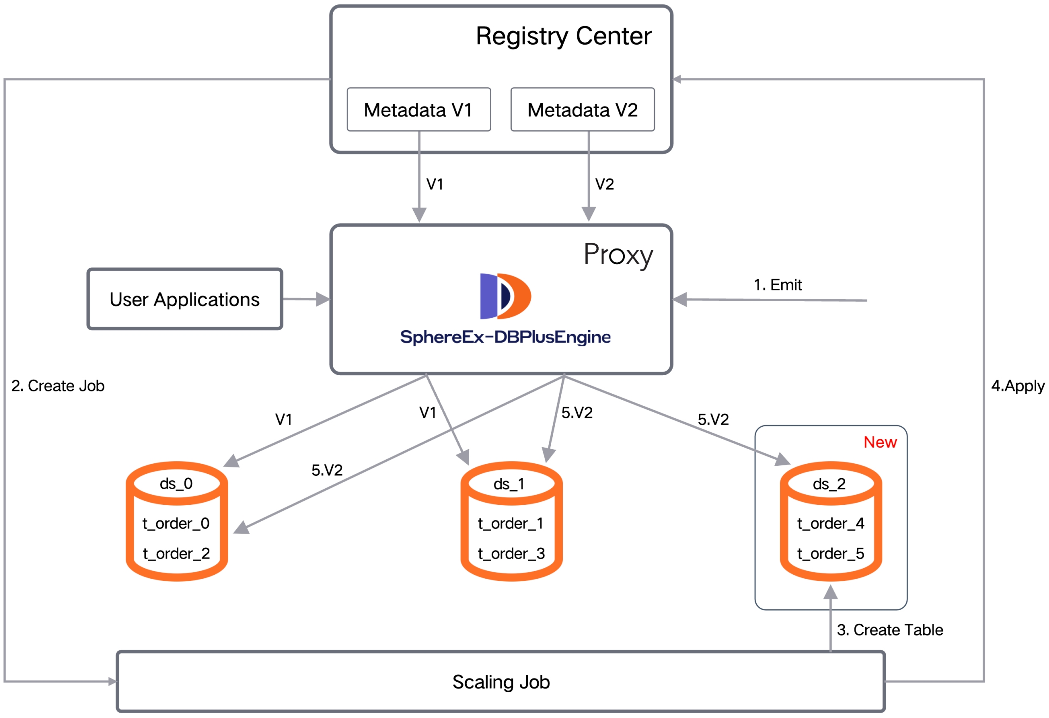


Fig. 7: Detailed process

### Range sharding/scaling-in

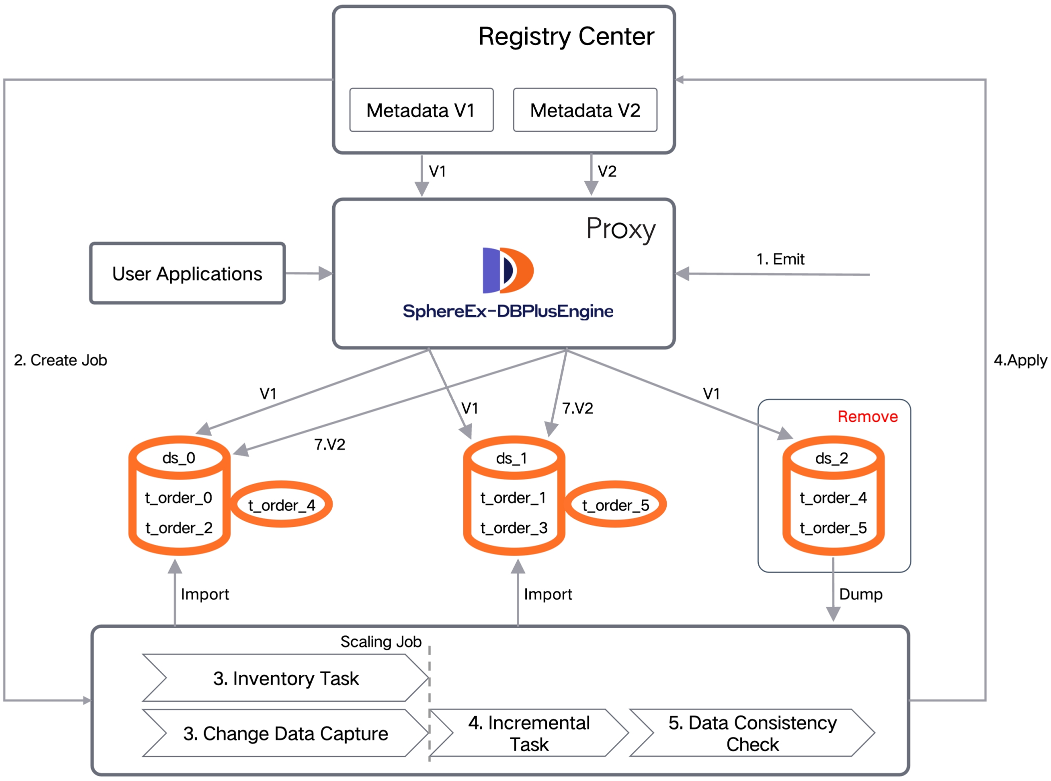


Fig. 8: Detailed process

## 2.8.5 Related References

Scaling

# 2.9 Encryption

## 2.9.1 Background

When the encryption and decryption configuration are used by the current user, only the names of plaintext column, ciphertext column and auxiliary query column can be configured, while the field definitions of these columns cannot be configured.

Therefore, when the user executes relevant DDL statements, the encryption column can only be created according to the definition of logical column. The type and length of the rewritten plaintext column and ciphertext column are copied according to the logical column. There are two problems with this rewriting method:

- Encryption algorithms with different field types before and after encryption cannot be supported (for example, a numeric type becomes a string type after encryption).
- For the field length, users cannot customize it. The length of ciphertext is usually greater than that of plaintext.

Here below, you can see the rewriting of encrypted columns when creating a table. We see definition of varchar(100) DEFAULT NULL and definition of user_cipher in user_id is exactly the same.

```
mysql> PREVIEW CREATE TABLE `t_encrypt` (`id` int(11) DEFAULT NULL, `user_id` varchar(100) DEFAULT NULL, `order_id`
varchar(100) DEFAULT NULL) ENGINE=InnoDB;
+-----------------+-------------------------------------------------------------------------------------------------------------------------------
----------------------------------+
| data_source_name | sql                                                                                                   |
+------------------+-------------------------------------------------------------------------------------------------------
----------------------------------+
| ds_0             | CREATE TABLE `t_encrypt` (`id` int(11) DEFAULT NULL, user_cipher varchar(100) DEFAULT NULL, user_plain
varchar(100) DEFAULT NULL, order_cipher varchar(100) DEFAULT NULL) ENGINE=InnoDB |
+------------------+-------------------------------------------------------------------------------------------------------
----------------------------------+
```

In order to improve the usability of the feature, it is necessary to add field type configuration for the encrypted column, allowing flexible configuration to the users.

Data encryption provides cloud key management and encryption function, as follows:

- Cloud key management

The encrypted key plaintext configuration adopted by encrypt has potential security risks in the props file. The cloud key management function can be added to manage encrypted keys through AWS.

- Encryption

Currently, DBPlusEngine provides an encryption solution. For new tables and new businesses, you can directly use the encryption rules to configure, but for existing data tables, you need to encrypt the plaintext fields in these tables to convert them into encrypted content. At the same time, if the user needs to change the key, we also provide the corresponding decryption function, and the two cooperate to complete the key replacement.

### 2.9.2 Challenges

In real business scenarios, relevant development teams often need to implement and maintain a set of encryption and decryption systems according to the needs of the company’ s security team.

When the encryption scenario changes, the self maintained encryption system often faces the risk of rebuild or modification.

Additionally, when it comes to existing businesses, it is relatively complex to achieve seamless encryption transformation in a transparent, safe and low-risk manner without modifying business logic and SQL.

### 2.9.3 Goal

By adding the configuration of encryption column field type definition, users can rewrite according to the configured plaintext column and ciphertext column when executing DDL statements, making the encryption and decryption feature easier to use.

### 2.9.4 Core Concept

#### Logic Column

Column name used to encrypt, it is the logical column identification in SQL.

It includes cipher column (required), query assistant column (optional) and plain column (optional).

### Logical column type (datatype)

It is used to define the types of logical columns, such as INT NOT NULL，VARCHAR(200) DEFAULT NULL etc. For details, see the definitions of various dialect fields in the official documents, such as the definition of column_definition in MySQL create statement (https://dev.mysql.com/doc/refman/8.0/en/create-table.html).

### Cipher Column (cipherColumn)

Encrypted data column.

### Ciphertext Column Type (cipherdatatype)

Used to define the type of ciphertext column, which is the same as that of logical column.

### Query Assistant Column (assistedQueryColumn)

Column used to assistant for query.

For non-idempotent encryption algorithms with higher security level, irreversible idempotent columns provided for query.

### Query Assistant Column Type (ssistedQueryDataType)

Used to define query assistant column types, the same as logical column types.

### Plain Column (plainColumn)

Column used to persist plain column, for service provided during data encryption.

Should be removed after data cleaning.

### Plain Column Type (plainDataType)

Used to define plain column types, the same as logical column types.

### Encrypting (encrypting)

Encrypt the unencrypted data in the database in batches.

### Decrypting (decrypting)

Decrypt the encrypted data in the database in batches.

## 2.9.5 Implementation

### Cloud key management

The key is managed in the cloud, for example, the secretkey function of AWS is used to save the key, to improve the security and convenience of the entire encryption.

When initializing the encryption algorithm, the program establishes a connection with AWS to obtain the relevant key stored in AWS, and then stores the key in the algorithm. The whole data encryption process does not involve network interaction with the cloud.

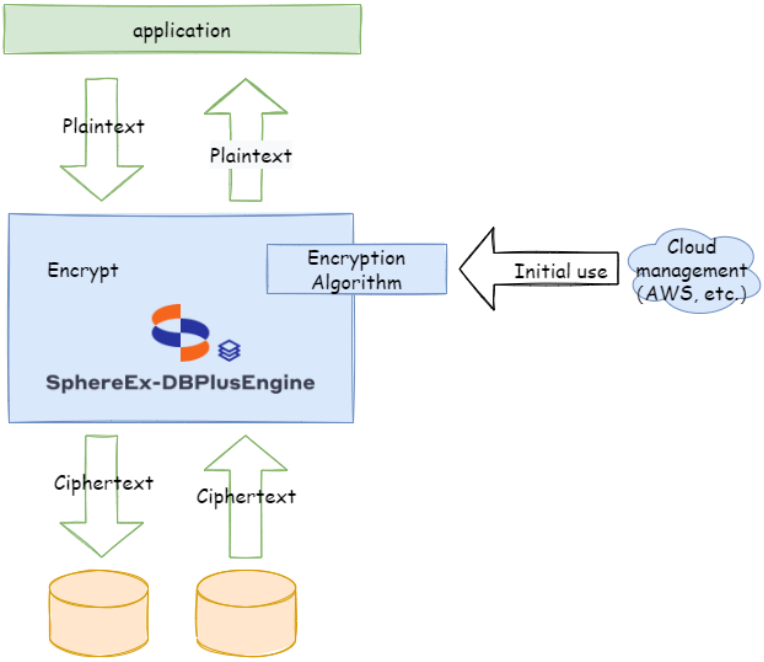


Fig. 9: implementation

### Encrypting

The encryption task is triggered by DistSQL. After receiving the request of the encryption task, the program will create the encryption task according to the current encryption rules. The encrypting task is mainly composed of two parts, one is the query task, the other is the update task. The query task is responsible for querying the user’ s table data and obtaining the plaintext fields to be encrypted, and then pushing them to the channel. The update task obtains data from the channel and encrypts the update. The whole task creation and execution process will interact with the governance center, so users can query the task progress and clean up tasks through relevant DistSQL.

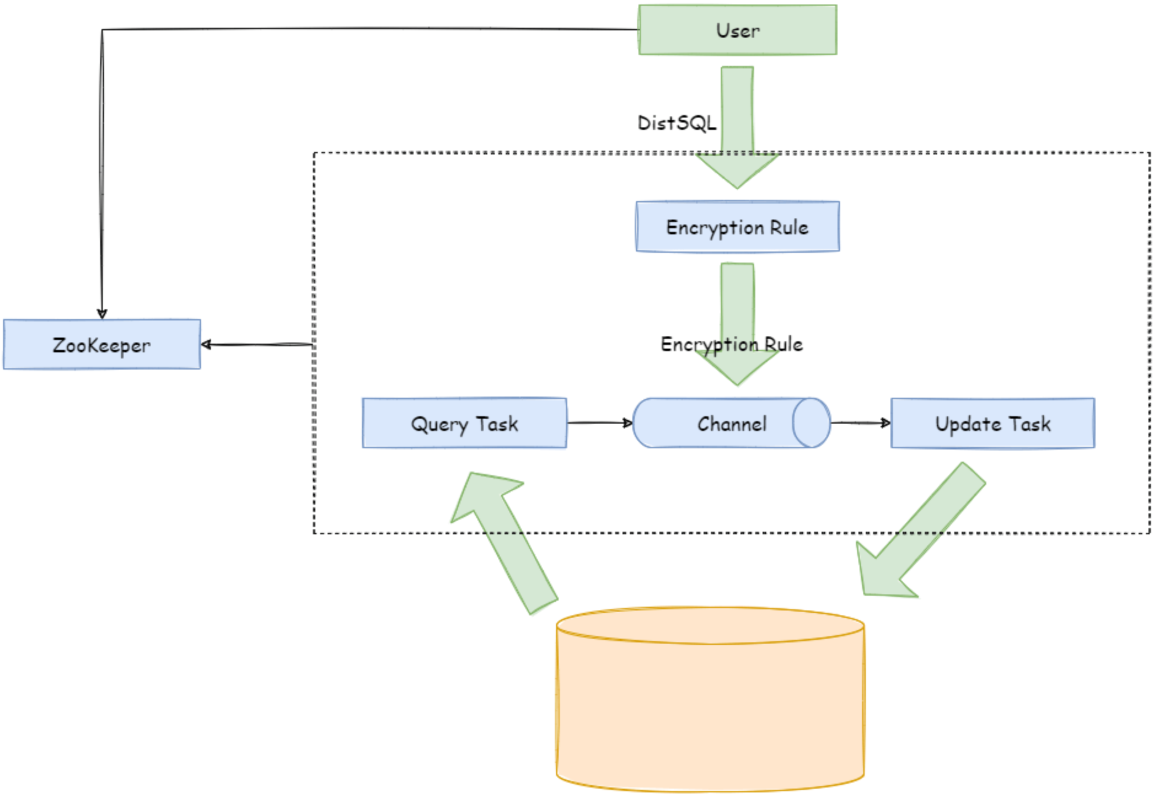


Fig. 10: implementation

### 2.9.6 Usage Norms

#### Supported

- Encrypt/decrypt one or more columns in the database table.
- Compatible with all regular SQL.

#### Unsupported

- It is necessary to process the original stock data in the database by itself.
- The value of encryption columns cannot support comparison, such as: >, <, ORDER BY, BETWEEN, LIKE, etc.
- The value of encryption columns cannot support calculation, such as AVG, SUM, and calculation expressions.

## 2.10 Shadow DB

### 2.10.1 Background

Under the distributed application architecture based on microservices, businesses require multiple services to be completed through a series of services and middleware calls. The pressure testing of a single service can no longer reflect the real scenario.

In the test environment, the cost of rebuild complete set of pressure test environment similar to the production environment is too high. It is usually impossible to simulate the complexity and data of the production environment.

So, it is the better way to use the production environment for pressure test. The test results obtained real capacity and performance of the system accurately.

### 2.10.2 Challenges

Pressure testing on production environment is a complex and huge task. Coordination and adjustments between microservices and middlewares required to cope with the transparent transmission of different flow rates and pressure test tags. Usually we will build a complete set of pressure testing platform for different test plans.

Data isolation have to be done at the database-level, in order to ensure the reliability and integrity of the production data, data generated by pressure testing routed to test database. Prevent test data from polluting the real data in the production database.

This requires business applications to perform data classification based on the transparently transmitted pressure test identification before executing SQL, and route the corresponding SQL to the corresponding data source.

### 2.10.3 Goal

DBPlusEngine focuses on data solutions in pressure testing on production environment.

The main goal of the DBPlusEngine shadow Database module is routing pressure testing data to user defined database automatically.

### 2.10.4 Core Concept

#### Production Database

The database used for production data.

#### Shadow Database

The database for pressure testing data isolation.

### Shadow Algorithm

The shadow algorithms are closely related to business, 2 types of shadow algorithms provided:

- Column based shadow algorithm

Recognize data from SQL and route to shadow databases. Suitable for test data driven scenario.

- Hint based shadow algorithm

Recognize comment from SQL and route to shadow databases. Suitable for identify passed by upstream system scenario.

## 2.10.5 Usage Norms

### Supported

- Hint based shadow algorithm support all SQL.
- Column based shadow algorithm support part of SQL.

### Unsupported

### Hint based shadow algorithm

- None

### Column based shadow algorithm

- Does not support DDL.
- Does not support range, group and subquery, for example: BETWEEN, GROUP BY …HAVING…;

SQL support list:

- INSERT

| SQL | S upported |
|---|---|
| INSERT INTO table (column,…) VALUES (value,…) | Y |
| INSERT INTO table (column,…) VALUES (value,…),(value,…),… | Y |
| INSERT INTO table (column,…) SELECT column1 from table1 where column1 = value1 | N |

- SELECT/UPDATE/DELETE

| Condition | SQL | Supported |
|---|---|---|
| = | SELECT/UPDATE/DELETE …WHERE column = value | Y |
| LIKE/NOT LIKE | SELECT/UPDATE/DELETE …WHERE column LIKE/NOT LIKE value | Y |
| IN/NOT IN | SELECT/UPDATE/DELETE …WHERE column IN/NOT IN (value1,value2,…) | Y |
| BETWEEN | SELECT/UPDATE/DELETE …WHERE column BETWEEN value1 AND value2 | N |
| GROUP BY …HAVING… | SELECT/UPDATE/DELETE …WHERE …GROUP BY column HAVING column > value | N |
| Subquery | SELECT/UPDATE/DELETE …WHERE column = (SELECT column FROM table WHERE column = value) | N |

## 2.11 Observability

### 2.11.1 Background

In order to grasp the distributed system status, observe running state of the cluster is a new challenge. The point-to-point operation mode of logging into a specific server is not suitable for large number of distributed servers. Telemetry through observable data is the recommended operation and maintenance mode in such cases. Tracking, metrics and logging are important ways to obtain observable data of system status.

APM (application performance monitoring) monitors and diagnoses the performance of the system by collecting, storing and analyzing the observable data of the system. Its main functions include performance index monitoring, call stack analysis, service topology, etc.

DBPlusEngine is not responsible for gathering, storing and demonstrating APM data, but provides the necessary information for the APM. In other words, DBPlusEngine is only responsible for generating valuable data and submitting it to relevant systems through standard protocols or plug-ins. Tracing is to obtain the tracking information of SQL parsing and SQL execution. DBPlusEngine provides support for SkyWalking, Zipkin, Jaeger and OpenTelemetry by default. It also supports users to develop customized components through plug-in.

- Use Zipkin or Jaeger

Just provides correct Zipkin or Jaeger server information in the agent configuration file.

- Use OpenTelemetry

OpenTelemetry was merged by OpenTracing and OpenCencus in 2019. In this way, you only need to fill in the appropriate configuration in the agent configuration file according to OpenTelemetry SDK Autoconfigure Guide.

- Use SkyWalking

Enable the SkyWalking plugin configuration file and configure the SkyWalking apm-toolkit.

- Use SkyWalking’s automatic monitor probe

In cooperation with the Apache SkyWalking team, the DBPlusEngine team has created ShardingSphere automatic monitor probe to automatically send performance data to SkyWalking. Note that automatic probe cannot be used together with DBPlusEngine plugin probe.

Metrics used to collect and display statistical indicator of cluster. DBPlusEngine supports Prometheus by default.

### 2.11.2 Challenges

Tracing and metrics need to collect system information through event tracking. Lots of events tracking make kernel code messy, difficult to maintain, and difficult to customize extend.

### 2.11.3 Goal

The goal of the DBPlusEngine observability module is providing as many performance and statistical indicators as possible and isolating kernel code and embedded code.

## 2.11.4 Core Concept

### Agent

Based on bytecode enhance and plugin design to provide tracing, metrics and logging features. Enable the plugin in agent to collect data and send data to the integrated 3rd APM system.

### APM

APM is the abbreviation for application performance monitoring. It works for performance diagnosis of distributed systems, including chain demonstration, service topology analysis and so on.

### Tracing

Tracing data between distributed services or internal processes will be collected by agent. It then will be sent to APM system.

### Metrics

System statistical indicator which collected from agent. Write to time series databases periodically. 3rd party UI can display the metrics data simply.

## 2.11.5 Usage Norms

### Compile source code

Download DBPlusEngine from GitHub,Then compile.

```
git clone --depth 1 https://github.com/apache/shardingsphere.git
cd shardingsphere
mvn clean install -Dmaven.javadoc.skip=true -Dcheckstyle.skip=true -Drat.skip=true -Djacoco.skip=true -DskipITs -DskipTests -
Prelease
```

Output directory: shardingsphere-agent/shardingsphere-agent-distribution/target/apache-shardingsphere-${latest.release.version}-shardingsphere-agent-bin.tar.gz

### Agent configuration

- Directory structure

  Create agent directory, and unzip agent distribution package to the directory. ``shell mkdir agent tar -zxvf apache-shardingsphere-:math:`{latest.release.version}-shardingsphere-agent-bin.tar.gz -C agent cd agent tree . ├── conf │ ├── agent.yaml │ └── logback.xml ├── plugins │ ├── shardingsphere-agent-logging-base-{latest.release.version}.jar │ ├── shardingsphere-agent-metrics-prometheus-$latest.release.version.jar$ │ ├──$shardingsphere - agent - tracing - jaeger -${latest.release.version}.jar │ ├── shardingsphere-agent-tracing-opentelemetry-$latest.release.version.jar$ │ ├──$shardingsphere - agent - tracing - opentracing -${latest.release.version}.jar │ └── shardingsphere-agent-tracing-zipkin-${latest.release.version}.jar └── shardingsphere-agent.jar

```
* Configuration file

agent.yaml is a configuration file. The plug-ins include Jaeger, opentracing, Zipkin, opentelemetry, logging and Prometheus.
Remove the corresponding plug-in in ignoredpluginnames to start the plug-in.
```

~~~
```yaml
applicationName: shardingsphere-agent
ignoredPluginNames:
  - Jaeger
  - OpenTracing
  - Zipkin
  - OpenTelemetry
  - Logging
  - Prometheus

plugins:
  Prometheus:
    host:  "localhost"
    port: 9090
    props:
      JVM_INFORMATION_COLLECTOR_ENABLED : "true"
  Jaeger:
    host: "localhost"
    port: 5775
    props:
      SERVICE_NAME: "shardingsphere-agent"
      JAEGER_SAMPLER_TYPE: "const"
      JAEGER_SAMPLER_PARAM: "1"
  Zipkin:
    host: "localhost"
    port: 9411
    props:
      SERVICE_NAME: "shardingsphere-agent"
      URL_VERSION: "/api/v2/spans"
      SAMPLER_TYPE: "const"
      SAMPLER_PARAM: "1"
  OpenTracing:
    props:
      OPENTRACING_TRACER_CLASS_NAME: "org.apache.skywalking.apm.toolkit.opentracing.SkywalkingTracer"
  OpenTelemetry:
    props:
      otel.resource.attributes: "service.name=shardingsphere-agent"
      otel.traces.exporter: "zipkin"
  Logging:
    props:
      LEVEL: "INFO"
~~~

- Parameter description:

| Name | Description | Value range | Default value |
|---|---|---|---|
| JVM _INFORMATION_CO LLECTOR_ENABLED | Start JVM collector | true、false | true |
| SERVICE_NAME | Tracking service name | Custom | shardi ngsphere-agent |
| JAEG ER_SAMPLER_TYPE | Jaeger sample rate type | const、proba bilistic、ratel imiting、remote | const |
| JAEGE R_SAMPLER_PARAM | Jaeger sample rate parameter | const:0、1, pr obabilistic:0.0 - 1.0, ratelimiting: > 0, Customize the number of acquisitions per secon d, remote: need to customize the remote service addres,JA EGER_SAMPLER_MA NAGER_HOST_PORT | 1（const type） |
| SAMPLER_TYPE | Zipkin sample rate type | const、co unting、ratelim iting、boundary | const |
| SAMPLER_PARAM | Zipkin sampling rate parameter | const:0、1, counting:0.01 - 1.0, ratelimiting: > 0, boundary:0.0001 - 1.0 | 1（const type） |
| otel.reso urce.attributes | open-telemetry properties | String key value pair (, split) | servi ce.name=shardi ngsphere-agent |
| otel. traces.exporter | Tracing expoter | zipkin、jaeger | zipkin |
| otel .traces.sampler | Open-telemetry sample rate type | alway s_on、always_of f、traceidratio | always_on |
| otel.tra ces.sampler.arg | Open-telemetry sample rate parameter | tr aceidratio：0.0 - 1.0 | 1.0 |

## Used in DBPlusEngine-Proxy

- Startup script

Configure the absolute path of shardingsphere-agent.jar to the start.sh startup script of shardingsphere proxy.

```
nohup java ${JAVA_OPTS} ${JAVA_MEM_OPTS} \
-javaagent:/xxxxx/agent/shardingsphere-agent.jar \
-classpath ${CLASS_PATH} ${MAIN_CLASS} >> ${STDOUT_FILE} 2>&1 &
```

- Launch plugin

```
bin/start.sh
```

After normal startup, you can view the startup log of the plugin in the DBPlusEngine proxy log, and you can view the data at the configured address.

## 2.12 Traffic Dual Routing

### 2.12.1 Background

In complex business scenarios, architects usually use a hybrid deployment architecture to flexibly build application systems suitable for various scenarios by mixing DBPlusEngine-Driver and DBPlusEngine-Proxy, while using a unified registry to configure the sharding strategy.

DBPlusEngine-Driver adopts a decentralized architecture and shares resources with applications. It is suitable for High-Performance Lightweight OLTP applications developed by Java. However, because DBPlusEngine-Driver shares resources with applications, when executing SQL that consumes more resources, the stability and performance of applications will be affected. At the same time, DBPlusEngine-Driver consumes a large number of connections. When applications and databases are deployed in different network partitions, the impact of network latency on performance will be more obvious.

DBPlusEngine-Proxy provides a unified static entry and is independent of application deployment. It is suitable for OLAP applications and scenarios of managing and maintaining sharding databases. Users use DBPlusEngine-Proxy to execute SQL that consumes more resources, which can effectively avoid affecting applications.

In order to improve the query performance and stability of the application, we can consider forwarding the SQL that consumes more resources at the DBPlusEngine-Driver access end to the DBPlusEngine-Proxy access end located in the same network as the database. The DBPlusEngine-Proxy access end calculates the query results and returns them to the application uniformly.

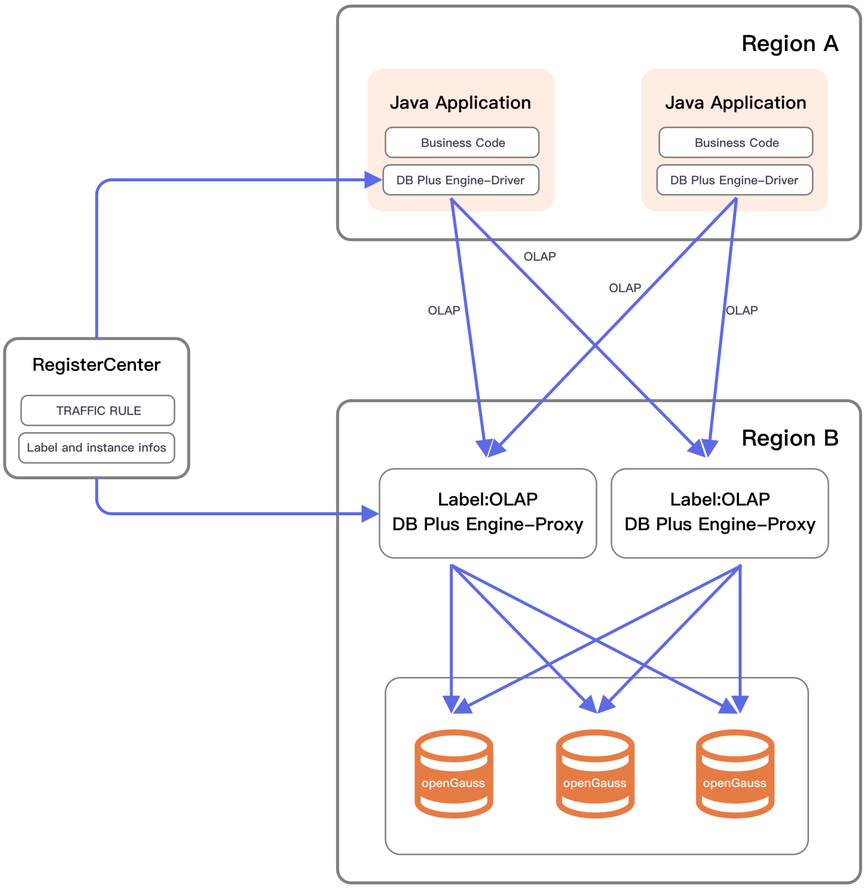


Fig. 11: Architecture

## 2.12.2 Challenges

Although the performance and stability of the application can be effectively improved by forwarding the SQL that consumes more resources at the access end of DBPlusEngine-Driver to DBPlusEngine-Proxy for execution, it also leads to more complex deployment architecture. Users need to determine which statements need to be forwarded to DBPlusEngine-Proxy, and develop relevant business logic at the DAO layer to control the forwarding of SQL.

In addition, in the scenario of starting a transaction, forwarding some SQL in the transaction to DBPlusEngine-Proxy for execution will affect the consistency and visibility of the transaction, thus affecting the use of the transaction by the business system.

## 2.12.3 Goal

The main design goal of the DBPlusEngine Traffic module is to make the impact of SQL forwarding as transparent as possible, and let the user use the mixed deployment cluster of DBPlusEngine-Driver and DBPlusEngine-Proxy as much as possible.

## 2.12.4 Core Concept

### Tag

The tag attribute configured for the DBPlusEngine-Proxy instance is used to distinguish instances. When the DBPlusEngine-Driver turns on the traffic function, the forwarding target is the DBPlusEngine-Proxy instance corresponding to the tag.

### Forwarding Strategy

For SQL at the access end of DBPlusEngine-Driver, use the strategy of forwarding, includes: target proxy instance label, forwarding algorithm and load balancing algorithm. According to different algorithms, the forwarding strategy can be divided into transaction forwarding strategy and ordinary forwarding strategy. When the algorithm in the strategy is configured as TransactionTrafficAlgorithm the forwarding strategy is transaction forwarding strategy, and when the algorithm in the strategy is configured as other algorithms, the forwarding strategy is ordinary forwarding strategy.

### Forwarding Algorithm

Algorithms used to determine whether the current SQL needs forwarding include HintTrafficAlgorithm, SegmentTrafficAlgorithm and TransactionTrafficAlgorithm.

- HintTrafficAlgorithm based on SQL Hint function to do SQL forwarding.
- SegmentTrafficAlgorithm forwards based on SQL statements, and internally provides forwarding algorithms based on SQL strings and SQL regular matching.
- TransactionTrafficAlgorithm is an algorithm specially used to handle how to forward SQL statements in a transaction when a transaction is started. At present, it supports FIRST_SQL, JDBC and Proxy three strategies. FIRST_SQL means the first forwarding result based on transaction statement, to forward SQL statement in the transaction. JDBC indicates that all SQL of transaction statements are forwarded to the JDBC access terminal for execution. Proxy means that all SQL of transaction statements are forwarded to the proxy access terminal for execution.

#### Load Balancing Algorithm

The load balancing algorithm forwards SQL statements to different proxy instances corresponding to tags for execution. At present, it has two load balancing algorithms: RANDOM and ROUND_ROBIN.

### 2.12.5 Use Norms

#### Supported

- Support forwarding of all commonly used SQL.

#### Unsupported

- SQL that is not supported by the kernel function configured by the user is still not supported after forwarding.
- It does not support forwarding SQL to different access terminals or proxy instances for execution after starting a transaction.

### 2.12.6 Related Reference

Configuration of Traffic Rule

## 2.13 Login Authentication

### 2.13.1 Overview

In order to ensure the security of user data and distributed configuration information, DBPlusEngine-Proxy provides the user authentication feature, which cannot be turned off. Otherwise, the unauthenticated client connection will be rejected. Currently, DBPlusEngine-Proxy supports a variety of user authentication protocols, including:

- MySQL Client: mysql_native_password, mysql_clear_password
- PostgreSQL Client: MD5, password
- openGauss Client: scram-sha-256

At the same time, in order to facilitate the unified identity management of enterprise users, DBPlusEngine also provides LDAP (Lightweight Directory Access Protocol) authentication. LDAP authentication has supported MySQL and PostgreSQL clients.

### 2.13.2 Characteristic

Users do not need to consider which protocol to choose when using DBPlusEngine-Proxy. The protocol negotiation process is automatically completed between DBPlusEngine and the client.

- When using MySQL client, mysql_native_password is used by default. Only when the user needs LDAP authentication, DBPlusEngine communication will require the client switch to mysql_clear_password.
- When using PostgreSQL client, MD5 is used by default. Only when the user needs LDAP authentication, DBPlusEngine communication will require the client switch to password.

## 2.14 Authority Control

### 2.14.1 Definition

DBPlusEngine provides distributed cooperation capability for the database. Concurrently, some database features are abstracted to the upper layer for unified management, to reduce user difficulties and improve operation efficiency.

Authority control is one of these capabilities.

The following is a list of some of the benefits for giving authority control to DBPlusEngine for unified management: - Avoid confusion for users when accessing heterogeneous resources, while eliminating the worry of which dialect to use for management. - Use logical database and logical table for authorization management, which is isolated from the lower real database table, making it more convenient for users to understand. - Avoid the inconsistency of authorization information caused by the change of database resources, and there will be no consumption due to information synchronization.

Therefore, in order to make authority control easier to use, the DBPlusEngine team created a new authority control system.

### 2.14.2 Related Concepts

#### User

Refers to the user of DBPlusEngine.

#### Initial user

Refers to the user set through the configuration file before DBPlusEngine is started.

#### Ordinary users

Corresponding to the initial user, ordinary users are dynamically created during the operation of DBPlusEngine.

#### Role

Role is a named collection of a certain number of authorities. Role based authority control can simplify the process of user authority management.

#### Privilege

Refers to the power of the user to perform operations on specific targets.

#### DistSQL

DistSQL (Distributed SQL) is DBPlusEngine’ s operating language. Once DBPlusEngine abstracts and unifies the authority control ability, it provides a proprietary DistSQL syntax to facilitate the administrator’ s management and maintenance of users and permissions.

#### DML

Data Manipulation Language, including INSERT, SELECT, UPDATE and DELETE statements.

#### DDL

Data Definition Language, including CREATE, ALTER, DROP and TRUNCATE statements.

### 2.14.3 Impact on the System

- Finely granular authority control

It can precisely control the operation authorities granted to each user at the database level, table level and column level.

- Unified interactive language

Use unique DistSQL to DBPlusEngine for user and authority management. No matter whether the storage node selects MySQL, PostgreSQL, openGauss or Oracle, it can carry out undifferentiated authority control.

- Authority control takes effect in real time

Changes to users or authorizations take effect in real time without restarting the DBPlusEngine.

- Authorization information is automatically synchronized in the cluster

When the user and authorization information are changed, other computing nodes in the cluster can also receive the change in real time to complete the user authorization update. The administrator does not need to repeat operations at multiple nodes to facilitate cluster management.

### 2.14.4 Principle

## Authority storage

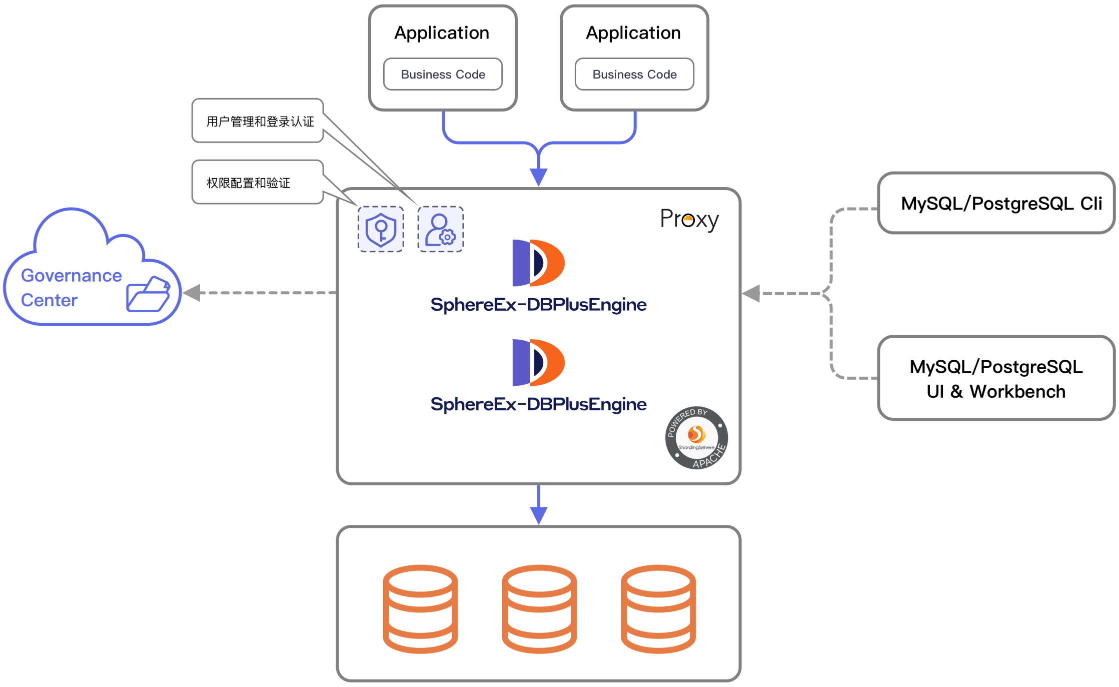


Fig. 12: Architecture

In the architecture of DBPlusEngine, the computing node (DBPlusEngine-Proxy) is stateless and does not provide data storage capacity. Therefore, the user account and authorization information will be stored in the governance center.

At the same time, thanks to the capability of the Governance Center, the information can be distributed to multiple computing nodes in the cluster in real time, which will greatly reduce the maintenance cost of users when using the cluster and provide management efficiency.

On the other hand, due to the unified authority management mechanism, the DBPlusEngine will no longer forward the received native DCL statements to the lower storage node, but will give an unsupported prompt. Users must use DistSQL provided by DBPlusEngine for account and authorization management.

## Authority provider

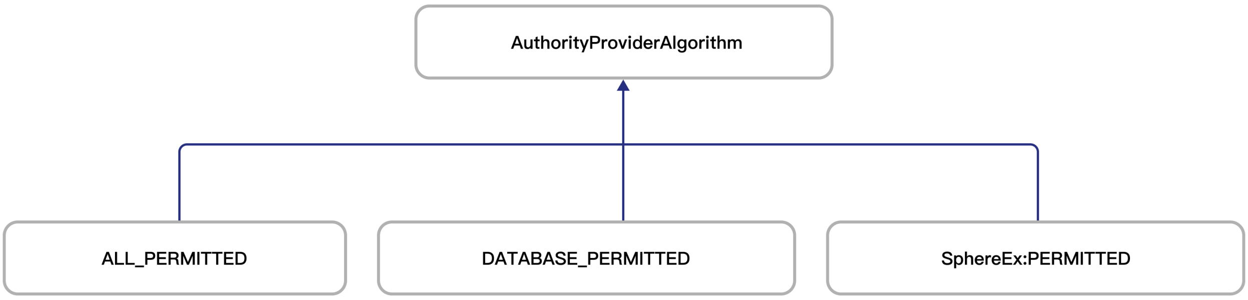


Fig. 13: Architecture

DBPlusEngine uses pluggable architecture to organize and expand features. Among them, the authority engine provides users with a variety of different authority providers, which are:

Note: The authority provider is specified by the administrator before the DBPlusEngine is started.

### Authentication process

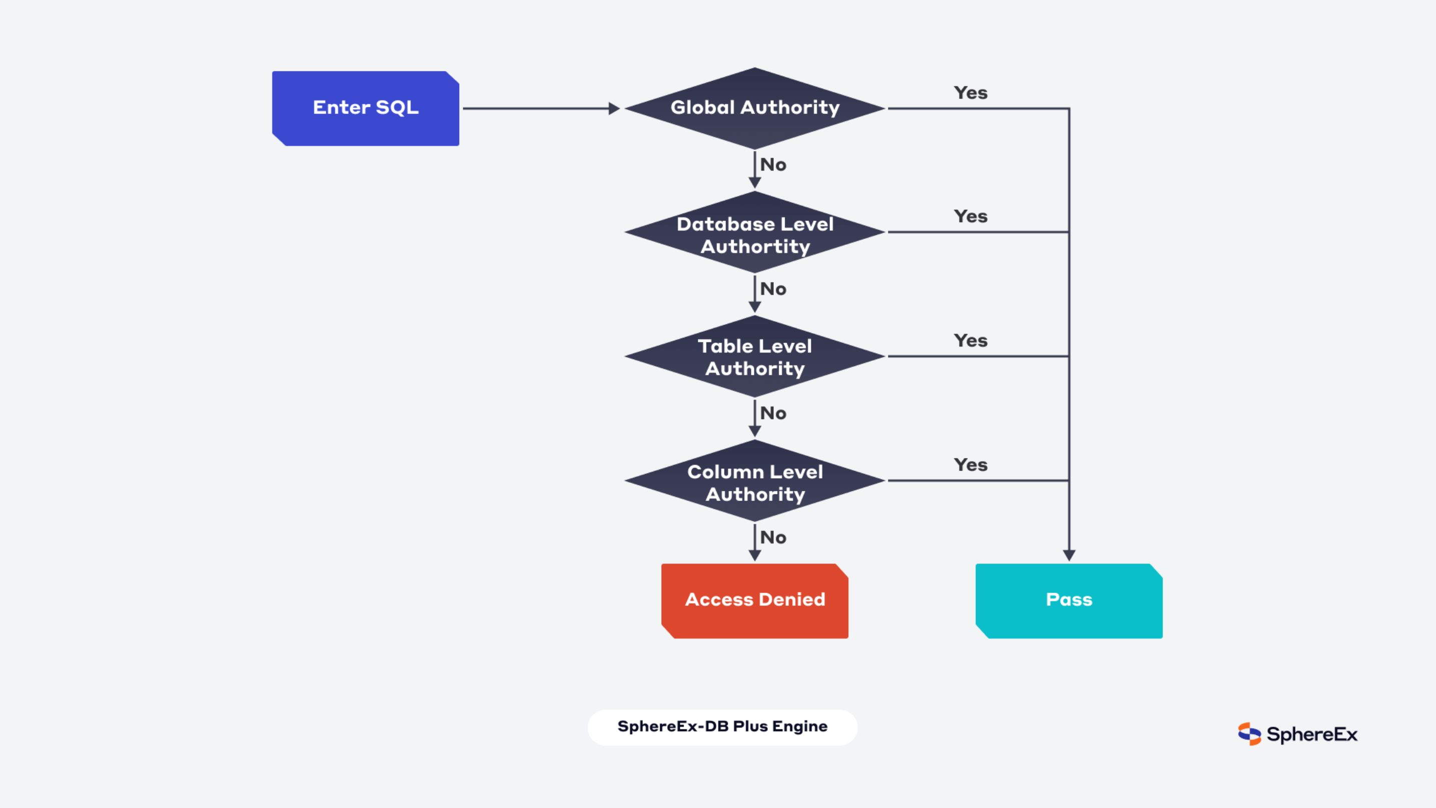


Fig. 14: Architecture

In DBPlusEngine, authority is verified level by level from top to bottom. When the user has the upper authority, it will not be checked down to ensure the authentication efficiency. Such as: - If the user has global SELECT authority, there is no need to check whether the user has the SELECT authorization of the target database table during the SELECT operation. - If the user has the INSERT authority at the database level, it is not necessary to check whether the user has the INSERT authorization of the target table during the INSERT operation.

And so on.

### 2.14.5 Relevant Reference

Configuration of Authority (Commercial Edition)

## 2.15 DistSQL

### 2.15.1 Background

DistSQL（Distributed SQL) is SphereEx-DBPlusEngine’s specific SQL, providing added-on operation capability if compared to standard SQL.

### 2.15.2 Challenges

When using DBPlusEngine-Proxy, developers can operate data just like using a database, but they need to configure resources and rules through YAML files (or registry center). However, the format of YAML and habits changed by using registry center are not friendly to DBA.

DistSQL allows users to operate SphereEx-DBPlusEngine just like a database, transforming it from a framework and middleware for developers to a database product for DBAs.

DistSQL is divided into RDL, RQL, RAL and RUL.

- RDL (Resource & Rule Definition Language) responsible for the definition of resources and rules;
- RQL (Resource & Rule Query Language) responsible for the query of resources and rules;
- RAL (Resource & Rule Administration Language) responsible for the added-on administrator feature of hint, transaction type switch, sharding execute planning and so on.
- RUL (Resource Utility Language) responsible for SQL parsing, SQL formatting, preview execution plan and more utility functions.

### 2.15.3 Goal

**DistSQL aims at breaking the boundary between middleware and database, and let developers use SphereEx-DBPlusEngine just like a database.**

### 2.15.4 Notice

Currently DistSQL can be used with DBPlusEngine-Proxy only, not with DBPlusEngine-Driver.

### 2.15.5 Related Reference

DistSQL

## 2.16 Auto Scaling on Cloud (HPA)

### 2.16.1 Definition

In Kubernetes, HorizontalPodAutoscaler(HPA) automatically updates workload resources to automatically scale workloads to meet requirements.

SphereEx-Operator uses the HPA feature in Kubernetes and combines the relevant indicators of SphereEx-DBPlusEngine to automatically scale the capacity during operation in the kubernetes cluster.

After the auto scaling feature is enabled, the SphereEx-Operator will apply an HPA object in the Kubernetes cluster while deploying a SphereEx-DBPlusEngine cluster.

## 2.16.2 Related Concepts

### HPA

HPA (HorizontalPodAutoscaler) refers to that SphereEx-Operator uses the HPA function in Kubernetes to auto scale the capacity of SphereEx-DBPlusEngine cluster.

## 2.16.3 Impact on the System

After auto scaling is enabled, manually setting the replica of SphereEx-DBPlusEngine in kubernetes will not take effect. The number of replicas of SphereEx-DBPlusEngine cluster is controlled by the maximum and minimum values of HPA controller, and elastic scaling will be processed between these two values.

The minimum number of starting copies of SphereEx-DBPlusEngine will also be controlled by the minimum number of copies of HPA. After the automatic scaling feature is turned on, SphereEx-DBPlusEngine will start with the minimum number of copies of HPA.

The scaling using HPA is the horizontal scaling of SphereEx-DBPlusEngine. Horizontal scaling means that the response to the increased load is to deploy more pods.

This is different from “vertical” scaling. For kubernetes, vertical scaling means allocating more resources (such as memory or CPU) to the pod that has been running for the workload.

## 2.16.4 Limitations

- At this stage, due to the insufficient indexes of SphereEx-DBPlusEngine, the stress load test can only be carried out through the runtime CPU. Other indicators will be added in the future to enrich the runtime pressure calculation method of SphereEx-DBPlusEngine.
- If you want to use the HPA function of SphereEx-DBPlusEngine in Kubernetes, you need to install metrics-server in your cluster and be able to use kubectl top function normally.
- While creating the SphereEx-DBPlusEngine cluster, the SphereEx-Operator will establish load balancing in front of the SphereEx-DBPlusEngine cluster. Your application and SphereEx-DBPlusEngine are linked through load balancing.
- Because SphereEx-DBPlusEngine establishes long links with your application, it will not significantly reduce the load on the existing long links in the process of scaling-out. The effect of scaling-out will only take effect on the newly established links after scaling-out.
- There will also be corresponding problems while scaling-in. In the process of scaling-in, your application will flash, because the reduced SphereEx-DBPlusEngine copy is in the process of scaling-in, it will be removed from the load balance, and the long link between your application and SphereEx-DBPlusEngine will also be destroyed.

## 2.16.5 How it works

The scaling using HPA is the horizontal scaling of SphereEx-DBPlusEngine. Horizontal scaling means that the response to the increased load is to deploy more pods.

This is different from “vertical” scaling. For kubernetes, vertical scaling means allocating more resources (such as memory or CPU) to the pod that has been running for the workload.

If the load is reduced and the number of pods is higher than the configured minimum value, horizontalpodautoscaler will indicate that the workload resources (deployment, statefulset, or other similar resources) are reduced.

In the kubernetes cluster, a controller will query the indicators in the HPA created by related resources at a certain interval. After meeting the threshold of the indicator, the corresponding resources will be scaling-out or scaling-in according to the calculation formula.

In the working process of SphereEx-Operator, HPA object acts on the deployment object of SphereEx-DBPlusEngine, and continuously queries the CPU utilization of each copy of SphereEx-DBPlusEngine.

The CPU utilization of SphereEx-DBPlusEngine obtains the CPU usage from the container /sys/fs/cgroup/cpu/cpuacct.usage, and the value set in the automaticScaling.target field in the shardingsphere.sphere-ex.com/v1alpha1.proxy is used as the percentage of the threshold value for continuous calculation.

When the calculated value reaches the threshold, the HPA controller calculates the number of copies according to the following formula:

```
Expected number of copies = ceil[Current number of copies * (Current indicators / Expected indicators)]
```

It is worth noting that the CPU utilization index is the CPU value in the resources.requests field of each copy.

Before checking the tolerance and determining the final value, the control plane will also consider whether any indicators are missing and how many pod are ready.

When CPU metrics are used to scale, any pod that is not ready (still initializing, or may be unhealthy) or the latest indicator measure is collected in the pod before the ready state, the pod will also be shelved.

Schematic diagram:

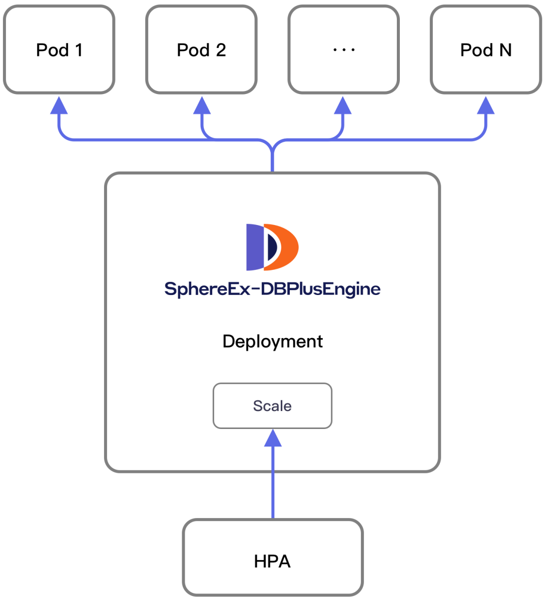


Fig. 15: HPA

### 2.16.6 Related References

Configuration of Auto Scaling on Cloud

# 3

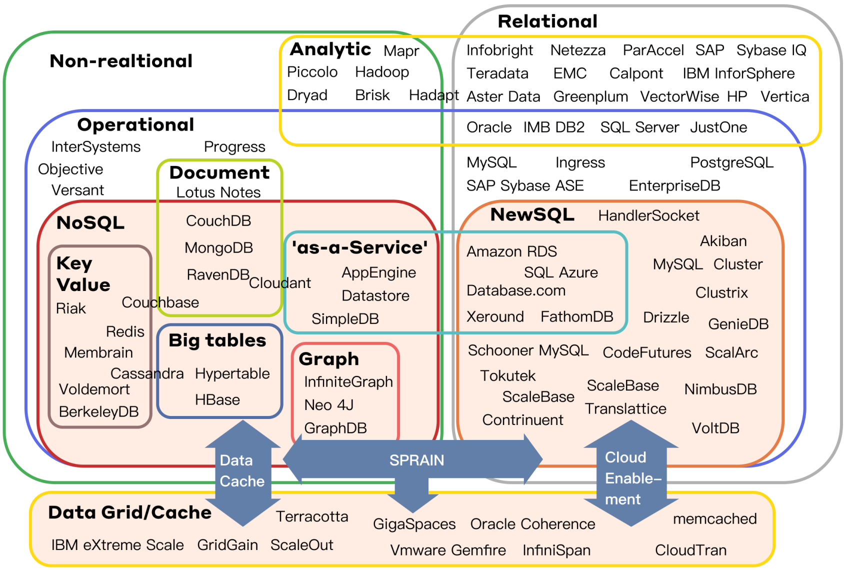
# Technical White Paper

## 3.1 Industry Trends

**Amidst the trend of database fragmentation, DBPlusEngine makes the connection between data and services easier.**

In recent years, driven by newly developing application scenarios, database technology has developed rapidly, and many excellent products have emerged. As the core of digital infrastructure, database plays an increasingly important role.

The database architecture of enterprises is gradually moving towards open source, distributed and cloud. At the same time, in terms of database categories, enterprises have changed from the original “unified” architecture to today’s multiple databases. The coexistence of multiple databases has become a normal and irreversible trend.



Twenty years ago, combined with the technological development, business complexity and volume at that time, it was not difficult for enterprises to choose a commercial database to meet all business scenarios.

Nowadays, business development drives architecture transformation, and the “unified” data architecture has been gradually abandoned by the times.

Under the trend of database fragmentation in IT architecture, we should not only deal with the increasing pressure from the business side, but also meet the diversified needs in the transformation of enterprise technology route.

Considering the various new challenges faced by the database, we can consider using an efficient and low-cost way to enhance the performance of the existing database. At the same time, we can flexibly supplement the functions of the database in combination with the business needs, and try to make the heterogeneous database transparent to the upper business. This is one of the best answers.

## 3.2 Principles

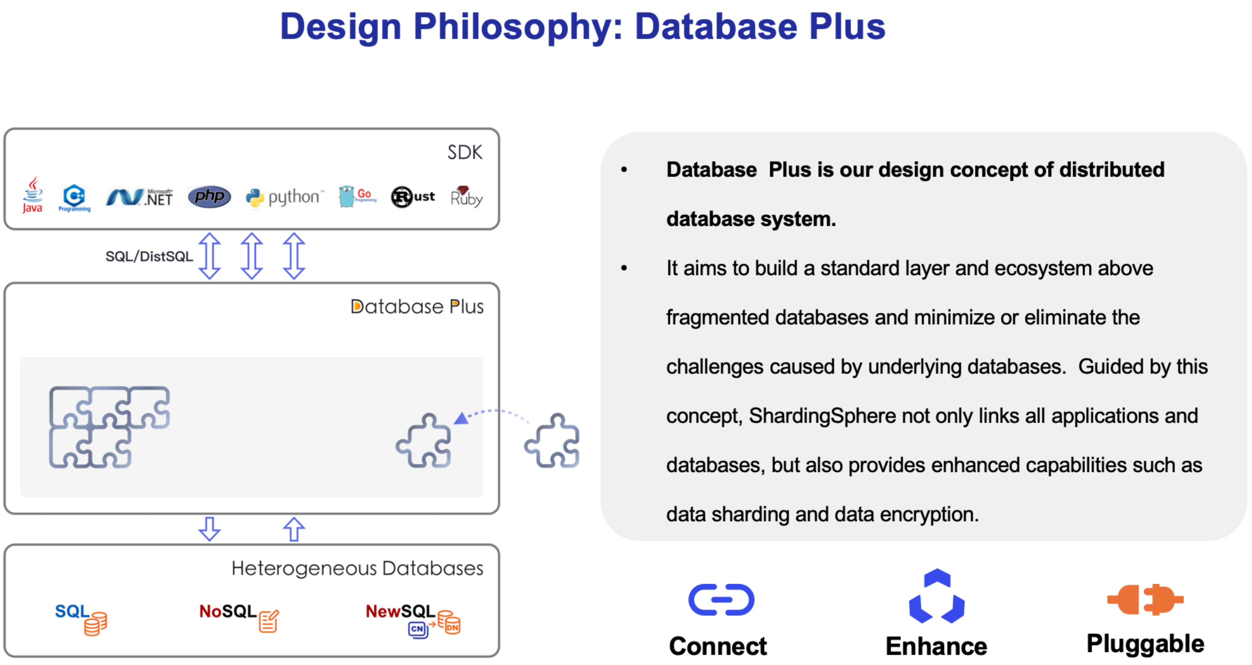
**DBPlusEngine adopts the database plus design philosophy, which is committed to building the standards and ecology of the upper layer of the database and supplementing the missing capabilities of the database in the ecology.**

- Database Plus: It is the design philosophy of ShardingSphere;
- ShardingSphere: It is the engineering implementation of Database Plus;
- DBPlusEngine: It is an enhanced product for enterprise users based on ShardingSphere.

As a guiding concept for creating a distributed database system, Database Plus establishes a new connection relationship on the database that has shown a trend of fragmentation. This process will provide data enhancement services in the upper layer of the database without intrusion, such as distribution, data control and flow control.

For the storage layer, the original database technology stack needs to be retained for users, the stability, compatibility and operation and maintenance habits have not changed, and the reliability is better guaranteed.

In this way, the application only needs to talk with the standard service layer, and does not need to care about the differences and capabilities between the underlying databases, forming a closed-loop ecological environment.



Design Philosophy: Database Plus

- **Database Plus is our design concept of distributed database system.**
- It aims to build a standard layer and ecosystem above fragmented databases and minimize or eliminate the challenges caused by underlying databases. Guided by this concept, ShardingSphere not only links all applications and databases, but also provides enhanced capabilities such as data sharding and data encryption.

The three core features of Database Plus are connectivity, enhancement, and pluggability. From “point” to “line”, and finally form an ecological “face”.

- Connect: Create database upper level standard

Connectivity is the foundation "point" of database plus capabilities. That is, in the form of "database gateway", it provides a unified database entry for the application system and an intermediate layer that can be adapted to various database SQL dialects and access protocols. Database plus parses the SQL into AST (abstract syntax tree) and regenerates the SQL according to the rules of other database dialects. It can shield the dialect differences of heterogeneous databases and the differences of heterogeneous development languages in services.

Upward, support multiple development languages with standard database protocols, so that applications can be used smoothly without feeling; Downward, it supports a variety of database products, which not only meet the requirements of multiple development languages in the north, but also support a variety of database products in the south. At present, it supports MySQL, PostgreSQL, openGauss and other database protocols, as well as MySql, PostgreSQL, openGauss, SQL Server, Oracle and all SQL dialects that support the SQL92 standard.

- Enhance: Database computing enhancement engine

Based on the connected "point", it can be further extended to this enhancement "line".

The practical experience of traditional databases is the essence precipitated by time and multiple scenarios. Database plus breaks through its computing power, capacity and functions on the basis of reusing the storage and native computing power of databases.

After the traffic enters the gateway, Database Plus enhances the three aspects of distribution, data control and traffic control through the global capability.

Data sharding, elastic scaling, high availability, read-write splitting, distributed transactions and Heterogeneous Database Federation query based on vertical splitting are all enhanced capabilities that Database Plus can provide to users in a permutation and combination manner at the global level of distributed heterogeneous databases.

- Pluggable: Building database function ecology

Facing users, Database Plus presents in the form of a "face", that is, an ecological face.

We can try to understand this "face" as a well-known App store. You only need to select the required functions from a large number of plug-ins. It can be used alone or in combination. Even you can create your own customized plug-ins in the App store.

Through the pluggable system, database plus will be able to truly build a database oriented functional ecosystem and unify the global capabilities of heterogeneous databases. It not only faces the distributed of centralized database, but also faces the integration of shaft functions of distributed database.

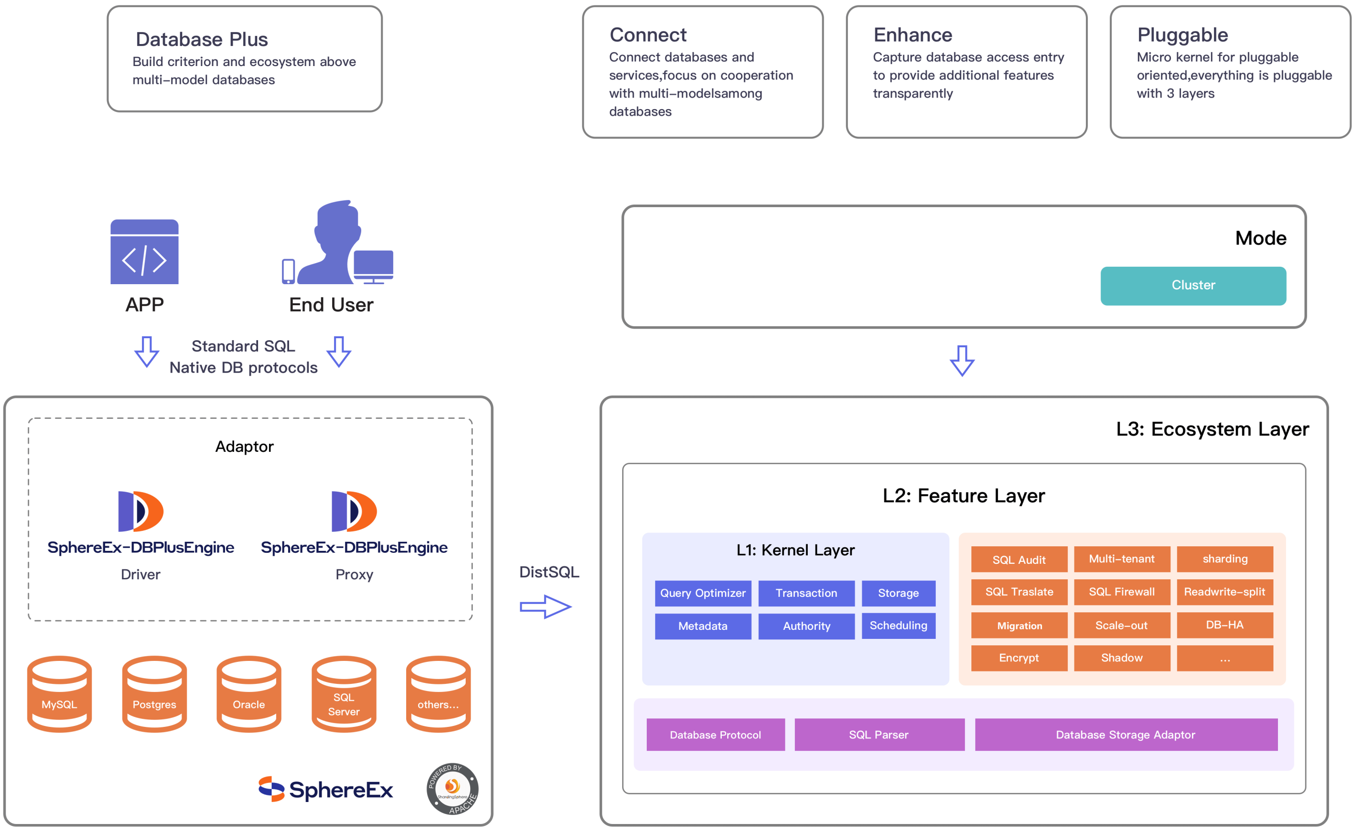
The pluggable feature can not only provide better expansion, but also achieve convergence, and only provide the content required by users. Ecological borderless, and capacity is pluggable.

## 3.3 Technical Architecture

### 3.3.1 Overall Architecture

The pluggable architecture of DBPlusEngine is divided into three layers: L1 kernel layer, L2 feature layer and L3 ecosystem layer.

The overall architecture of DBPlusEngine is shown in the following figure.



## L1 Kernel Layer

An abstraction of basic capabilities of database. All components are required and the specific implementation can be replaced by pluggable way. It includes query optimizer, distributed transaction engine, distributed execution engine, authority engine and scheduling engine.

## L2 Feature Layer

Used to provide enhanced capability. All components are optional and can contain zero or multiple components. Components isolate each other and multiple components can be used together superimposed. It includes data sharding, readwrite-splitting, database highly availability, data encryption, shadow database and so on. The user-defined feature can be fully customized and extended for the top-level interface defined by Apache ShardingSphere without changing kernel codes.

## L3 Ecosystem Layer

Used to integrate into the current database ecosystem. It includes database protocol, SQL parser and storage adapter.

**The access modes of DBPlusEngine include driver and proxy, namely DBPlusEngine-Driver and DBPlusEngine-Proxy.**

### DBPlusEngine-Driver

The lightweight Java framework provides additional services in the JDBC layer of Java. It uses the client to connect directly to the database and provides services in the form of jar packages without additional deployment and dependency. It can be understood as an enhanced JDBC driver and is fully compatible with JDBC and various ORM frameworks.

- It is applicable to any JDBC based ORM framework, such as JPA, Hibernate, Mybatis, Spring JDBC template or direct use of JDBC;
- Support any third-party database connection pool, such as DBCP, C3P0, BoneCP, HikariCP, etc;
- It supports any database that implements the JDBC specification. Currently, it supports MySQL, PostgreSQL, Oracle, SQLServer and any database that can be accessed using JDBC.

### DBPlusEngine-Proxy

The transparent database agent provides a server version that encapsulates the database binary protocol to support heterogeneous languages. At present, MySQL and PostgreSQL (compatible with openGauss and other PostgreSQL based databases) are available. It can use any access client compatible with MySQL/PostgreSQL protocol (such as MySQL Command Client, MySQL Workbench, Navicat and DBeaver) to operate data, which is more friendly to DBAs.

- It is completely transparent to applications and can be directly used as MySQL/PostgreSQL;
- Applicable to any client compatible with MySQL/PostgreSQL protocol.

## 3.3.2 Security System

DBPlusEngine-Proxy (hereinafter referred to as Proxy) provides a complete security system, taking into account engine security and data security.

### Login Authentication

Proxy has a strict login authentication mechanism. Only authenticated users can successfully establish a connection.

**Password Authentication**

By default, Proxy uses password authentication. The login user must provide the correct user name and password.

In particular, because Proxy supports a variety of database protocols (such as MySQL, PostgreSQL, etc.), when users apply different database clients, Proxy can automatically adapt the password communication protocol to provide users with a consistent security experience in complex scenarios.

**Host Restrictions**

The administrator can restrict the login host address for Proxy users to improve the security level. For example:

```
authority:
 users:
  - user: root@127.0.0.1
    password: root
```

The above configuration specifies that the root user can only access the proxy from the address 127.0.0.1. When logging in from another address, even if the password is correct, it will be rejected.

**LDAP Authentication**

To facilitate unified authentication management for enterprise users, proxy also provides LDAP (Lightweight Directory Access Protocol) authentication. LDAP authentication now supports MySQL and PostgreSQL clients.

At the same time, Proxy allows users to access LDAP in a very flexible way, such as:

1. It can be configured to use LDAP by default, so that all users can pass LDAP authentication;

2. It supports configuring auth attribute for users, specifying that users use password or LDAP authentication. Each user can use different methods;
3. Each user can use different LDAP authenticators, that is, connect to different LDAP services;
4. It supports specifying DN templates for users to meet the needs of complex scenarios;
5. Support LDAPS protocol, which can further improve the security level.

### Security of Management

In DBPlusEngine-Proxy, users can perform multiple dimension management operations through DistSQL, including but not limited to:

1. Proxy configuration management, such as transaction type, log switch, etc;
2. Logical database management;
3. Storage resource management;
4. Data sharding rule management;
5. Read write splitting rule management;
6. Encryption and decryption rule management;
7. Database discovery rule management;
8. Shadow rule management;
9. Metadata viewing, etc.

Due to the powerful DistSQL function, the database administrator can assign different DistSQL authorities to different users to achieve proxy management security. For example:

```
GRANT DIST SHOW SHARDING ON sharding_db.* TO 'sharding'@'%';
```

Through the above authorization statement, grant ‘sharding@%’ the authorization of ‘view sharding rules’ in the logical database sharding_db, then the user can execute ‘SHOW SHARDING TABLE RULES’, ‘SHOW SHARDING BINDING TABLE RULES’ and other sharding related RQLs in sharding_db, but cannot execute other unauthorized DistSQL.

For example, if the’ sharding@%’ user executes the CREATE SHARDING TABLE RULE statement at this time, he will get an exception prompt:

```
Access denied for operation [CREATE] of subject sharding_db.table_name:SHARDING.]
```

To grant all DistSQL authorization to the ‘sharding@%’ user, do the following:

```
GRANT DIST RDL,RQL,RAL ON sharding_db.* TO 'sharding'@'%';
```

### Access Security

Data access security is one of the necessary capabilities of enterprise database. As the portal of the distributed database cluster, DBPlusEngine-Proxy provides users with comprehensive access control capabilities.

Unlike traditional centralized databases or single protocol databases, DBPlusEngine has the ability to manage multi type underlying databases and connect multi protocol clients. It will face many challenges in access control:

- Different database types have different logical concepts;
- Different database types use different dialects;
- Different databases provide different storage structures.

In order to provide users with a consistent security experience, Proxy shields the differences of underlying databases and provides a unified and easy-to-use security system, which has the following characteristics:

- Fine granularity authorization management: support database level, table level and column level access control;
- Unified interactive language: use DistSQL to manage users and authorizations, which is applicable to different database protocols;
- Independent storage: the authorization information is stored in the governance center of DBPlusEngine and does not depend on the underlying database;
- Real time effective: the control of users and authorizations takes effect in real time without restart or manual refresh.

For example, if the administrator grants the ‘sharding@%’ authorization to query and write to the t_order table in sharding_db, you can execute distsql as follows:

```
GRANT DIST SELECT, INSERT ON sharding_db.t_order TO ‘sharding’@'%';
```

After the operation is completed, the ‘sharding@%’ user will get the corresponding authorization immediately. If the user performs unauthorized operations, such as DELETE, he will receive a rejection prompt:

```
=> DELETE FROM sharding_db.t_order WHERE id = 1;
Access denied for operation [DELETE] of subject sharding_db.t_order]
```

### Storage Security

In recent years, more and more attention has been paid to data security and privacy protection. Dealing with data encryption and data desensitization has also become an important task for many enterprises. Under such a trend, the technical team faces new challenges:

- Is the encryption process on the application side or in the database?
- In the case of the application side, each project team will have coding tasks, and the later modification of the algorithm will also have a great impact;
- If different databases have different encryption methods in the database, there will be a variety of encryption methods in the enterprise, which can not be reused well.

To this end, the DBPlusEngine provides a new idea. By supporting configurable encryption rules in the data engine, it not only simplifies the workload of the development team, but also provides more general data security services. The data encryption function in DBPlusEngine has many advantages:

- No invasion to the application, no need to modify the source code;
- Widely available, applicable to any database accessed through DBPlusEngine;
- A variety of encryption algorithms are available, including SHA, SM3, SM4 and other built-in algorithm support;
- Flexible customization: users can customize the encryption algorithm according to SPI to meet personalized needs.

Take the user’s mobile number encryption scenario as an example. After the application accesses the DBPlusEngine, it only needs to define an ENCRYPT RULE through YAML or DistSQL, such as:

```
CREATE ENCRYPT RULE t_user (
COLUMNS(
(NAME=mobile,CIPHER=mobile,TYPE(NAME="AES",PROPERTIES("aes-key-value"="123456abc"'")))));
```

In this way, when the application writes a new record to the t_user table, the DBPlusEngine will automatically encrypt the mobile field and store the ciphertext content. On the contrary, the ciphertext can also be decrypted automatically during query, and the query results obtained by the application will be restored to plaintext.

In this way, it not only ensures that the application is insensitive to the data encryption process, but also ensures the security of the stored data. If a hacker event such as “off database” occurs, only the ciphertext content will be leaked, and the security risk will be greatly reduced.

### 3.3.3 Architecture Advantages

- High performance

The driver side has been polished for many years, and its efficiency is close to that of native JDBC, with extreme performance. Among the mainstream competitive products, only ShardingSphere currently provides the driver access form.

- High compatibility

The agent side is applicable to any client compatible with MySQL/PostgreSQL protocol, and the driver side supports any database that implements JDBC specification.

- High expansion

In the database replacement scenario, the DBPlusEngine can meet the requirements of smooth business migration and zero intrusion into business.

- Low cost

It retains the original database technology stack, is friendly to DBA, and has low learning and management costs.

- Safe and stable

It does not interfere with the database kernel, and provides increased capacity based on the mature database base, taking into account security and stability.

- Elastic extension

With elastic computing and storage capabilities, it can meet the changing needs of computing and storage layers, and complete database splitting and migration online;

- Open ecology

The pluggable model enables the kernel, functional components and ecological docking to be pluggable and expanded in a flexible way. Users can customize unique systems suitable for business like building blocks.

## 3.4 Deployment and Maintenance

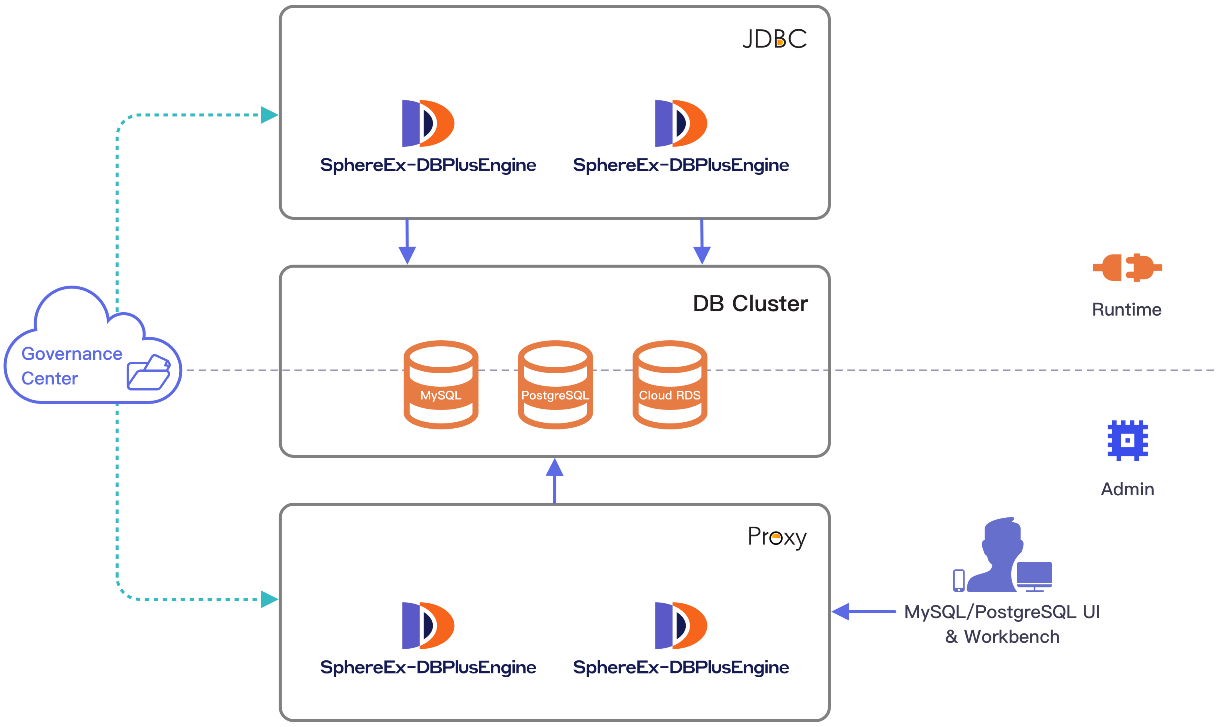
### 3.4.1 Deployment Form

The driver side focuses on performance, while the proxy side is more friendly to operation and maintenance management. In terms of deployment mode, mixed use is a better choice.

Both Apache Sharingsphere and the DBPlusEngine based on it are ecosystems composed of multiple access terminals.

- DBPlusEngine-Driver adopts a decentralized architecture, shares resources with applications, and is suitable for High-Performance Lightweight OLTP applications developed by Java;
- DBPlusEngine-Proxy It provides support for static portals and heterogeneous languages, which is independent of application deployment. It is suitable for OLAP applications and scenarios of management, operation and maintenance of sharding databases.

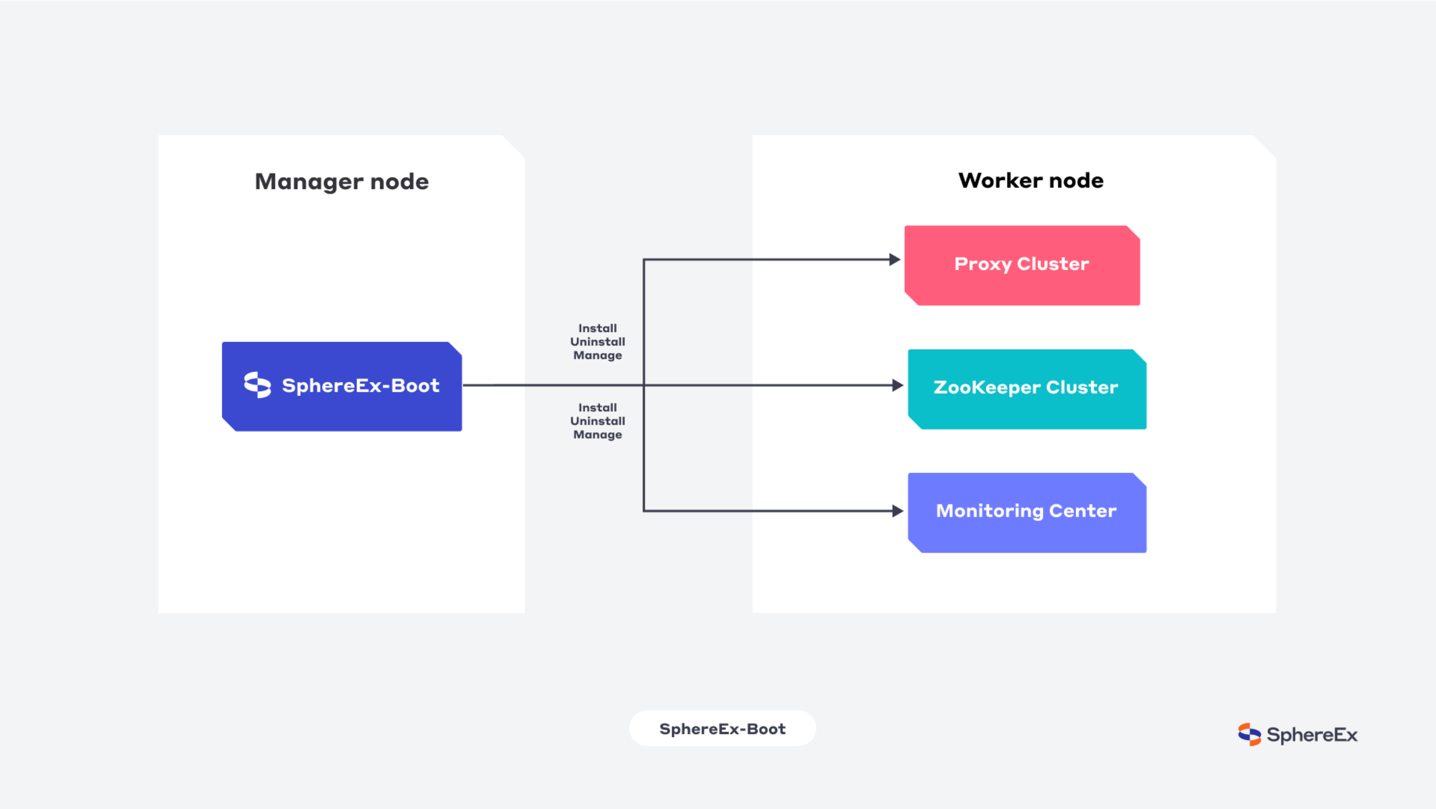
By mixing DBPlusEngine-Driver and DBPlusEngine-Proxy, and using the same registry to uniformly configure the sharding strategy, it can flexibly build application systems suitable for various scenarios, making architects more free to adjust the best system architecture suitable for the current business.



### 3.4.2 Operation and Maintenance Tool Boot

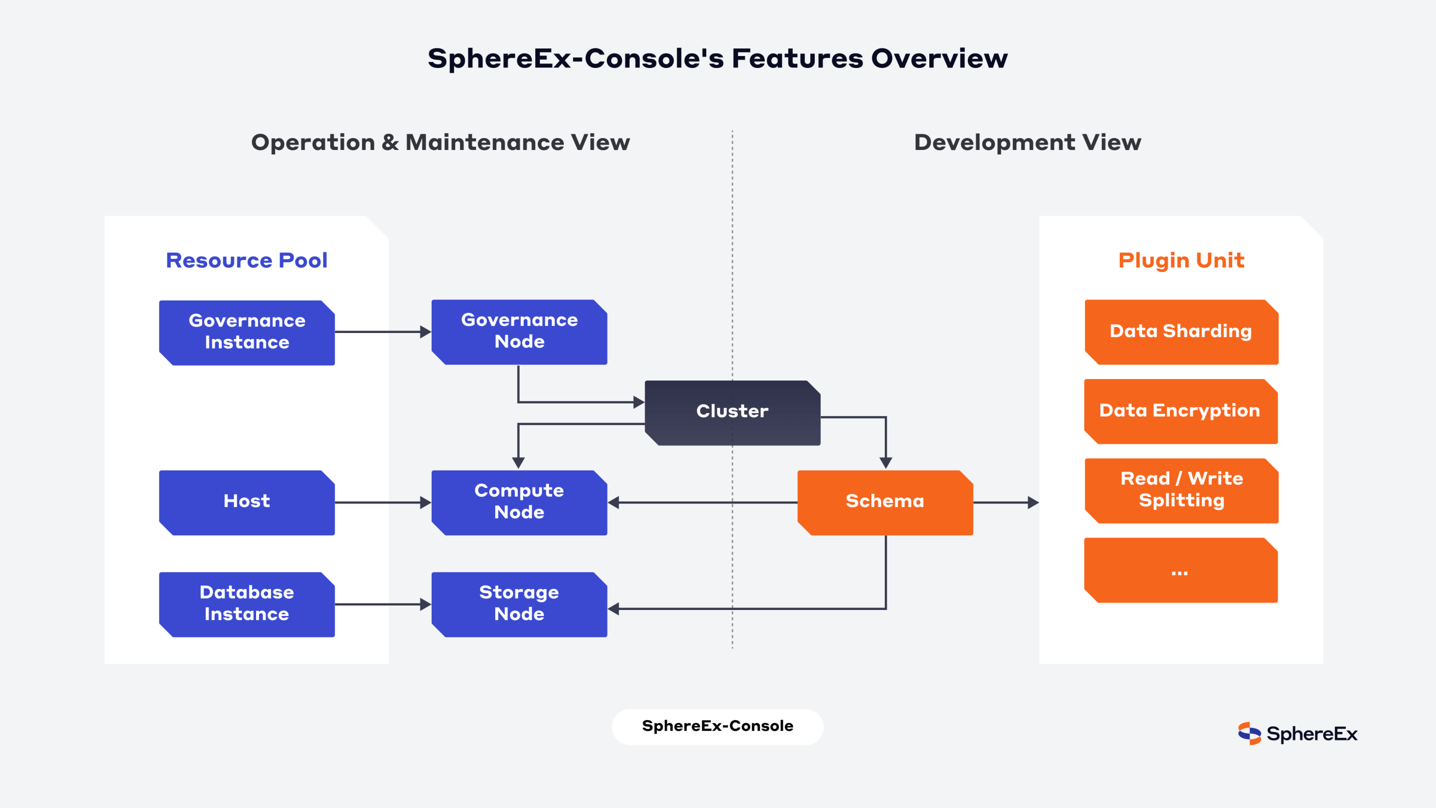
SphereEx-Boot tool is a command line tool based on Python to facilitate the management of DBPlusEngine-Proxy clusters.

The main functions of SphereEx-Boot are to install, uninstall, start, stop, view the running status, and other operations on DBPlusEngine-Proxy.



### 3.4.3 Management and Control Tool Console

SphereEx-Console is a visual operation platform applied to the management and control of DBPlusEngine, providing a more user-friendly experience. At the same time, a comprehensive solution with ShardingSphere as the core is built, packaging of multiple functions such as resource layer, instance layer and application layer, to provide users with a one-stop solution.



SphereEx-Console's Features Overview
Operation & Maintenance View
Development View
Resource Pool
Governance Instance
Host
Database Instance
Governance Node
Compute Node
Storage Node
Cluster
Schema
Plugin Unit
Data Sharding
Data Encryption
Read / Write Splitting
...
SphereEx-Console
SphereEx

# 4 Product White Paper

## 4.1 Why We Created DBPlusEngine

With the advent of digital transformation, the business needs of enterprises iterate rapidly, and the amount of data and concurrent visits increased exponentially. The traditional relational database face challenges of limited expansion capacity and low big data processing performance.

Data is becoming more and more important to enterprises. On the one hand, enterprises have increasingly urgent demand for professional services with database as the core. On the other hand, enterprises are also waiting for the new database to meet the application needs of different data scenarios.

The database is rapidly evolving. However, due to the complex technical system of enterprise database, the gap between various databases is obvious and the management is difficult. Controlling cost and improving efficiency are the long-term and constant demands of enterprises for the underlying database. Therefore, under the multi-directional role, enterprises need service providers to be able to provide horizontal mainstream database products and vertical multi-version technical service coverage, to face the demands from insensitive to solve the diversified needs of users and the common problems on the industry side and form solutions. In this case, the utilization of data and the transformation of underlying architecture are particularly urgent. It may be a feasible way to improve the flexibility and scalability of the database and build a component database.

Based on the Database Plus concept proposed by Apache ShardingSphere, SphereEx-DBPlusEngine promotes the design and implementation of Apache ShardingSphere’s microkernel and pluggable architecture model, provides community users with lightweight and flexible component open source products, and also provides a unified entrance to incremental functions for enterprise products.

SphereEx-DBPlusEngine is not satisfied with the positioning of its database middleware tool, and began to transform towards platform and ecology. The practice of DatabasePlus makes the SphereEx-DBPlusEngine not only shield the underlying details between various databases, but also make up for the differences between various shardings of distributed database. Specifically, through the flexible adaptation of database protocol, SQL dialect and database storage, developers can quickly connect applications with heterogeneous databases, so that developers do not need to pay attention to the differentiation of SQL dialect and focus on business research and development.

Based on the ecology of this layer of database, SphereEx-DBPlusEngine takes the database as the final storage node and meets various database protocols, SQL dialects and database drop docking at the access end, so as to provide various incremental capabilities to the database application architecture on this layer of storage nodes. Therefore, users can regard SphereEx-DBPlusEngine as a “database gateway”, which can obtain the access traffic of the database and provide transparent functions such as traffic redirection (data sharding, read-write splitting, shadow DB), traffic deformation (data encryption, data desensitization), traffic authentication (security, audit, authority), traffic governance (fusing, flow restriction) and traffic analysis (service quality analysis, observability), these enhancements make SphereEx-DBPlusEngine the strongest partner of database products. Users do not need to worry about how to uniformly manage various databases. The mainstream authentication and governance functions can be completed through the expansion of DistSQL.

## 4.2 Seamlessly Connect Data & Applications: SphereEx-DBPlusEngine

### 4.2.1 Overview

Based on the open source kernel of ShardingSphere, DBPlusEngine is encapsulated after adding and enhancing some enterprise level features. It can provide enterprises with enhanced data service capabilities, including but not limited to data sharding, data security, etc.

It consists of two products, DBPlusEngine-Driver and DBPlusEngine-Proxy, which can be deployed independently and or concurrently. They all provide standardized horizontal data expansion, distributed transaction, distributed governance and other features, which can be applied to various application scenarios such as Java isomorphism, heterogeneous language, cloud native and so on.

### 4.2.2 Features

DBPlusEngine is positioned as a Database Plus, and aims at building a standard layer and ecosystem above heterogeneous databases. It focuses on how to reuse existing databases and their respective upper layer, rather than creating a new database. DBPlusEngine stands at the upper level of the database and pays more attention to the cooperation between them than the database itself.

The concepts at the core of the project are Connect, Enhance and Pluggable.

Connect: flexible adaptation of database protocol, SQL dialect and database storage. It can quickly connect applications and heterogeneous databases quickly.

Enhance: capture database access entry to provide additional features transparently, such as: redirect (sharding, read/write splitting and shadow), transform (data encryption and masking), authentication (security, audit and authority), governance (circuit breaker and access limitation), and analyze (QoS and observability).

Pluggable: adopts micro kernel and 3 layers pluggable mode, so that the kernel, features and database ecosystem can be embedded flexibily. Developers can customize their ShardingSphere just like building with LEGO blocks.

### 4.2.3 Product Capability

Refer to Architecture。

## 4.3 SphereEx-DBPlusEngine Best Practices

### 4.3.1 One-click Deployment

To get started with DBPlusEngine-Proxy, you will use the SphereEx-Boot tool, which is a command-line tool developed based on Python to facilitate the management of DBPlusEngine-Proxy clusters.

Its main function is to install, start, stop, view the running status, uninstall and other related management of DBPlusEngine-Proxy. With the SphereEx-Boot tool, you can run any DBPlusEngine-Proxy cluster component with just one line of command, greatly reducing the operation and maintenance cost.

The SphereEx-Boot tool also provides standardized horizontal expansion function, which can dynamically expand the cluster anytime and anywhere by increasing the number of data servers.

### 4.3.2 Operation Visualization

SphereEx-Console is a visual operation platform applied to the management and control of the SphereEx Enterprise Data Service Platform, providing a more user-friendly experience. At the same time, a comprehensive solution with ShardingSphere as the core is built, packaging of multiple functions such as resource layer, instance layer and application layer, to provide users with a one-stop solution.

**Easy to use**

It improves the user experience and avoids configuration errors. Users do not need to operate the SphereEx Enterprise Data Service Platform through configuration and commands, which greatly improves ease of use and achieves platform "zero" bottlenecks.

**Comprehensive**

Evolving the management capabilities of the open source version, it provides a unified solution for the management and control of the SphereEx Enterprise Data Service Platform from basic resources to plugin capabilities.

**Visual monitoring**

With its data visualization & management dashboard, SphereEx Enterprise Data Service Platform's clusters, instances and hosts monitoring data are visible online and in real time.

### 4.3.3 Improve Enterprise Efficiency

- Efficiency enhancement: enhance the ability of the database without changing the original structure of the customer:
  - Obtain the access traffic of the database, and provide transparent incremental functions such as traffic redirection (data sharding, read-write splitting, shadow database), traffic deformation (data encryption, data desensitization), traffic authentication (security, audit, authority), traffic governance (fusing, flow restriction) and traffic analysis (service quality analysis, observability).
  - The project adopts micro kernel + three-tier pluggable model, so that the kernel, functional components and ecological docking can be pluggable and expanded in a flexible way, and developers can customize their own unique system like building blocks.
- Cost reduction: Based on the original architecture, SphereEx-DBPlusEngine does not introduce new database types, which will not increase the learning cost of DBA; Reusing the original architecture will not increase the procurement cost and greatly reduce the distributed operation and maintenance and transformation cost of customers.

### 4.3.4 Support Enterprises Digital Transformation

The data sharding feature of SphereEx-DBPlusEngine helps enterprises solve the problems of low performance, poor availability and high operation and maintenance cost in the scenarios with massive data using a solution of centralized storage of data to a single node in the process of digitization.

- In terms of performance, most relational databases use B + tree indexes. When the amount of data exceeds the threshold, the increase of index depth will also increase the IO times of disk access, resulting in the decline of query performance. At the same time, high concurrent access requests also make the centralized database the biggest bottleneck of the system.
- In terms of availability, the stateless nature of service can achieve the random expansion at a small cost, which will inevitably lead to the final pressure of the system falling on the database. A single data node, or a simple primary-secondary architecture, has become more and more difficult to bear. The availability of database has become the key of the whole system.

- In terms of operation and maintenance cost, when the data in a database instance reaches above the threshold, the operation and maintenance pressure on DBA will increase. The time cost of data backup and recovery will become more and more uncontrollable with the amount of data. Generally speaking, the data threshold of a single database instance is within 1TB, which is a reasonable range.

When the traditional relational database cannot meet the needs of Internet scenarios, there are more and more attempts to store data in native NoSQL that supports distributed. However, the incompatibility of NoSQL with SQL and the imperfection of ecosystem make them unable to complete a fatal blow in the game with relational database, while the position of relational database is still unshakable.

Data sharding refers to the decentralized storage of data stored in a single database in multiple databases or tables according to a certain dimension, to improve the performance bottleneck and availability. The effective means of data sharding is to divide the relational database into database and table. Both sub database and sub table can effectively avoid the query bottleneck caused by the amount of data exceeding the acceptable threshold. In addition, the sub database can also be used to effectively disperse the number of visits to a single point of the database.

Although split tables cannot alleviate the pressure of database, they can provide the possibility to transform distributed transactions into local transactions as much as possible. Once cross database update operations are involved, distributed transactions often complicate the problem. Using multi master and multi slave sharding can effectively avoid single point of data, so as to improve the availability of data architecture.

SphereEx-DBPlusEngine keeps the data volume of each table below the threshold through data splitting by database and table, and dredges the traffic to deal with high access volume. It is an effective means to deal with high concurrency and massive data systems.

## 4.4 Case study: solving a fintech user's data storage and expansion with hundreds of billions of rows

### 4.4.1 Customer Pain Points

To cope with the rapid development of business and the surge of massive amounts of data, data architectures have evolved several times. Nevertheless, with the product upgrade iteration, the earlier solution became an immediate problem. The data sharding scheme achieved through the business framework led to the increase of business code complexity, rising maintenance costs and the disadvantages of tight coupling. Each application upgrade requires more energy to accordingly adjust the sharding, making it is difficult for R & D teams to focus on the business itself. After lengthy consideration, the technical team began to consider using mature sub database and sub table components to undertake this part of the work, so that the business system upgrade and architecture adjustment are no longer complex.

### 4.4.2 Customer Research

The comparison between ShardingSphere based sharding and self-developed framework based sharding is as follows:

| | sharding based on self-developed frame | sharding based on DBPlusEngine |
|---|---|---|
| Performance | High | High |
| Code coupling | High | Low |
| Business intrusion degree | High | Low |
| Upgrade difficulty | High | Low |
| Expansibility | Commonly | Good |

Advantages of DBPlusEngine: 1. Mature and stable products 2. Ultimate performance 3. Processing massive data 4. Flexible extension of Architecture

### 4.4.3 Customer Transformation and Architecture

SphereEx-DBPlusEngine has provided more support and improvement to the function and performance of the product in the process of landing the Internet financial business of the main customers, and the product has experienced the polishing of typical cases again.

1. Upgrade SQL engine
2. Distributed primary key
3. Business sharding key value injection
4. SQL parsing result cache
5. JDBC metadata information cache
6. Use of bind table and broadcast table
7. Automatic execution engine and stream merging

Through the cooperation of the two teams, the indicators of the combination of the customer' s Internet financial business and SphereEx-DBPlusEngine meet the expectations, and the performance is almost consistent with that of native JDBC.

The reconstructed structure is as follows:

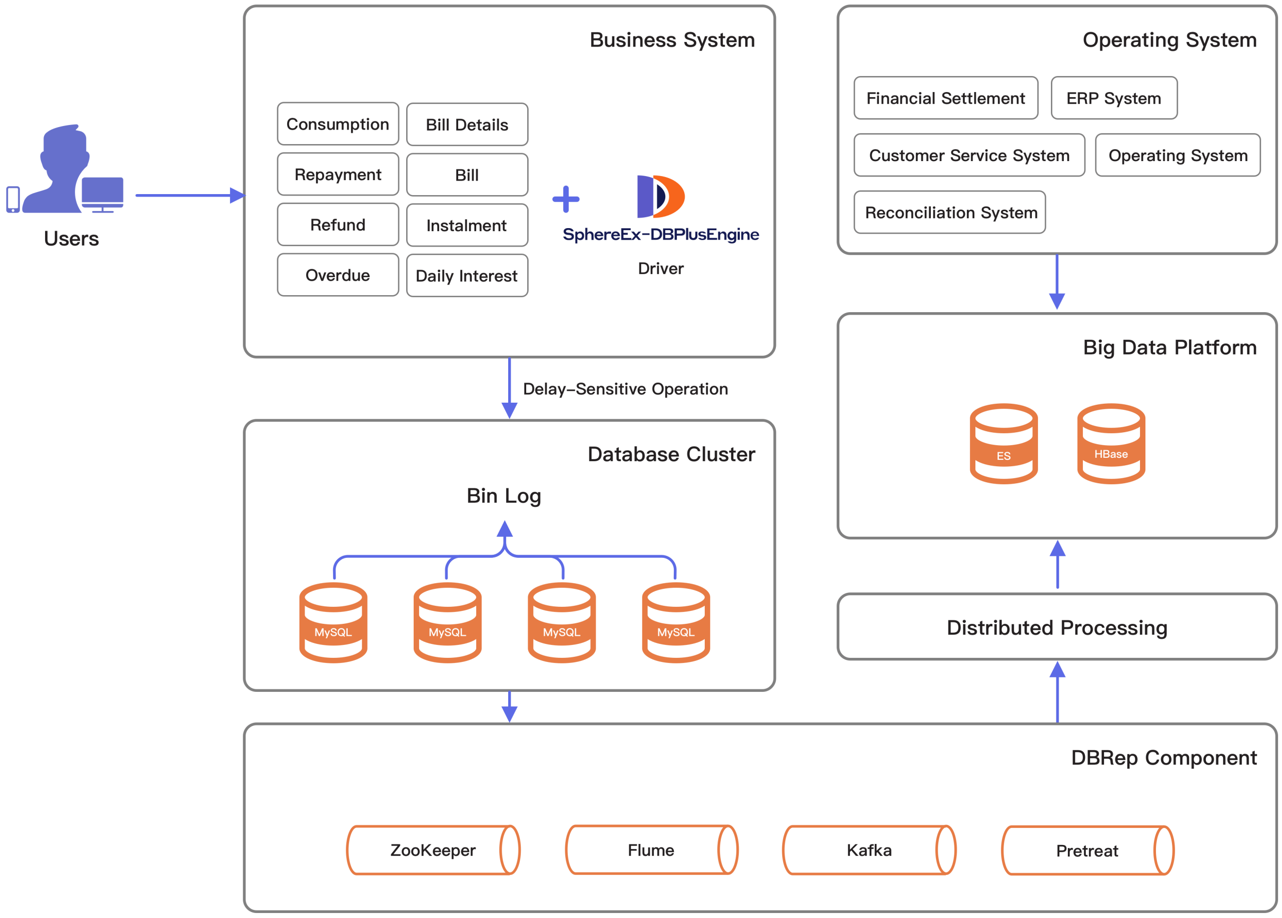


Fig. 1: Architecture

## 4.5 Service Guarantee

### 4.5.1 Service Feedback

SphereEx has a professional aftersales service team to provide you with industry leading aftersales service. The contact information can be found at contact us。

## 4.6 SphereEx-DBPlusEngine Operation Guide

Refer to Quick Start。

# 5 Performance White Paper

This document showcases the performance results of our products in benchmark tests and scenario tests.

Users can refer to this document for information about product performance and technology selection.

As the kernel and features code continue to be optimized, the benchmark test results cannot represent the optimal performance. The documents will also be updated in tandem with the code. If you' d like to evaluate the performance of our products, you can follow the test methodology below.

## 5.1 TPC-C Benchmark Test

TPC-C test scenario simulates how the online e-commerce transaction works.

Suppose there' s a large commodity wholesaler and its businesses stretch across multiple **districts** and are managed through warehouses. When its business expands, the company will add new **warehouses**.

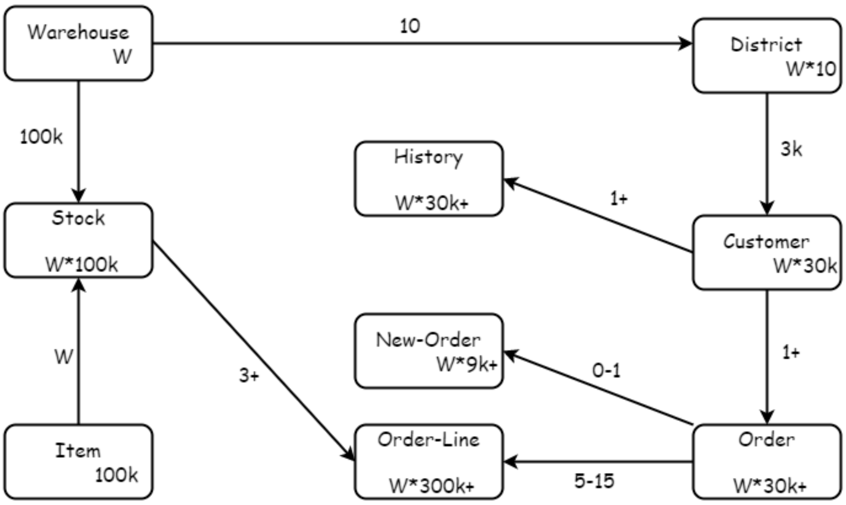
Each warehouse supplies 10 areas and each area serves 3,000 **customers**. All its warehouses maintain and record the **stock** of the 100,000 **items** the company is selling.

Each customer **order** includes 10 **order lines** on average. And around 1% of the order lines are out of stock in the warehouses they belong to, which have to be supplied by other warehouses.

Customers issue new orders to the company' s system or query their order status in the system.

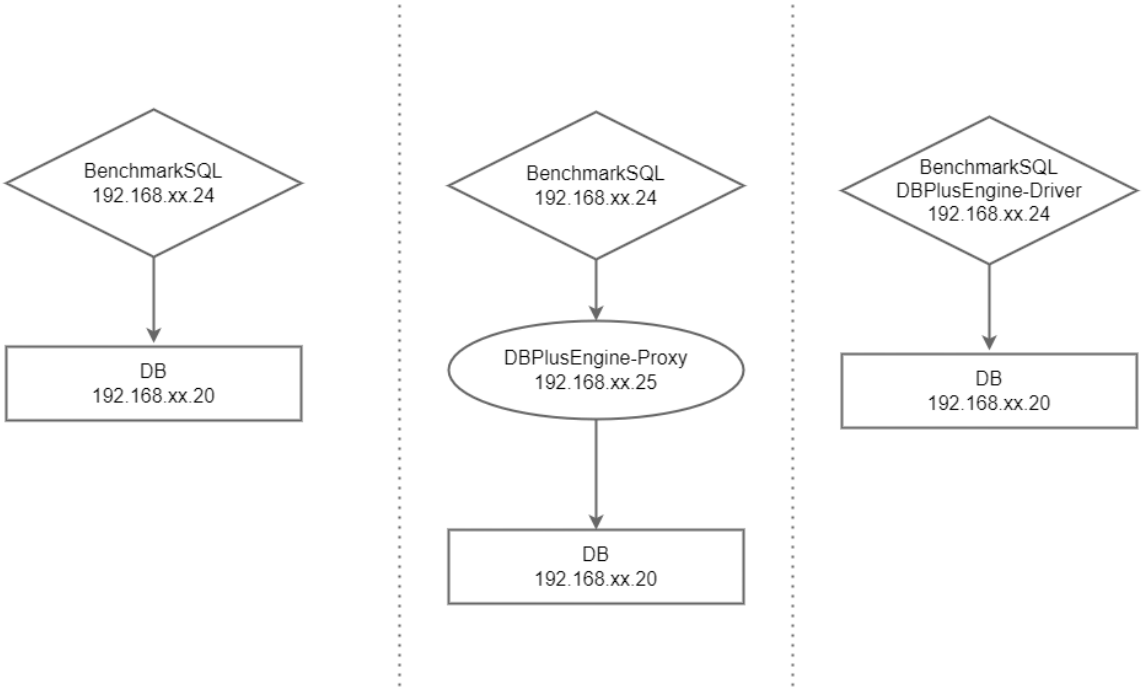
The system is also used to deal with payments and orders that have been delivered, examining the stock to find potential supply insufficiency.

The following figure illustrates the relationships between **warehouses**, **districts**, and **customers**.



### 5.1.1 Test Plan A: performance test with a single storage node and a computing node

Use BenchmarkSQL to test DBPlusEngine-Proxy and DBPlusEngine-Driver with a single storage node and compare their performance with MySQL's.



### Test Objective

BenchmarkSQL stress testing model is used to compare the performance of DBPlusEngine-Proxy, DBPlusEngine-Driver, and MySQL with the same amount of data so that users can have a detailed understanding of the performance of DBPlusEngine-Proxy and DBPlusEngine-Driver.

### Test Tool

BenchmarkSQL is a typical open-source database test tool with embedded TPC-C test scripts, which can test PostgreSQL, MySQL, Oracle, SQL Server, and other databases.

### Test Environment

| Application | IP Address | Port | Version |
|---|---|---|---|
| DBPlusEngine-Proxy | 192.168.xx.25 | 3307 | 1.2 |
| DBPlusEngine-Driver | 192.168.xx.24 | - | 1.2 |
| MySQL | 192.168.xx.20 | 13306 | 8.0.29 |

Server Configuration

| Item | Configuration |
|---|---|
| CPU | 48 C |
| Memory | 96 G |
| Hard Disk | SSD 820 G |
| JDK | 17.0.2 |

### Test Procedure

1. Create a test database.

```
mysql -utest -h192.168.xx.20 -P13306 -p
```

```
DROP DATABASE IF EXISTS test_tpcc;

CREATE DATABASE IF NOT EXISTS test_tpcc;
```

2. Install test tools.

```
wget https://udomain.dl.sourceforge.net/project/benchmarksql/benchmarksql-5.0.zip
yum -y install ant
unzip benchmarksql-5.0.zip -d  /usr/local
cd /usr/local/benchmarksql-5.0
ant
cp mysql-connector-java-8.0.24.jar benchmarksql-5.0/run/lib
```

Create props.proxy file.

```
db=postgres
driver=com.mysql.jdbc.Driver
conn=jdbc:mysql://192.168.xx.25:3307/sharding_db?useSSL=false&useServerPrepStmts=true&cachePrepStmts=true&
prepStmtCacheSize=8192&prepStmtCacheSqlLimit=8000
user=root
password=root
warehouses=200
loadWorkers=100
terminals=200
runTxnsPerTerminal=0
runMins=10
limitTxnsPerMin=0
terminalWarehouseFixed=true
newOrderWeight=45
paymentWeight=43
orderStatusWeight=4
deliveryWeight=4
stockLevelWeight=4
resultDirectory=my_result_%tY-%tm-%td_%tH%tM%tS
shardingNumber=1
```

3. YAML

```
rules:
 - !SHARDING
  bindingTables:
   - bmsql_warehouse, bmsql_customer
   - bmsql_stock, bmsql_district, bmsql_order_line
  defaultDatabaseStrategy:
   none:
  defaultTableStrategy:
   none:
  keyGenerators:
   snowflake:
```

```
  type: SNOWFLAKE
tables:
 bmsql_config:
  actualDataNodes: ds_0.bmsql_config

 bmsql_warehouse:
  actualDataNodes: ds_${0..0}.bmsql_warehouse
  databaseStrategy:
   standard:
    shardingColumn: w_id
    shardingAlgorithmName: mod_1

 bmsql_district:
  actualDataNodes: ds_${0..0}.bmsql_district
  databaseStrategy:
   standard:
    shardingColumn: d_w_id
    shardingAlgorithmName: mod_1

 bmsql_customer:
  actualDataNodes: ds_${0..0}.bmsql_customer
  databaseStrategy:
   standard:
    shardingColumn: c_w_id
    shardingAlgorithmName: mod_1

 bmsql_item:
  actualDataNodes: ds_${0..0}.bmsql_item
  databaseStrategy:
   standard:
    shardingColumn: i_id
    shardingAlgorithmName: mod_1

 bmsql_history:
  actualDataNodes: ds_${0..0}.bmsql_history
  databaseStrategy:
   standard:
    shardingColumn: h_w_id
    shardingAlgorithmName: mod_1

 bmsql_oorder:
  actualDataNodes: ds_${0..0}.bmsql_oorder
  databaseStrategy:
   standard:
    shardingColumn: o_w_id
    shardingAlgorithmName: mod_1

 bmsql_stock:
  actualDataNodes: ds_${0..0}.bmsql_stock
  databaseStrategy:
   standard:
    shardingColumn: s_w_id
    shardingAlgorithmName: mod_1

 bmsql_new_order:
  actualDataNodes: ds_${0..0}.bmsql_new_order
  databaseStrategy:
   standard:
    shardingColumn: no_w_id
    shardingAlgorithmName: mod_1

 bmsql_order_line:
  actualDataNodes: ds_${0..0}.bmsql_order_line
  databaseStrategy:
   standard:
    shardingColumn: ol_w_id
    shardingAlgorithmName: mod_1

shardingAlgorithms:
 mod_1:
  type: MOD
  props:
   sharding-count: 1
```

4. Obtain data

Initialize the database, create tables through Proxy and insert data.

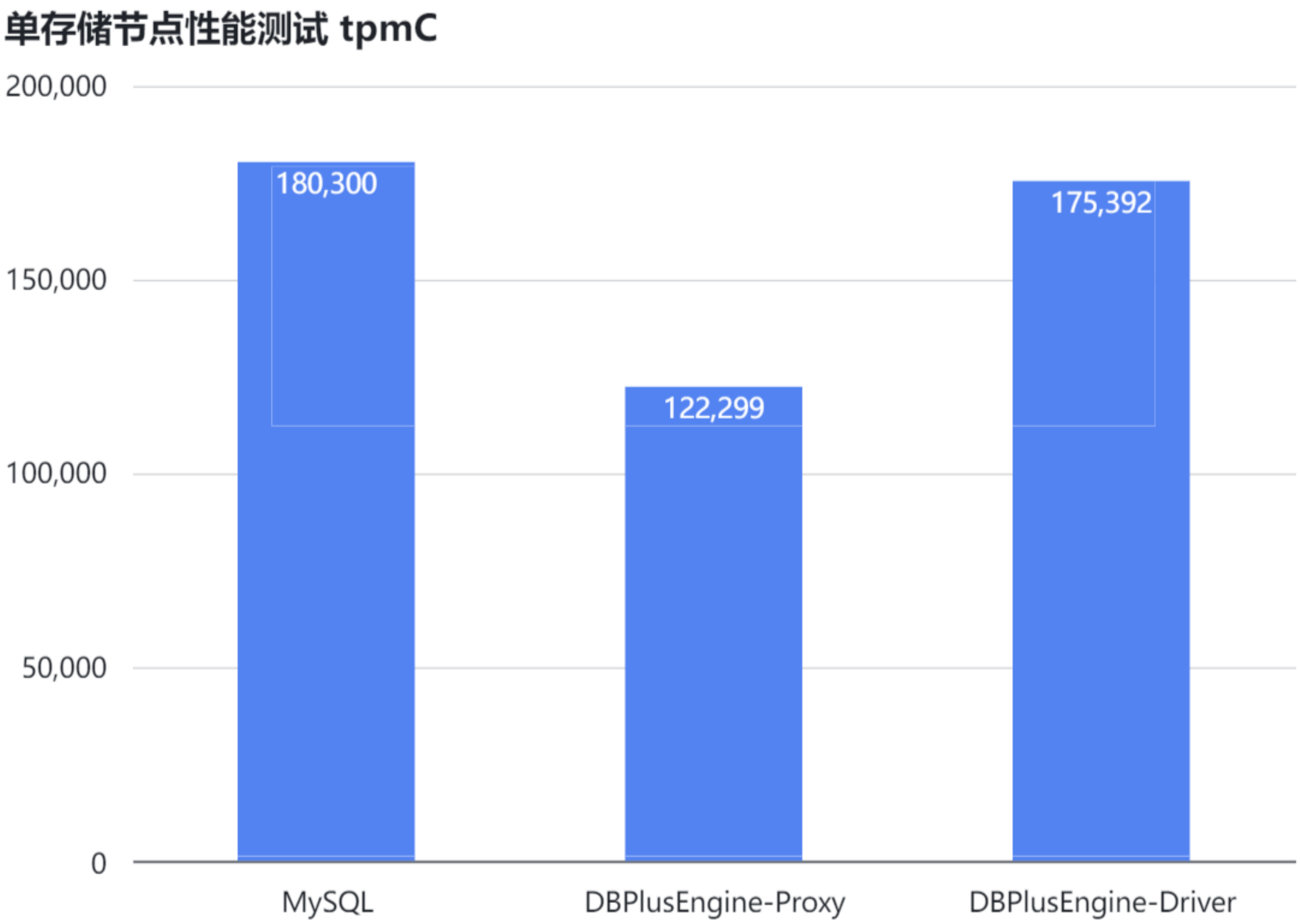
```
cd /usr/local/benchmarksql/run

./runDatabaseDestroy.sh props.proxy

./runDatabaseBuild.sh props.proxy
```

5. Carry out stress testing.

```
./runBenchmark.sh props.proxy
```

### Test Result

| Test Object | tpmC |
|---|---|
| MySQL | 180,300 |
| DBPlusEngine-Proxy | 122,299 |
| DBPlusEngine-Driver | 175,392 |

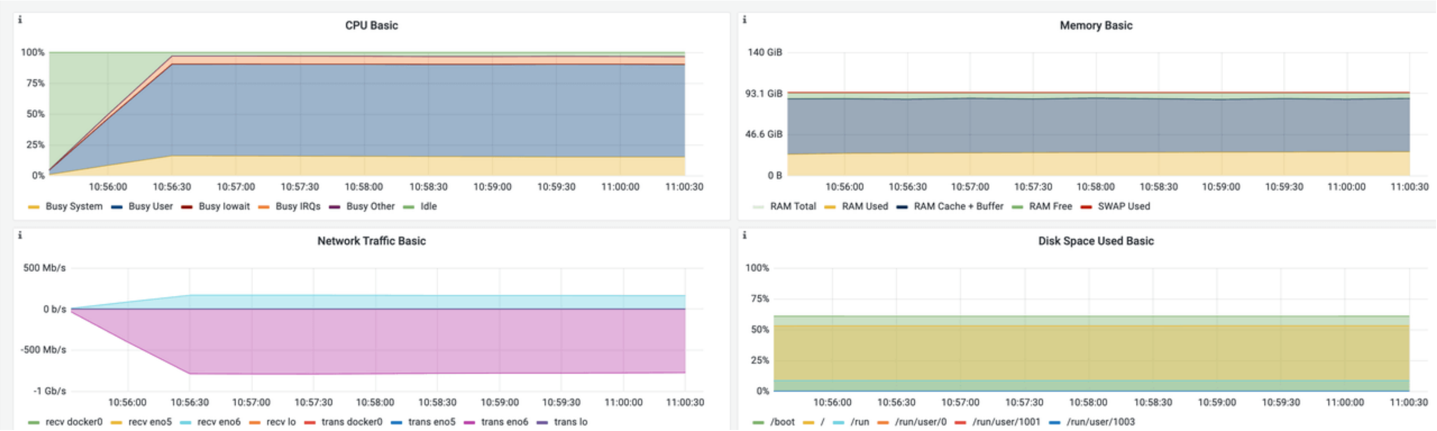


Result: with a single storage node, DBPlusEngine-Driver is slightly lower than MySQL and higher than DBPlusEngine-Proxy in terms of performance.

## Monitoring Information

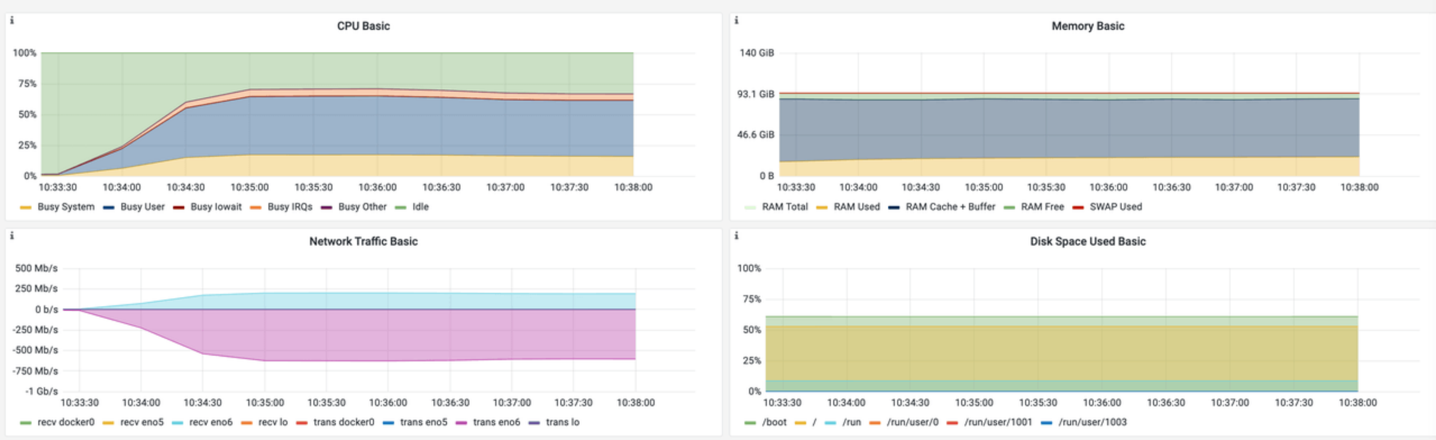
### BenchmarkSQL ——> MySQL

- MySQL

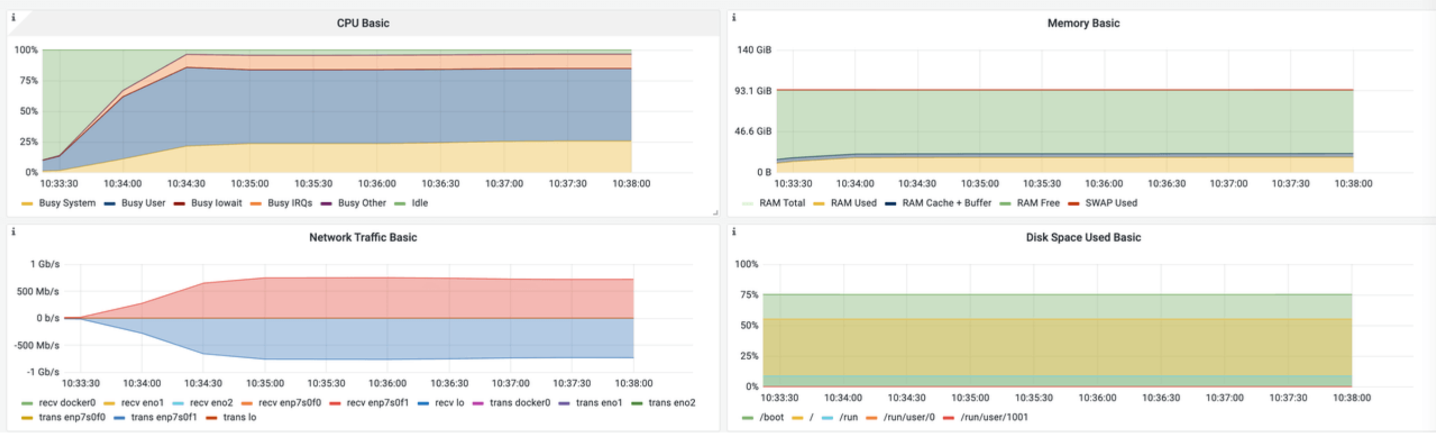


## BenchmarkSQL ——> Proxy ——> MySQL

- MySQL

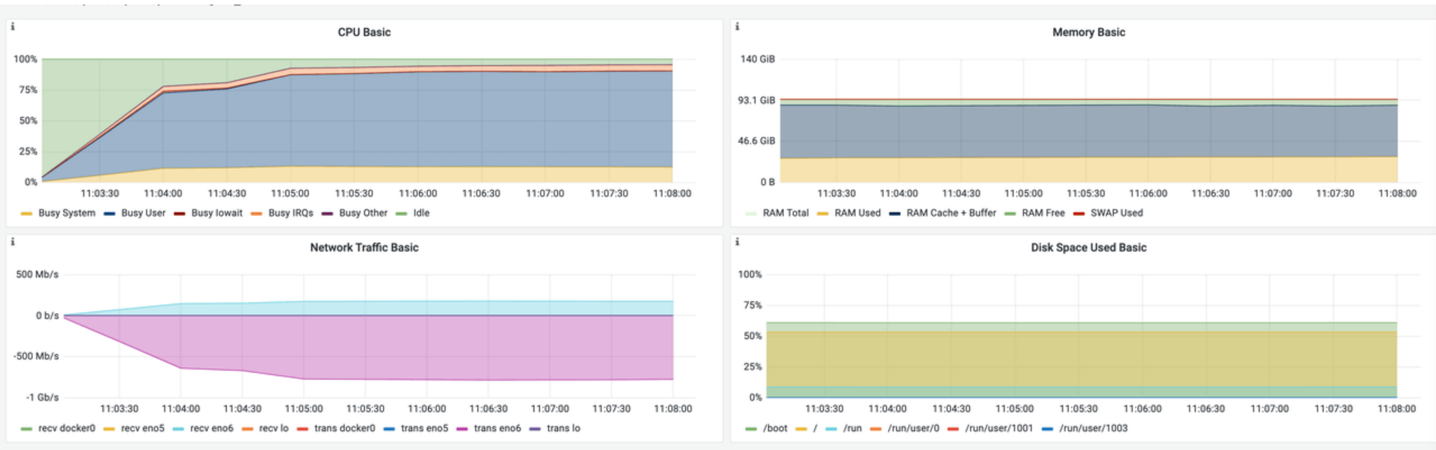


- Proxy

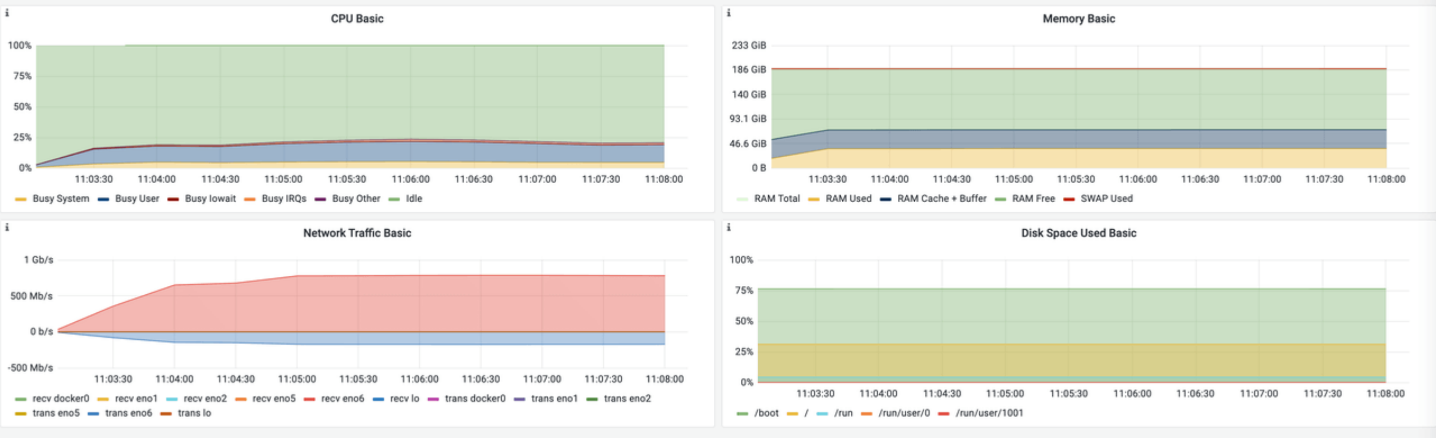


## BenchmarkSQL ——> Driver ——> MySQL

- MySQL



- Driver



### Glossary

Warehouses: the number of warehouses, which determines the amount of data.

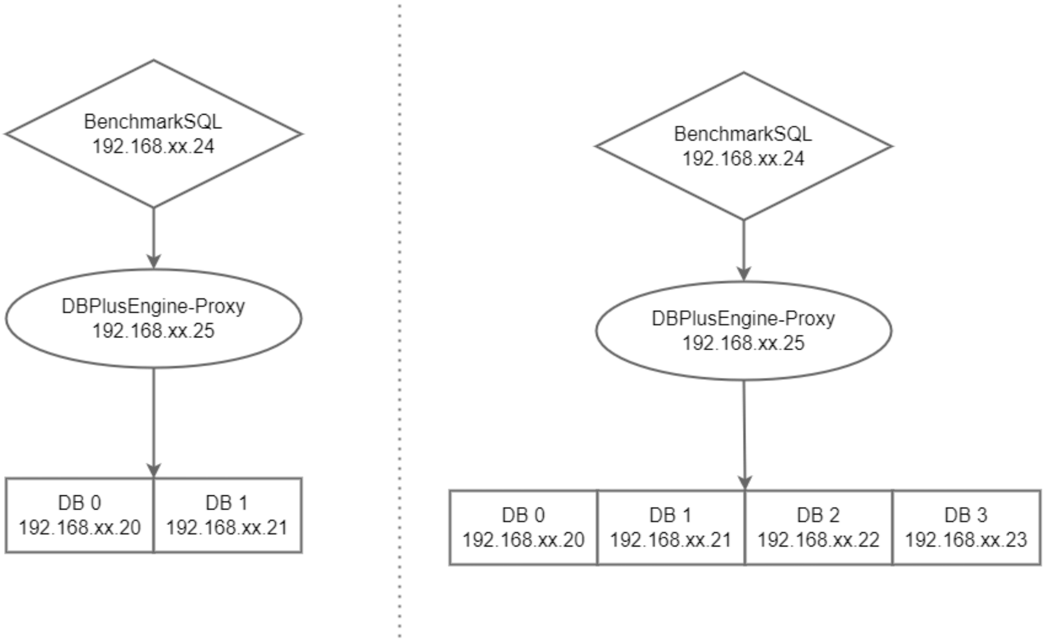
Terminals: the number of threads

runMins: test duration

tpmC: transactions per min

## 5.1.2 Test Plan B: Scale-out the storage node

Use the BenchmarkSQL tool to carry out stress testing on two and four storage nodes respectively.



## Test Objective

With the same amount of data, the BenchmarkSQL stress testing model is used to compare the results of DBPlusEngine-Proxy with two or four storage nodes. By analyzing the results, it' s easier for users to understand the advantages of sharding scenarios.

## Test Tool

BenchmarkSQL

## Test Environment

| Application | IP Address | Port | Version |
|---|---|---|---|
| DBPlusEngine-Proxy | 192.168.xx.25 | 3307 | 1.2 |
| MySQL | 192.168.xx.20 | 13306 | 8.0.29 |
| MySQL | 192.168.xx.21 | 13306 | 8.0.29 |
| MySQL | 192.168.xx.22 | 13306 | 8.0.29 |
| MySQL | 192.168.xx.23 | 13306 | 8.0.29 |

Server Configuration

| Item | Configuration |
|---|---|
| CPU | 48 C |
| Memory | 96 G |
| Hard Disk | SSD 820 G |
| JDK | 17.0.2 |

### Test Procedure

1. Create a test table.

```
mysql -utest -h192.168.xx.20 -P13306 -p
```

```
DROP DATABASE IF EXISTS test_tpcc;

CREATE DATABASE IF NOT EXISTS test_tpcc;
```

```
mysql -utest -h192.168.xx.21 -P13306 -p
```

```
DROP DATABASE IF EXISTS test_tpcc;

CREATE DATABASE IF NOT EXISTS test_tpcc;
```

```
mysql -utest -h192.168.xx.22 -P13306 -p
```

```
DROP DATABASE IF EXISTS test_tpcc;

CREATE DATABASE IF NOT EXISTS test_tpcc;
```

```
mysql -utest -h192.168.xx.23 -P13306 -p
```

```
DROP DATABASE IF EXISTS test_tpcc;

CREATE DATABASE IF NOT EXISTS test_tpcc;
```

2. Install test tools.

```
wget https://udomain.dl.sourceforge.net/project/benchmarksql/benchmarksql-5.0.zip
yum -y install ant
unzip benchmarksql-5.0.zip -d  /usr/local
cd /usr/local/benchmarksql-5.0
ant
cp mysql-connector-java-8.0.24.jar benchmarksql-5.0/run/lib
```

Create props.proxy file.

```
db=postgres
driver=com.mysql.jdbc.Driver
conn=jdbc:mysql://192.168.xx.25:3307/sharding_db?useSSL=false&useServerPrepStmts=true&cachePrepStmts=true&
prepStmtCacheSize=8192&prepStmtCacheSqlLimit=8000
user=root
password=root
warehouses=200
loadWorkers=100
terminals=200
runTxnsPerTerminal=0
runMins=10
limitTxnsPerMin=0
terminalWarehouseFixed=true
newOrderWeight=45
```

```
paymentWeight=43
orderStatusWeight=4
deliveryWeight=4
stockLevelWeight=4
resultDirectory=my_result_%tY-%tm-%td_%tH%tM%tS
shardingNumber=2
```

3. YAML

```
rules:
 - !SHARDING
  bindingTables:
   - bmsql_warehouse, bmsql_customer
   - bmsql_stock, bmsql_district, bmsql_order_line
  defaultDatabaseStrategy:
   none:
  defaultTableStrategy:
   none:
  keyGenerators:
   snowflake:
    type: SNOWFLAKE
  tables:
   bmsql_config:
    actualDataNodes: ds_0.bmsql_config

   bmsql_warehouse:
    actualDataNodes: ds_${0..1}.bmsql_warehouse
    databaseStrategy:
     standard:
      shardingColumn: w_id
      shardingAlgorithmName: mod_2

   bmsql_district:
    actualDataNodes: ds_${0..1}.bmsql_district
    databaseStrategy:
     standard:
      shardingColumn: d_w_id
      shardingAlgorithmName: mod_2

   bmsql_customer:
    actualDataNodes: ds_${0..1}.bmsql_customer
    databaseStrategy:
     standard:
      shardingColumn: c_w_id
      shardingAlgorithmName: mod_2

   bmsql_item:
    actualDataNodes: ds_${0..1}.bmsql_item
    databaseStrategy:
     standard:
      shardingColumn: i_id
      shardingAlgorithmName: mod_2

   bmsql_history:
    actualDataNodes: ds_${0..1}.bmsql_history
    databaseStrategy:
     standard:
      shardingColumn: h_w_id
      shardingAlgorithmName: mod_2

   bmsql_oorder:
    actualDataNodes: ds_${0..1}.bmsql_oorder
    databaseStrategy:
     standard:
          shardingColumn: o_w_id
          shardingAlgorithmName: mod_2

    bmsql_stock:
      actualDataNodes: ds_${0..1}.bmsql_stock
      databaseStrategy:
        standard:
          shardingColumn: s_w_id
          shardingAlgorithmName: mod_2

    bmsql_new_order:
      actualDataNodes: ds_${0..1}.bmsql_new_order
      databaseStrategy:
        standard:
          shardingColumn: no_w_id
          shardingAlgorithmName: mod_2

    bmsql_order_line:
      actualDataNodes: ds_${0..1}.bmsql_order_line
      databaseStrategy:
        standard:
          shardingColumn: ol_w_id
          shardingAlgorithmName: mod_2

  shardingAlgorithms:
    mod_2:
      type: MOD
      props:
        sharding-count: 2
```

4. Obtain data.

Initialize the database, create tables through Proxy and insert data.

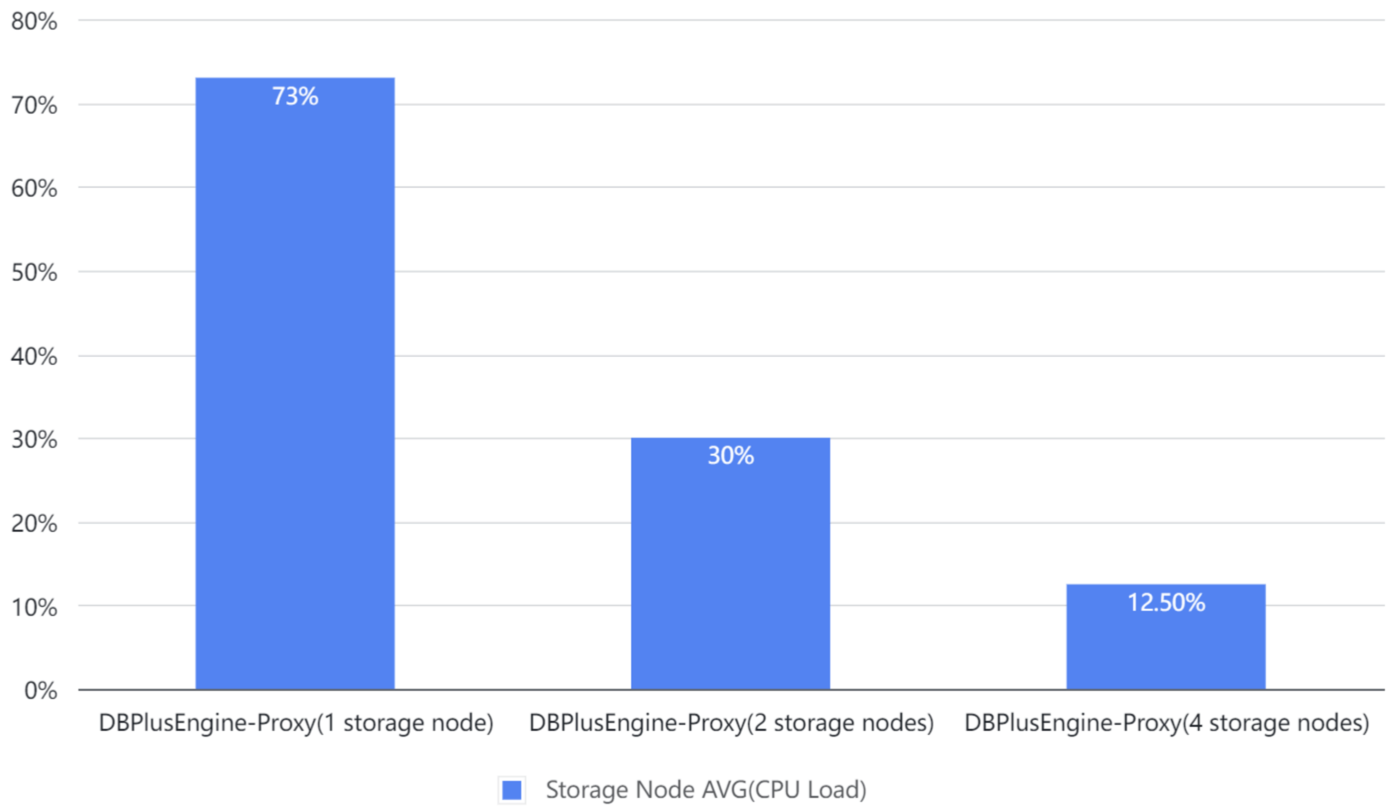
```
cd /usr/local/benchmarksql/run

./runDatabaseDestroy.sh props.proxy

./runDatabaseBuild.sh props.proxy
```

5. Carry out stress testing.

```
./runBenchmark.sh props.proxy
```

## Test Result

| Test Object | Storage Node AVG(CPU Load) | tpmC |
|---|---|---|
| DBPlusEngine-Proxy(1 storage node) | 73% | 122,299 |
| DBPlusEngine-Proxy(2 storage nodes) | 30% | 122,580 |
| DBPlusEngine-Proxy(4 storage nodes) | 12.5% | 128,035 |



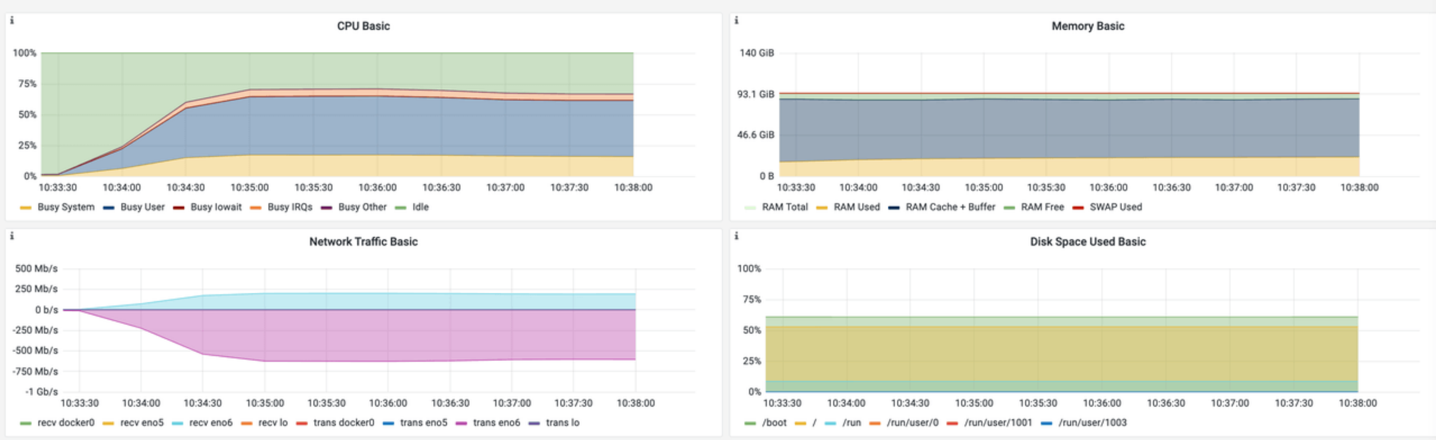
Result: with one Proxy node plus one storage node, it can be seen that the performance bottleneck is located on the CPU of the Proxy server. Therefore, scaling out the storage node only witnessed a slight improvement in performance, and its advantage lies in the decrease of the CPU load of each storage node.

The result is for reference only. Users can carry out a test based on actual business scenarios.

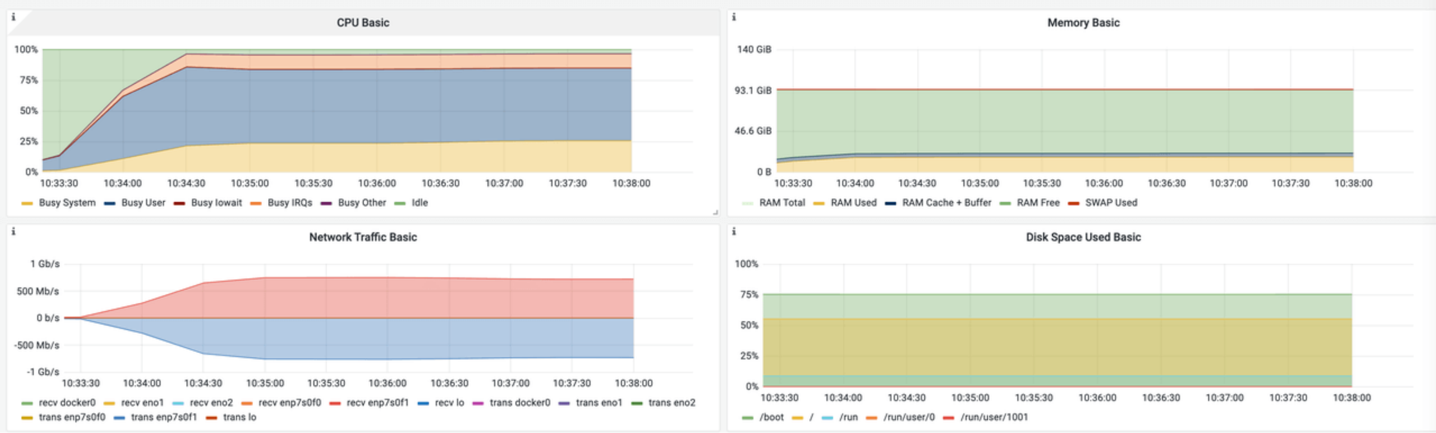
### Monitoring Information

#### BenchmarkSQL ——> Proxy ——> MySQL(1)

- MySQL

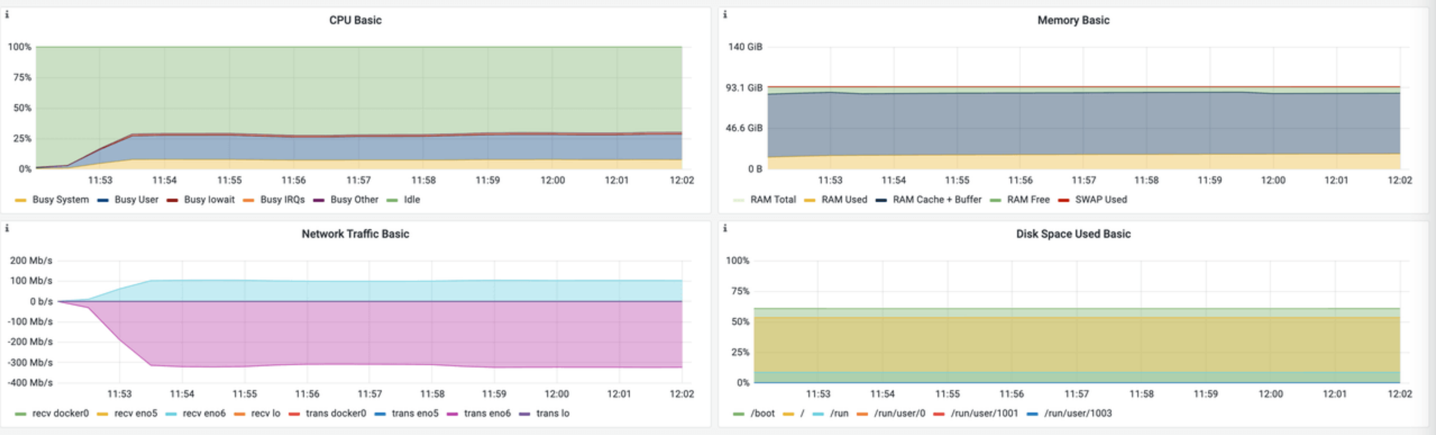


- Proxy

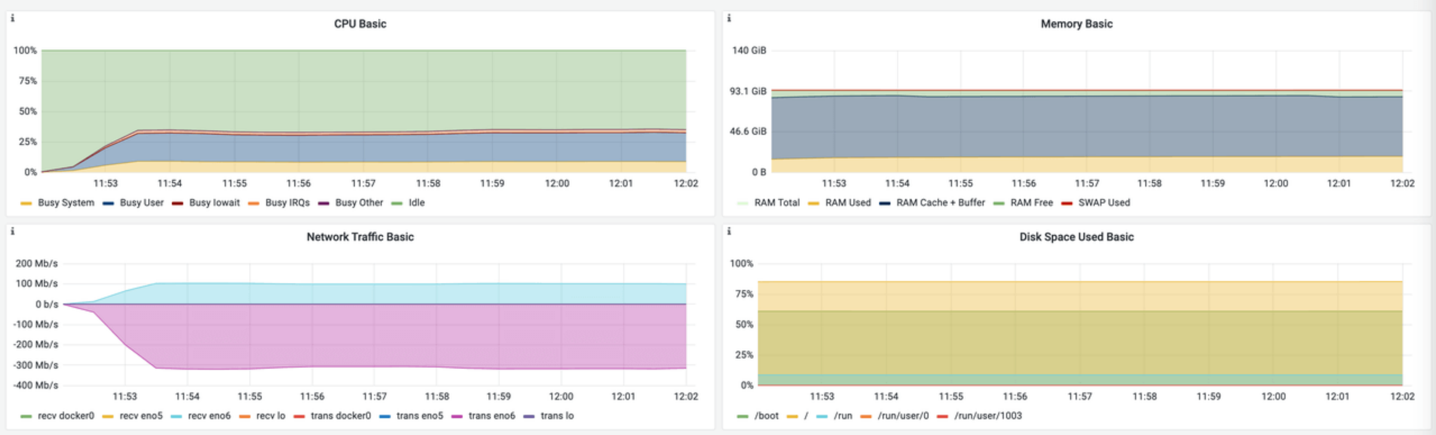


## BenchmarkSQL ——> Proxy ——> MySQL(2)

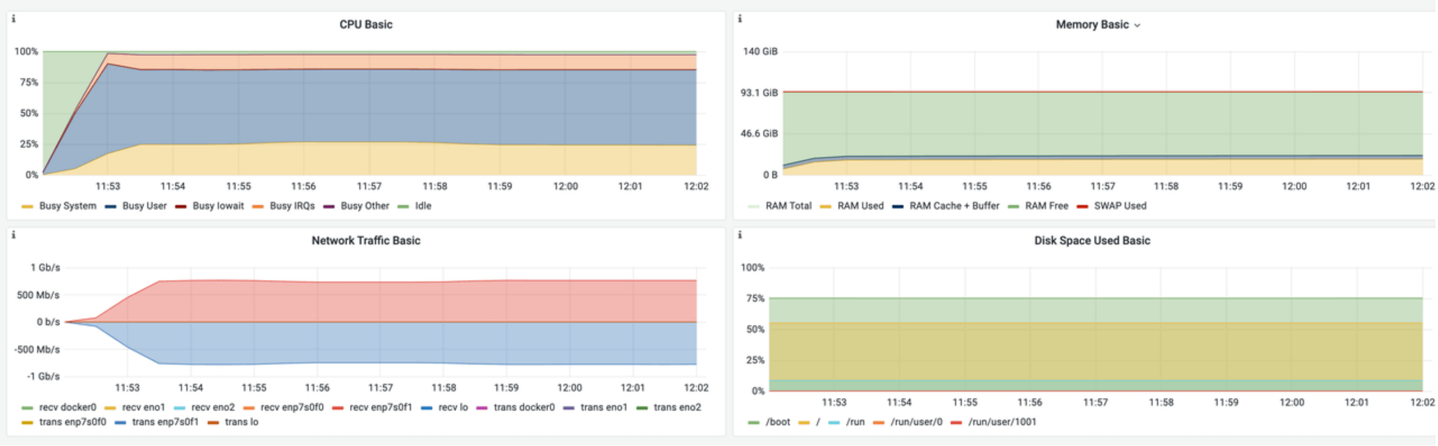
- MySQL(1)



- MySQL(2)

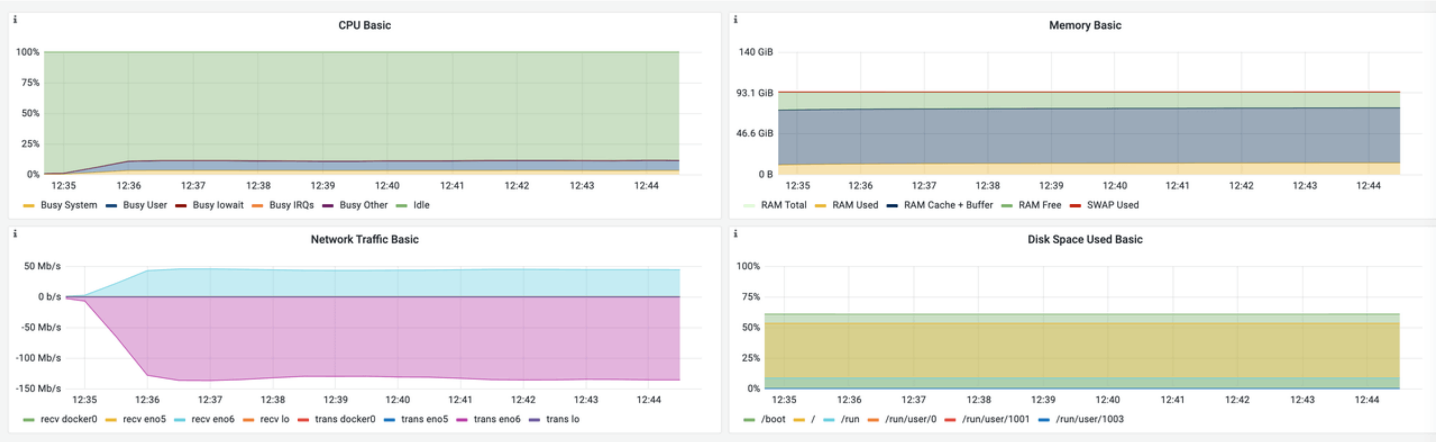


- Proxy

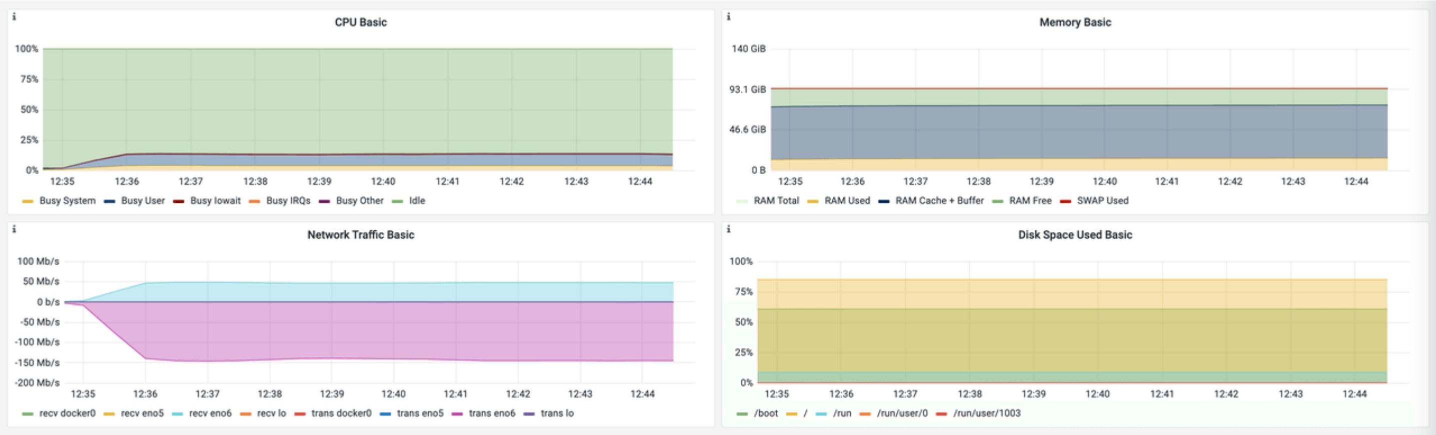


## BenchmarkSQL ——> Proxy ——> MySQL(4)

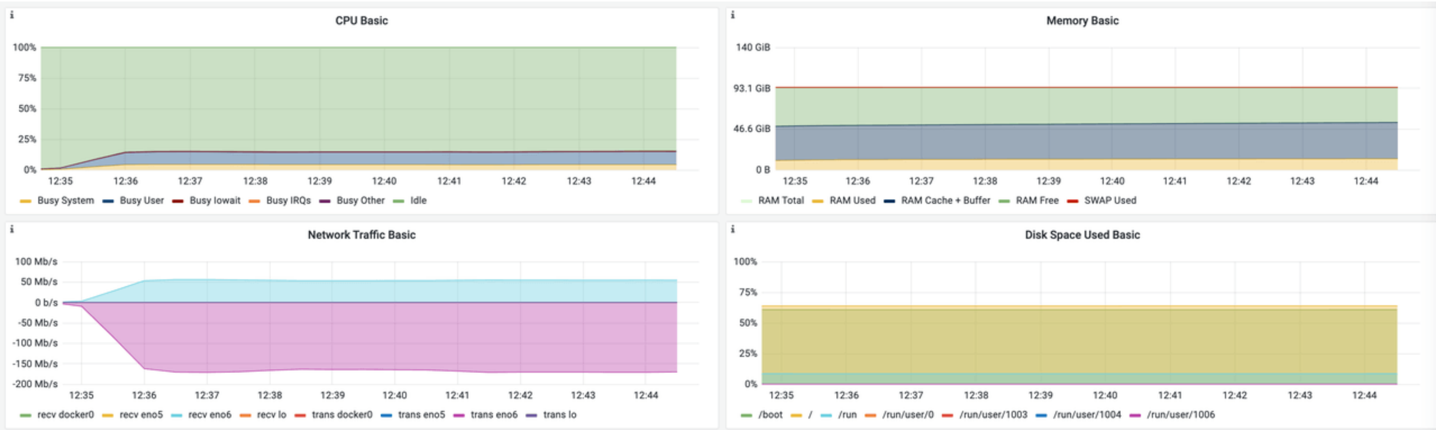
- MySQL(1)



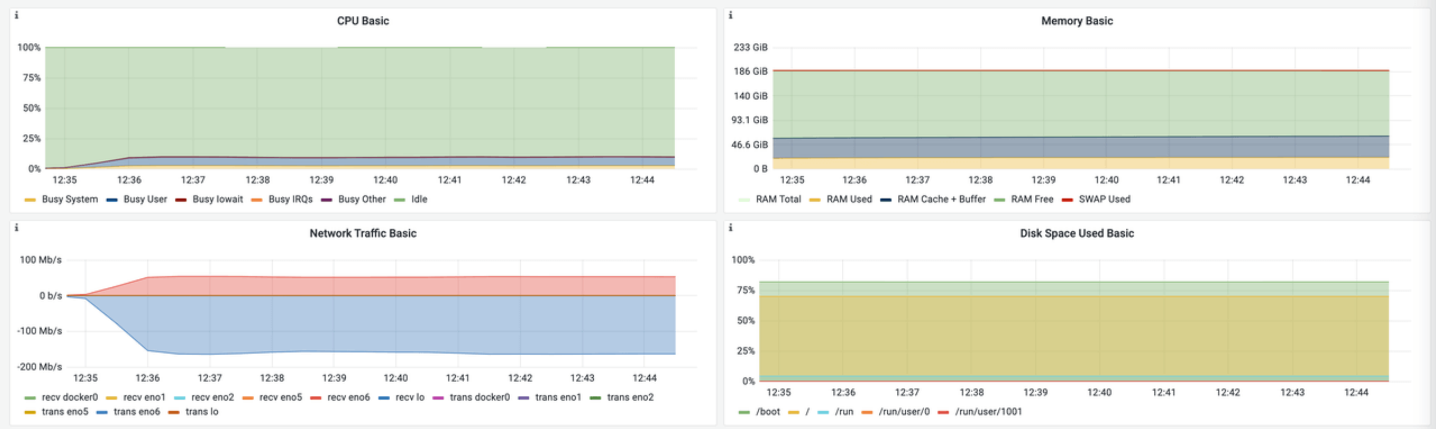
- MySQL(2)



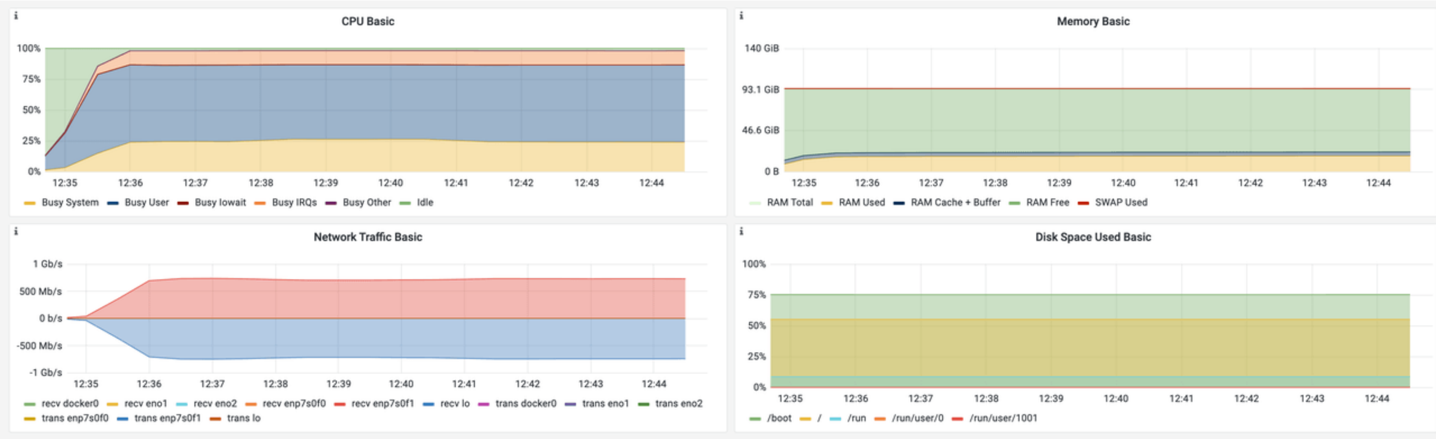
- MySQL(3)



- MySQL(4)



- Proxy



## Glossary

Warehouses: the number of warehouses, which determines the amount of data.

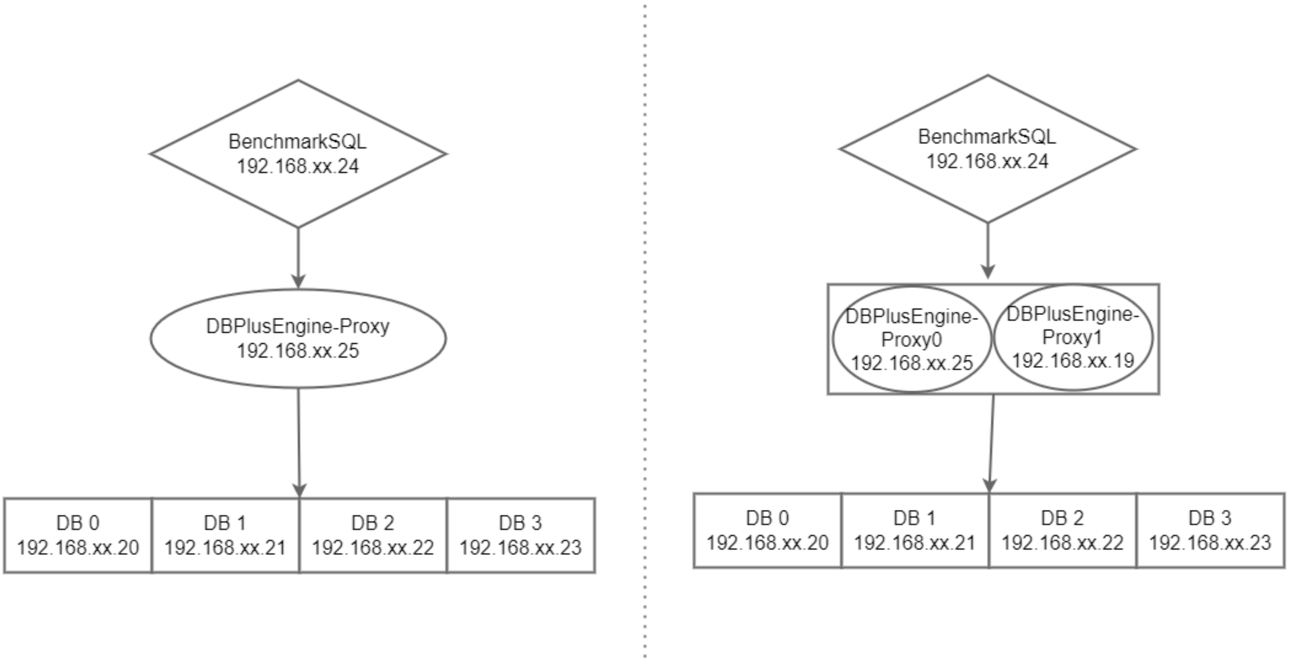
Terminals: the number of threads.

runMins: test duration.

tpmC: transactions per min.

### 5.1.3 Test Plan C: Scale-out the computing node.

Use BenchmarkSQL tools to carry out stress testing on four storage nodes with one computing node and two computing nodes respectively.



### Test Objective

With the same amount of data, the BenchmarkSQL stress testing model is used to compare the performance of DBPlusEngine-Proxy under four storage nodes with one or two computing nodes. By analyzing the results, it' s easier for users to understand the advantages of computing node scale-out.

## Test Tool

BenchmarkSQL

## Test Environment

| Application | IP Address | Port | Version |
|---|---|---|---|
| DBPlusEngine-Proxy | 192.168.xx.25 | 3307 | 1.2 |
| DBPlusEngine-Proxy | 192.168.xx.19 | 3307 | 1.2 |
| MySQL | 192.168.xx.20 | 13306 | 8.0.29 |
| MySQL | 192.168.xx.21 | 13306 | 8.0.29 |
| MySQL | 192.168.xx.22 | 13306 | 8.0.29 |
| MySQL | 192.168.xx.23 | 13306 | 8.0.29 |

Server Configuration

| Item | Configuration |
|---|---|
| CPU | 48 C |
| Memory | 96 G |
| Hard Disk | SSD 820 G |
| JDK | 17.0.2 |

## Test Procedure

1. Create a test table.

```
mysql -utest -h192.168.xx.20 -P13306 -p
```

```
DROP DATABASE IF EXISTS test_tpcc;

CREATE DATABASE IF NOT EXISTS test_tpcc;
```

```
mysql -utest -h192.168.xx.21 -P13306 -p
```

```
DROP DATABASE IF EXISTS test_tpcc;

CREATE DATABASE IF NOT EXISTS test_tpcc;
```

```
mysql -utest -h192.168.xx.22 -P13306 -p
```

```
DROP DATABASE IF EXISTS test_tpcc;

CREATE DATABASE IF NOT EXISTS test_tpcc;
```

```
mysql -utest -h192.168.xx.23 -P13306 -p
```

```
DROP DATABASE IF EXISTS test_tpcc;

CREATE DATABASE IF NOT EXISTS test_tpcc;
```

2. Install test tools.

```
wget https://udomain.dl.sourceforge.net/project/benchmarksql/benchmarksql-5.0.zip
yum -y install ant
unzip benchmarksql-5.0.zip -d  /usr/local
```

```
cd /usr/local/benchmarksql-5.0
ant
cp mysql-connector-java-8.0.24.jar benchmarksql-5.0/run/lib
```

Create props.proxy file.

```
db=postgres
driver=com.mysql.jdbc.Driver
conn=jdbc:mysql://192.168.xx.25:3307/sharding_db?useSSL=false&useServerPrepStmts=true&cachePrepStmts=true&
prepStmtCacheSize=8192&prepStmtCacheSqlLimit=8000
user=root
password=root
warehouses=200
loadWorkers=80
terminals=200
runTxnsPerTerminal=0
runMins=10
limitTxnsPerMin=0
terminalWarehouseFixed=true
newOrderWeight=45
paymentWeight=43
orderStatusWeight=4
deliveryWeight=4
stockLevelWeight=4
resultDirectory=my_result_%tY-%tm-%td_%tH%tM%tS
shardingNumber=4
connBalance=192.168.xx.25:3333,192.168.xx.19:3333
```

3. YAML

```
rules:
 - !SHARDING
  bindingTables:
   - bmsql_warehouse, bmsql_customer
   - bmsql_stock, bmsql_district, bmsql_order_line
  defaultDatabaseStrategy:
   none:
  defaultTableStrategy:
   none:
  keyGenerators:
   snowflake:
    type: SNOWFLAKE
  tables:
   bmsql_config:
    actualDataNodes: ds_0.bmsql_config

   bmsql_warehouse:
    actualDataNodes: ds_${0..3}.bmsql_warehouse
    databaseStrategy:
     standard:
      shardingColumn: w_id
      shardingAlgorithmName: mod_4

   bmsql_district:
    actualDataNodes: ds_${0..3}.bmsql_district
    databaseStrategy:
     standard:
      shardingColumn: d_w_id
      shardingAlgorithmName: mod_4

   bmsql_customer:
    actualDataNodes: ds_${0..3}.bmsql_customer
    databaseStrategy:
     standard:
          shardingColumn: c_w_id
          shardingAlgorithmName: mod_4

    bmsql_item:
      actualDataNodes: ds_${0..3}.bmsql_item
      databaseStrategy:
        standard:
          shardingColumn: i_id
          shardingAlgorithmName: mod_4

    bmsql_history:
      actualDataNodes: ds_${0..3}.bmsql_history
      databaseStrategy:
        standard:
          shardingColumn: h_w_id
          shardingAlgorithmName: mod_4

    bmsql_oorder:
      actualDataNodes: ds_${0..3}.bmsql_oorder
      databaseStrategy:
        standard:
          shardingColumn: o_w_id
          shardingAlgorithmName: mod_4

    bmsql_stock:
      actualDataNodes: ds_${0..3}.bmsql_stock
      databaseStrategy:
        standard:
          shardingColumn: s_w_id
          shardingAlgorithmName: mod_4

    bmsql_new_order:
      actualDataNodes: ds_${0..3}.bmsql_new_order
      databaseStrategy:
        standard:
          shardingColumn: no_w_id
          shardingAlgorithmName: mod_4

    bmsql_order_line:
      actualDataNodes: ds_${0..3}.bmsql_order_line
      databaseStrategy:
        standard:
          shardingColumn: ol_w_id
          shardingAlgorithmName: mod_4

  shardingAlgorithms:
    mod_4:
      type: MOD
      props:
        sharding-count: 4
```

4. Obtain data.

Initialize the database, create tables through Proxy and insert data.

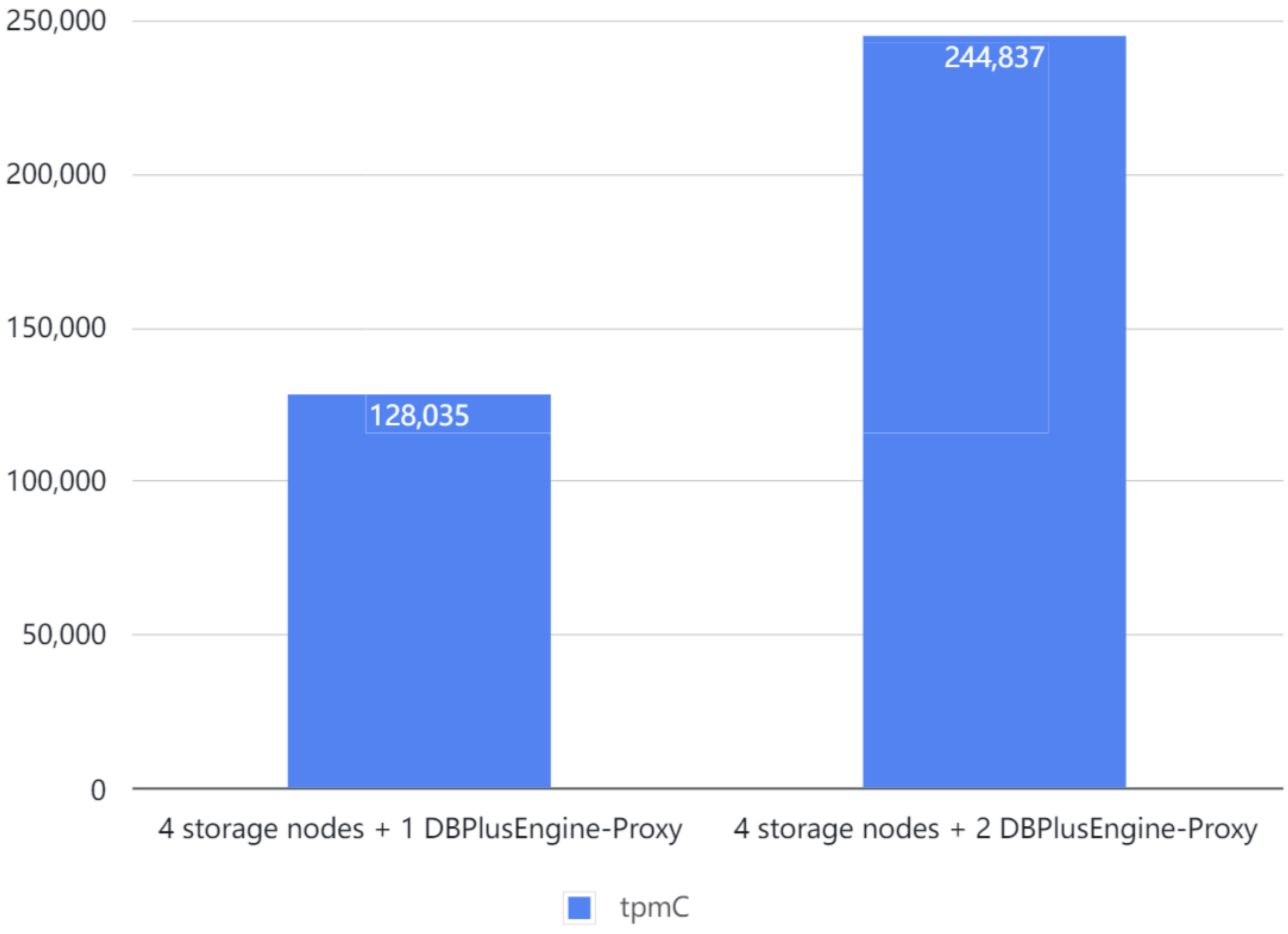
```
cd /usr/local/benchmarksql/run

./runDatabaseDestroy.sh props.proxy

./runDatabaseBuild.sh props.proxy
```

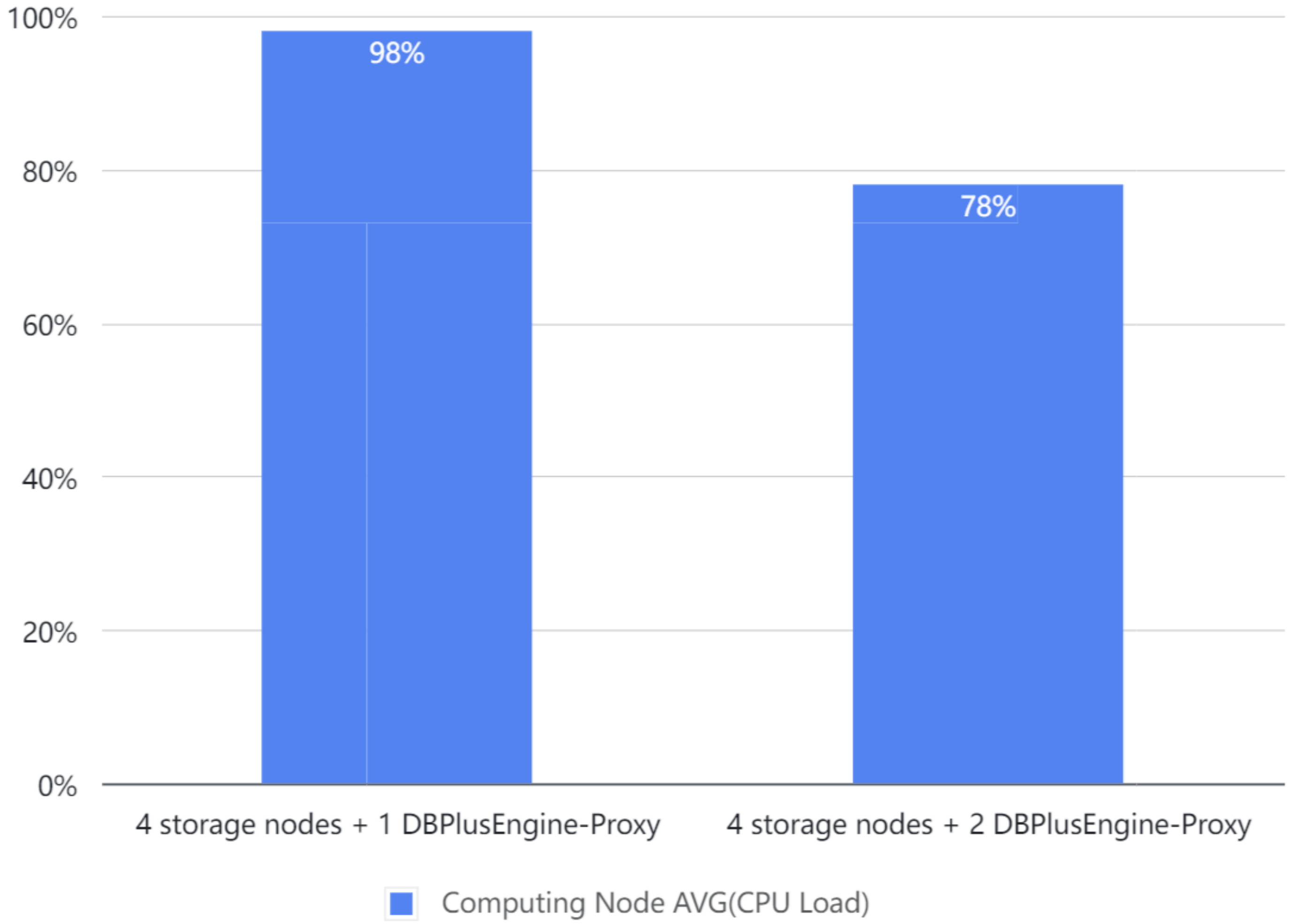
5. Carry out stress testing.

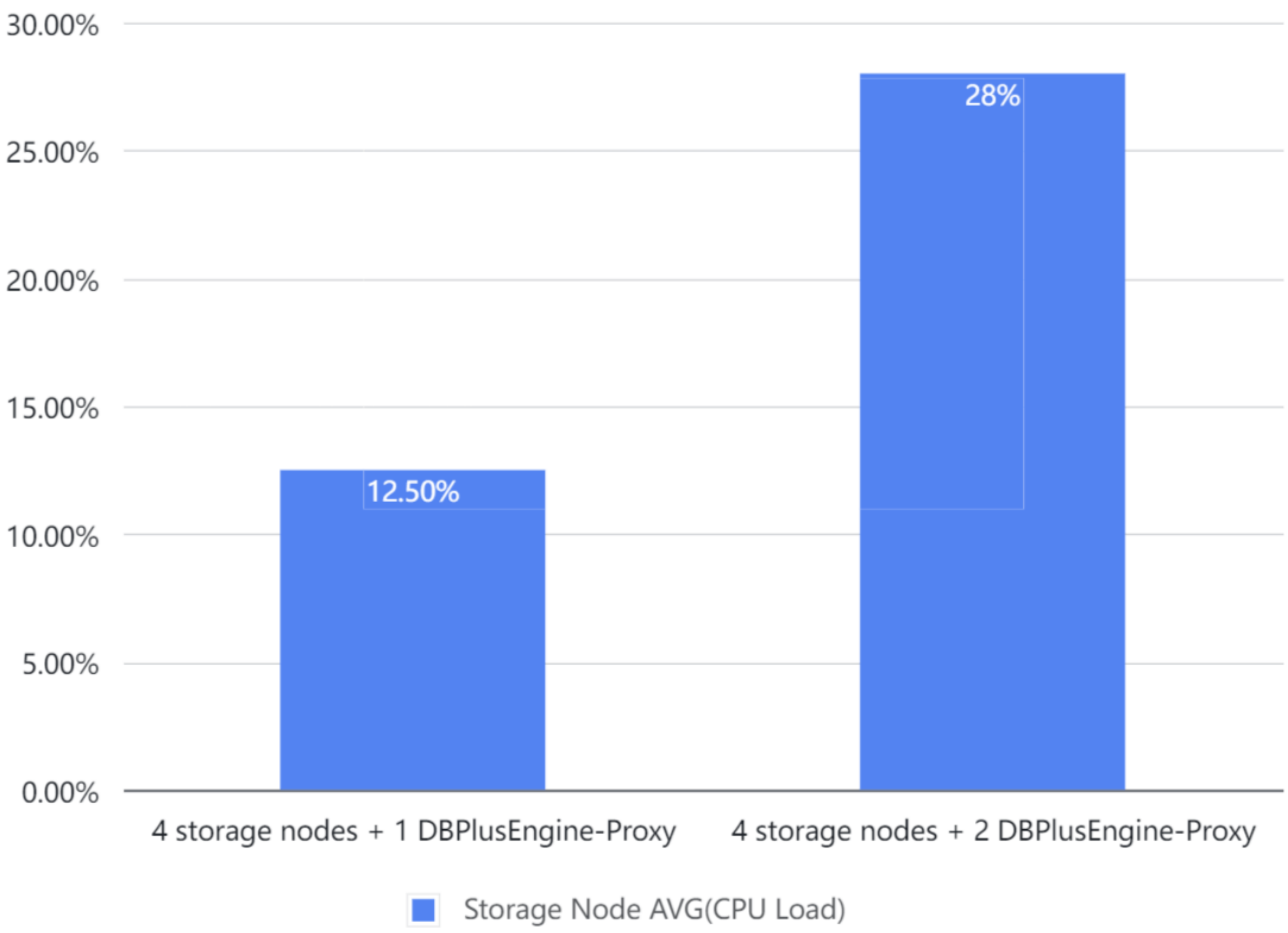
```
./runBenchmark.sh props.proxy
```

### Test Result

| Test Object | tpmC | Storage Node AVG(CPU Load) | Computing Node AVG(CPU Load) |
|---|---|---|---|
| 4 storage nodes + 1 DB PlusEngine-Proxy | 128,035 | 12.5% | 98% |
| 4 storage nodes + 2 DB PlusEngine-Proxy | 244,837 | 28% | 78% |




100%
80%
60%
40%
20%
0%
98%
78%
4 storage nodes + 1 DBPlusEngine-Proxy
4 storage nodes + 2 DBPlusEngine-Proxy
Computing Node AVG(CPU Load)

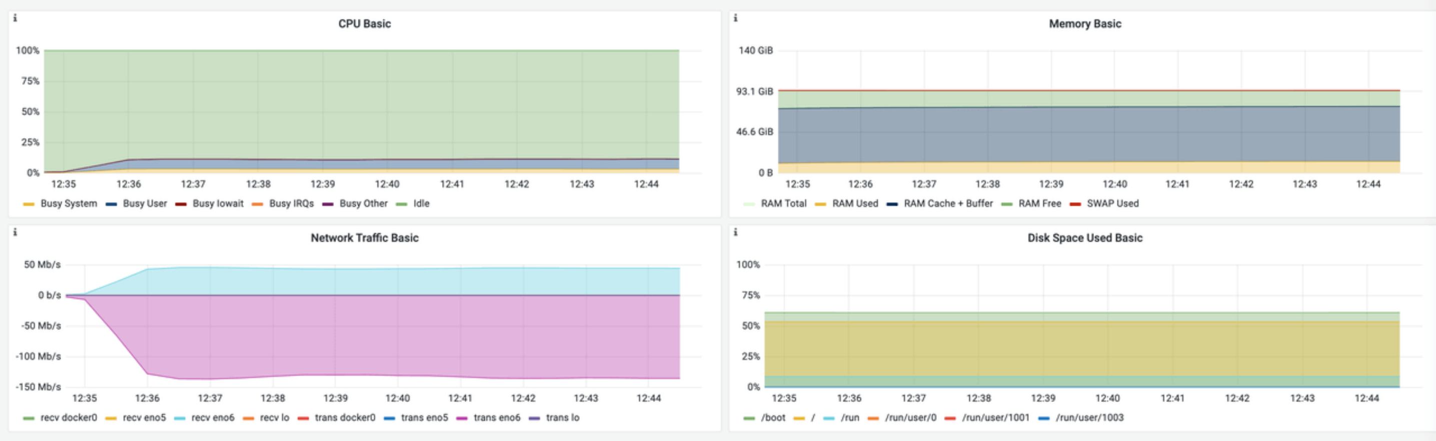


Result: scaling out computing nodes can improve the overall throughput linearly.

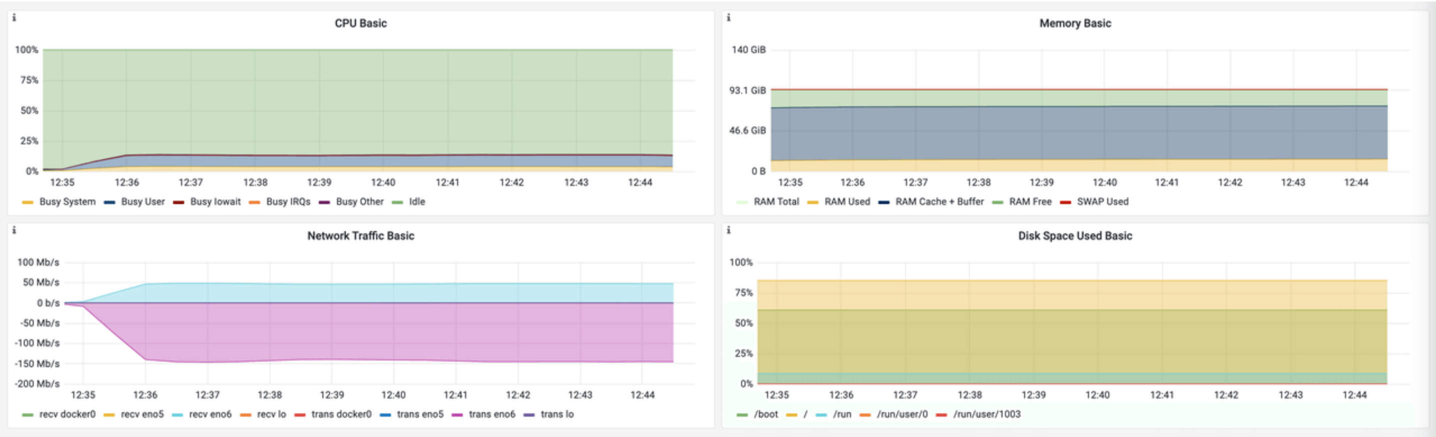
## Monitoring Information

### BenchmarkSQL ——> Proxy ——> MySQL(4)

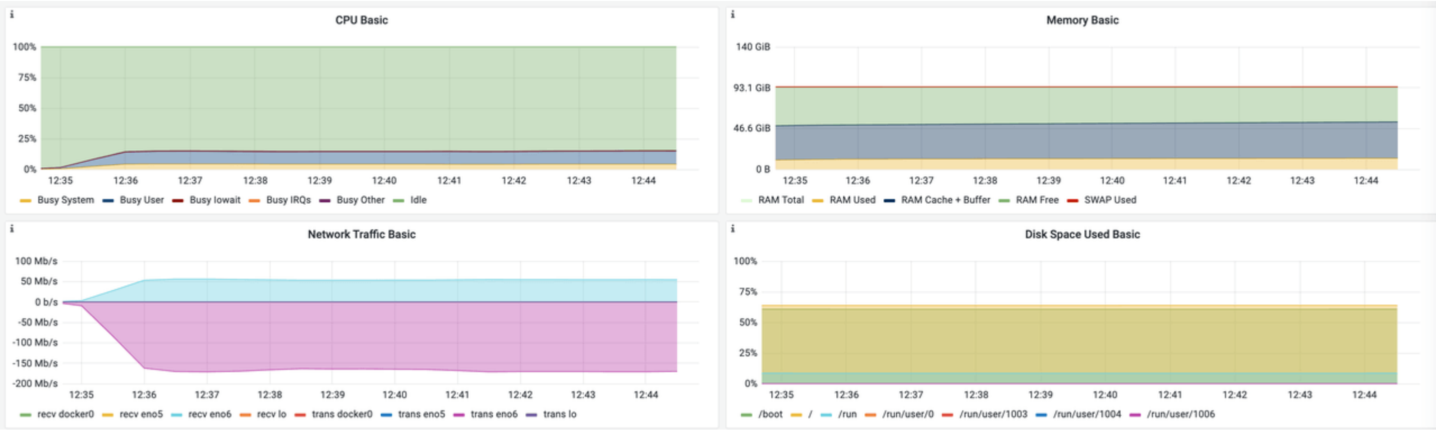
- MySQL(1)



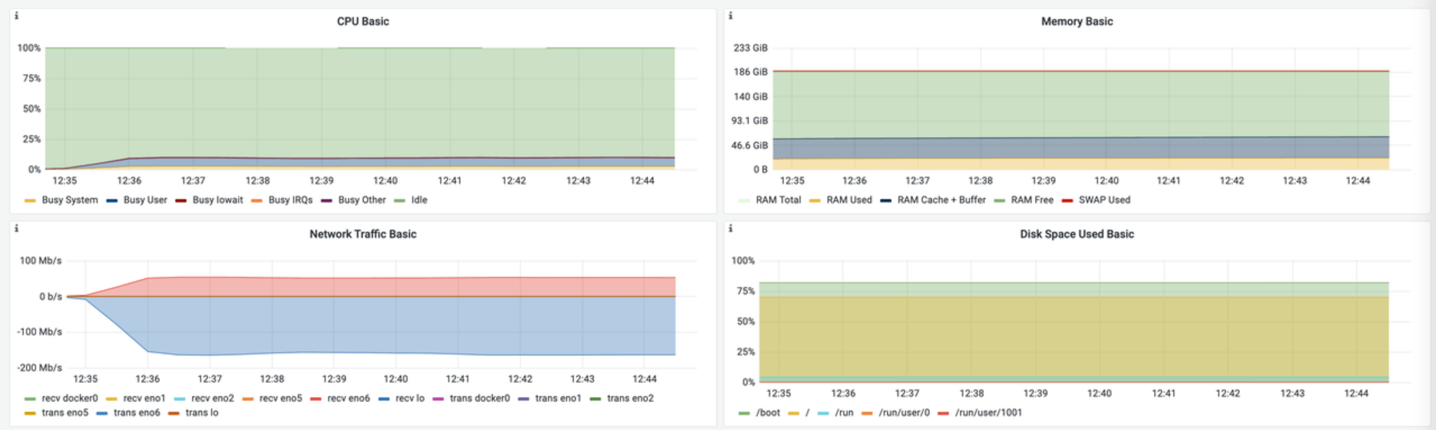
- MySQL(2)



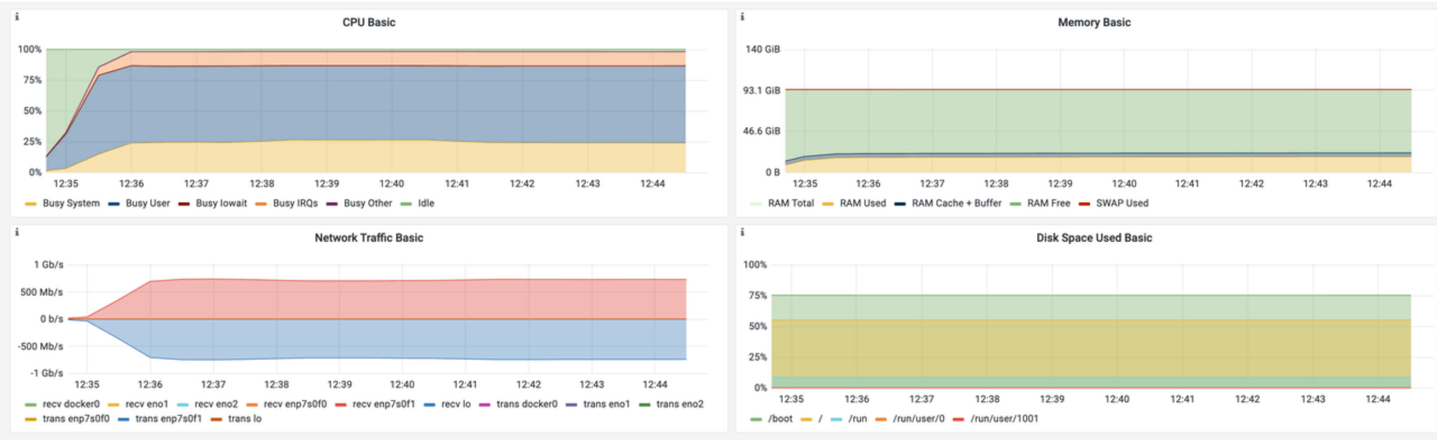
- MySQL(3)



- MySQL(4)

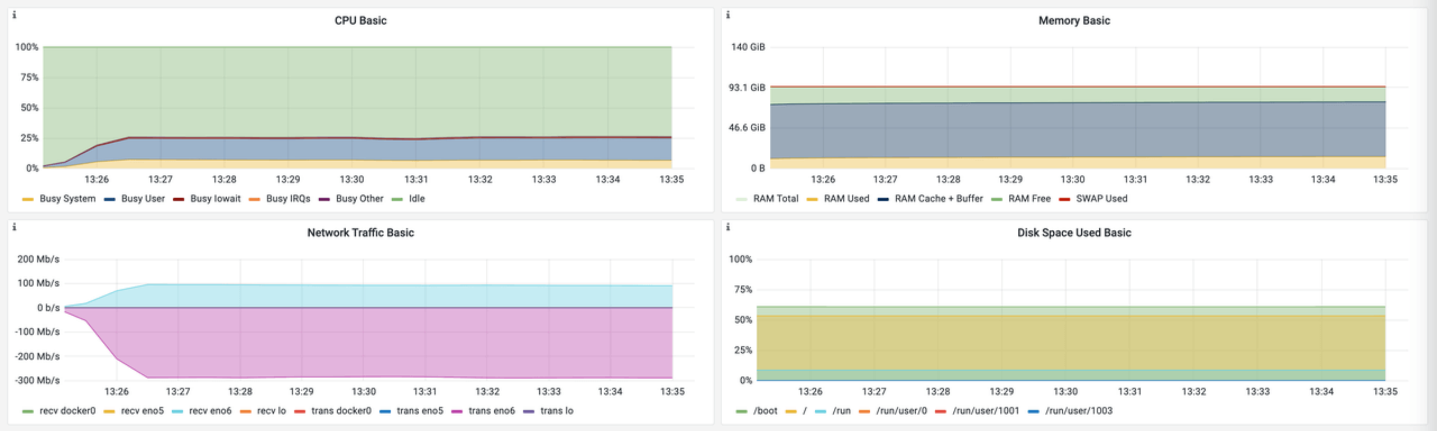


- Proxy

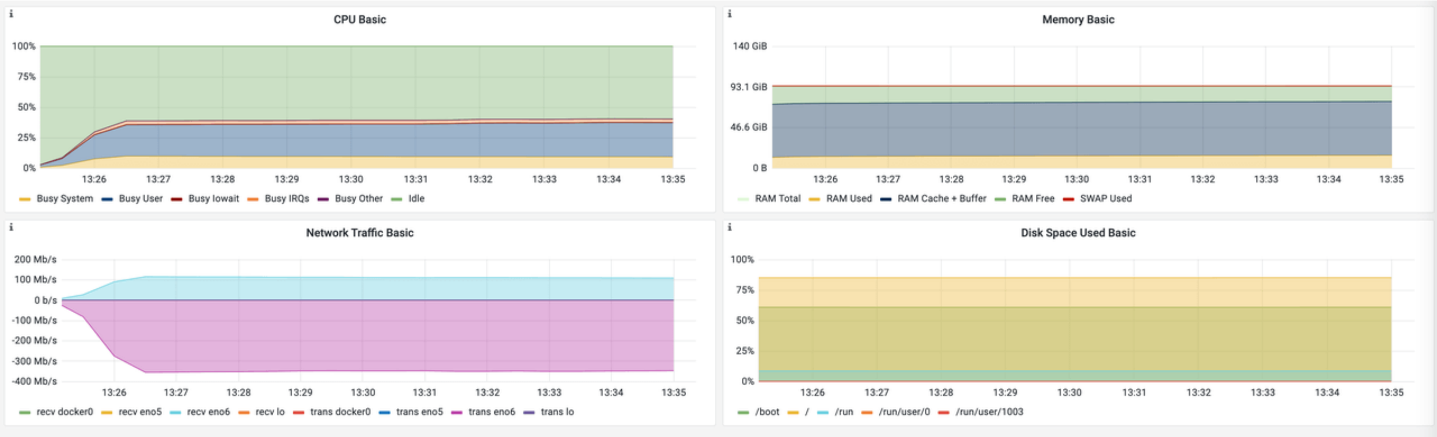


## BenchmarkSQL ——> Proxy(2) ——> MySQL(4)

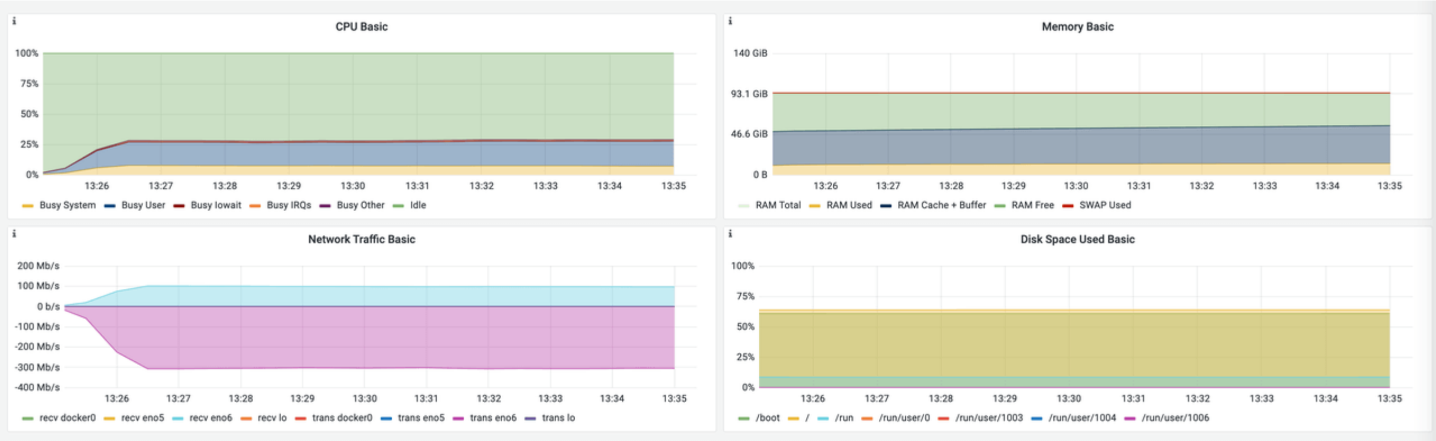
- MySQL(1)



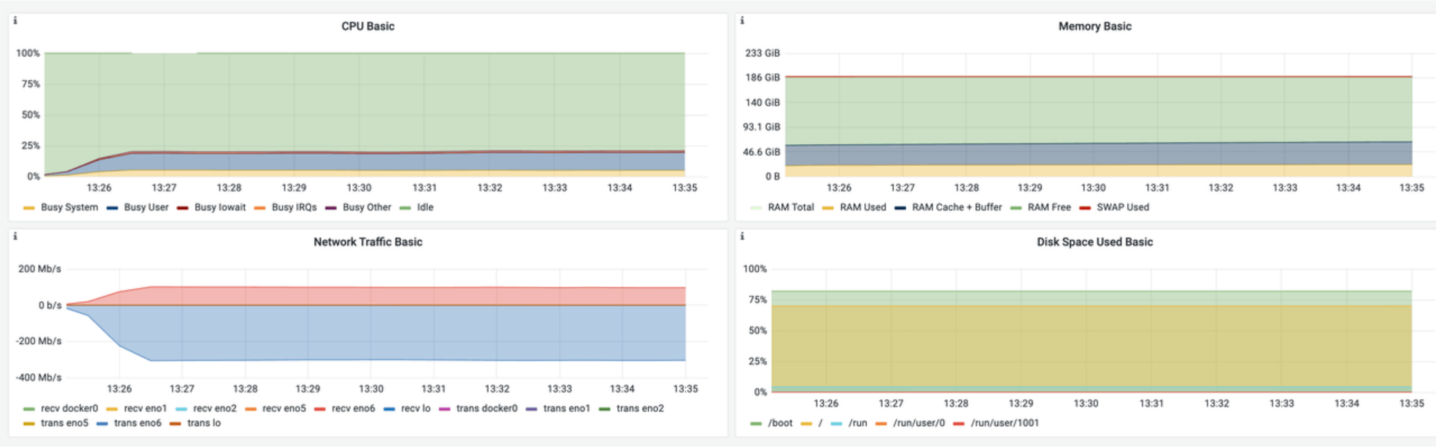
- MySQL(2)



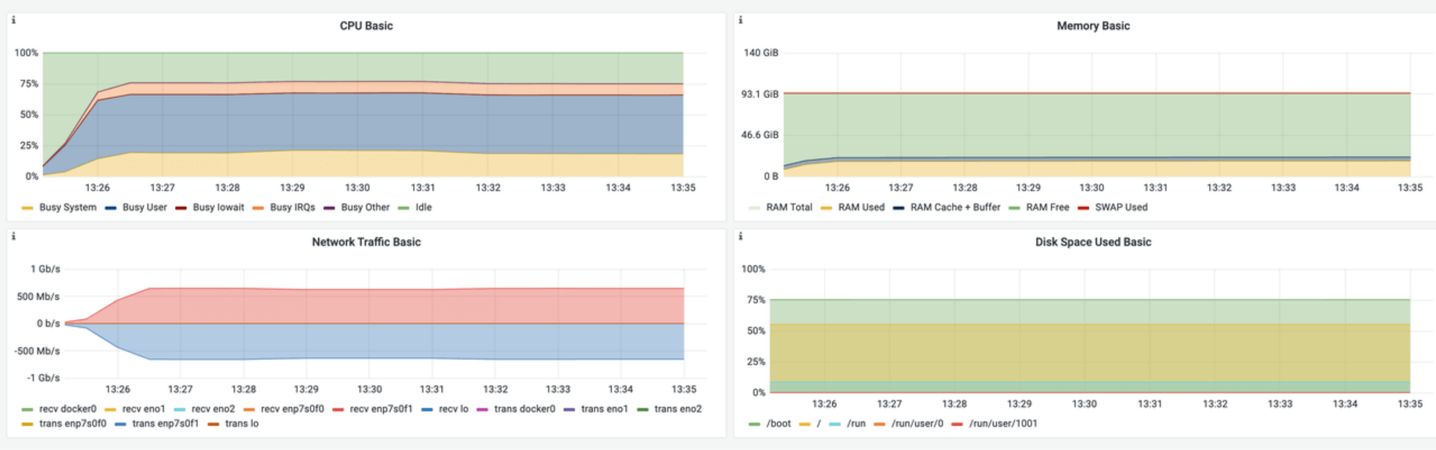
- MySQL(3)



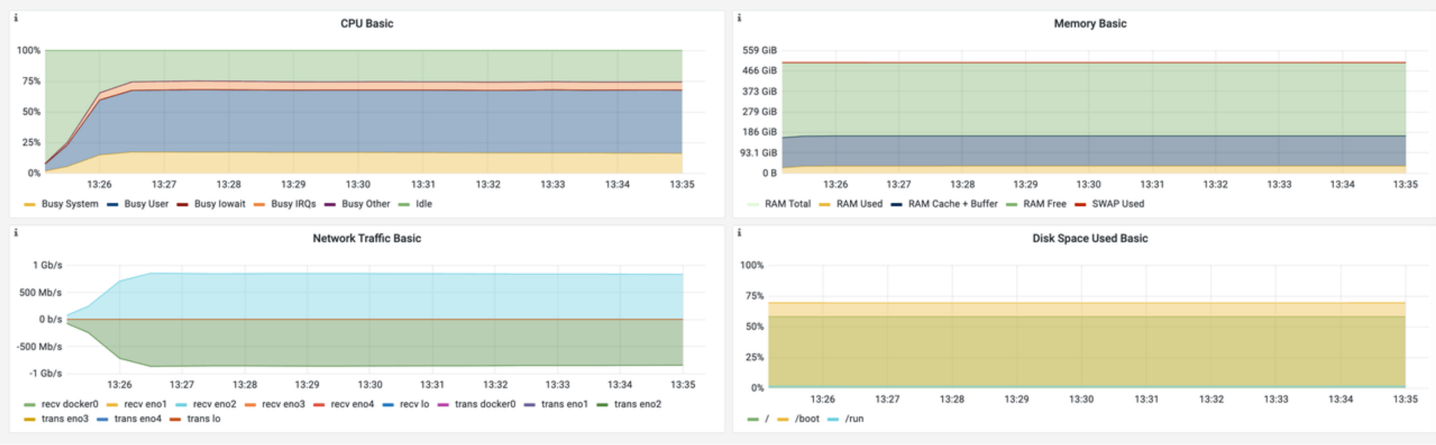
- MySQL(4)



- Proxy(1)



- Proxy(2)



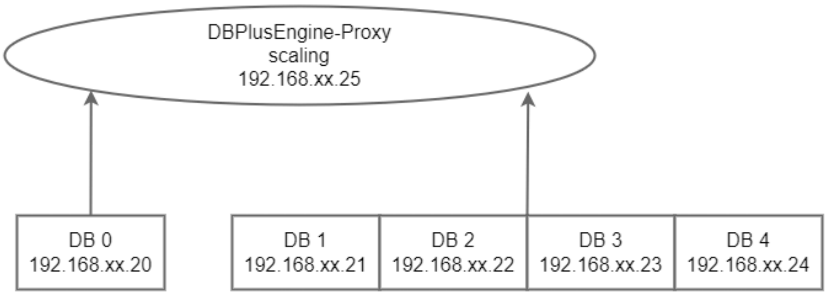
## Glossary

Warehouses: the number of warehouses, which determines the amount of data.
Terminals: the number of threads.
runMins: test duration.
tpmC: transactions per min.

# 5.2 Data Integration Test

## 5.2.1 Test Plan A: Data Export

Data migration. Migrate 36 gigabytes of data (140 million lines) from a single database and table into the target source.



## Test Objective

Once the data reaches a large amount, databases' performance will decrease notably, affecting the read/write operations. We can use DBPlusEngine-Proxy's scaling feature to migrate the original data to multiple target databases without affecting the business operations.

## Test Tool

Sysbench

## Test Environment

| Application | IP Address | Port |
|---|---|---|
| DBPlusEngine-Proxy | 192.168.xx.25 | 3307 |
| MySQL | 192.168.xx.20 | 13306 |
| MySQL | 192.168.xx.21 | 13306 |
| MySQL | 192.168.xx.22 | 13306 |
| MySQL | 192.168.xx.23 | 13306 |
| MySQL | 192.168.xx.24 | 13306 |

Server Configuration

| Item | Configuration |
|---|---|
| CPU | 48 C |
| Memory | 96 G |
| Hard Disk | SSD 820 G |
| JDK | 17.0.2 |

### Test Procedure

1. Create a test table.

```
mysql -utest -h192.168.xx.20 -P13306 -p
```

```
DROP DATABASE IF EXISTS migration_ds_0;

CREATE DATABASE IF NOT EXISTS migration_ds_0;
```

```
mysql -utest -h192.168.xx.21 -P13306 -p
```

```
DROP DATABASE IF EXISTS migration_ds_1;

CREATE DATABASE IF NOT EXISTS migration_ds_1;
```

```
mysql -utest -h192.168.xx.22 -P13306 -p
```

```
DROP DATABASE IF EXISTS migration_ds_2;

CREATE DATABASE IF NOT EXISTS migration_ds_2;
```

```
mysql -utest -h192.168.xx.23 -P13306 -p
```

```
DROP DATABASE IF EXISTS migration_ds_3;

CREATE DATABASE IF NOT EXISTS migration_ds_3;
```

```
mysql -utest -h192.168.xx.24 -P13306 -p
```

```
DROP DATABASE IF EXISTS migration_ds_4;

CREATE DATABASE IF NOT EXISTS migration_ds_4;
```

```
mysql -uroot -h192.168.xx.25 -P3307 -proot
```

```
DROP DATABASE IF EXISTS sphereex_demo;

CREATE DATABASE IF NOT EXISTS sphereex_demo;
```

2. Install test tools.

```
tar -xzvf sysbench-1.0.20.tar.gz -C /usr/local
cd /usr/local/sysbench-1.0.20
./autogen.sh
./configure
make && make install
```

3. Obtain data.

Use the Sysbench tool to insert 140 million pieces of data into the source database.

```
sysbench oltp_read_only --mysql-host='192.168.xx.20' --mysql-port=13306 --mysql-user=test --mysql-password='test' --mysql-db=migration_ds_0 --tables=1 --table-size=140000000 --report-interval=10 --time=3600 --threads=256 --max-requests=0 --percentile=99 --mysql-ignore-errors="all" --rand-type=uniform --range_selects=off --auto_inc=off cleanup

sysbench oltp_read_only --mysql-host='192.168.xx.20' --mysql-port=13306 --mysql-user=test --mysql-password='test' --mysql-db=migration_ds_0 --tables=1 --table-size=140000000 --report-interval=10 --time=3600 --threads=256 --max-requests=0 --percentile=99 --mysql-ignore-errors="all" --rand-type=uniform --range_selects=off --auto_inc=off prepare
```

4. Migration Procedure.

Create a new logical database and configure resources and rules through Proxy.

```
mysql -uroot -h 192.168.xx.25 -P3307 -proot
```

```
CREATE DATABASE sphereex_demo;

USE sphereex_demo;

REGISTER STORAGE UNIT ds_0 (
   URL="jdbc:mysql://192.168.xx.20:13306/migration_ds_0?serverTimezone=UTC&useSSL=false",
   USER="test",
   PASSWORD="test",
   PROPERTIES("maximumPoolSize"=500,"idleTimeout"=60000)
);
```

Add target sources.

```
REGISTER STORAGE UNIT ds_1 (
   URL="jdbc:mysql://192.168.xx.21:13306/migration_ds_1?serverTimezone=UTC&useSSL=false",
   USER="test",
   PASSWORD="test",
   PROPERTIES("maximumPoolSize"=50,"idleTimeout"="60000")
), ds_2 (
   URL="jdbc:mysql://192.168.xx.22:13306/migration_ds_2?serverTimezone=UTC&useSSL=false",
   USER="test",
   PASSWORD="test",
   PROPERTIES("maximumPoolSize"=50,"idleTimeout"="60000")
), ds_3 (
   URL="jdbc:mysql://192.168.xx.23:13306/migration_ds_3?serverTimezone=UTC&useSSL=false",
   USER="test",
   PASSWORD="test",
   PROPERTIES("maximumPoolSize"=50,"idleTimeout"="60000")
), ds_4 (
   URL="jdbc:mysql://192.168.xx.24:13306/migration_ds_4?serverTimezone=UTC&useSSL=false",
   USER="test",
   PASSWORD="test",
   PROPERTIES("maximumPoolSize"=50,"idleTimeout"="60000")
);
```

Create rules.

```
CREATE SHARDING TABLE RULE sbtest1(
STORAGE_UNITS(ds_1,ds_2,ds_3,ds_4),
SHARDING_COLUMN=id,
TYPE(NAME="hash_mod",PROPERTIES("sharding-count"=4)),
KEY_GENERATE_STRATEGY(COLUMN=id,TYPE(NAME="snowflake"))
);
```

Add the migration configuration.

```
ALTER MIGRATION RULE (
READ(
 WORKER_THREAD=40,
 BATCH_SIZE=1000,
 SHARDING_SIZE=7000000
),
WRITE(
 WORKER_THREAD=40,
 BATCH_SIZE=1000
)
);
```

Configure source resources.

```
REGISTER MIGRATION SOURCE STORAGE UNIT ds_0 (
   URL="jdbc:mysql://192.168.xx.20:13306/migration_ds_0?serverTimezone=UTC&useSSL=false",
   USER="test",
   PASSWORD="test",
   PROPERTIES("maximumPoolSize"=50,"idleTimeout"="60000")
);
```

Start executing the task.

```
MIGRATE TABLE ds_0.sbtest1 INTO sbtest1;
```

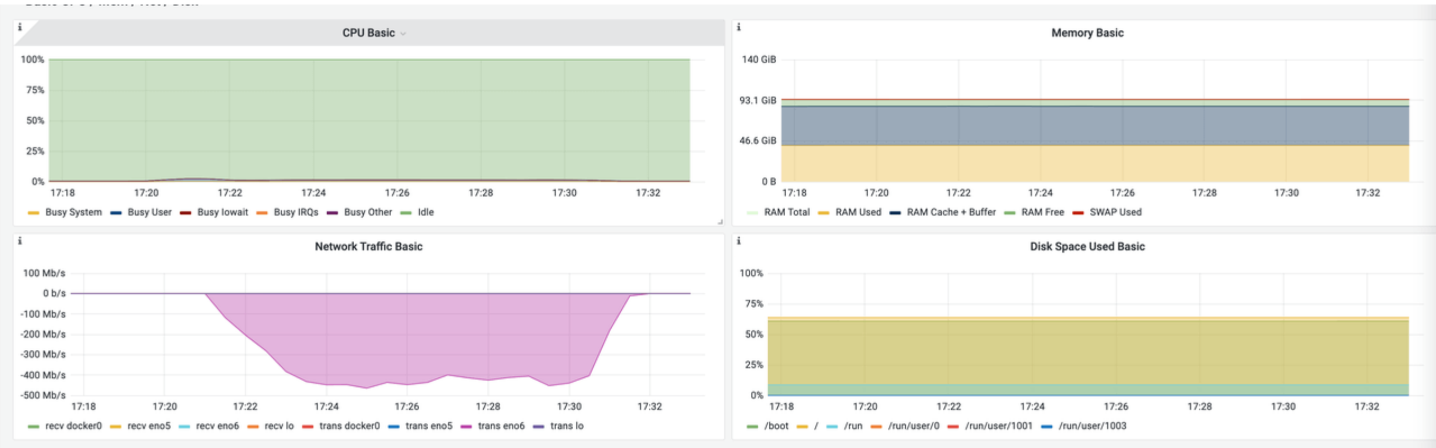
### Test Result

The migration process takes 10 mins 49 s.

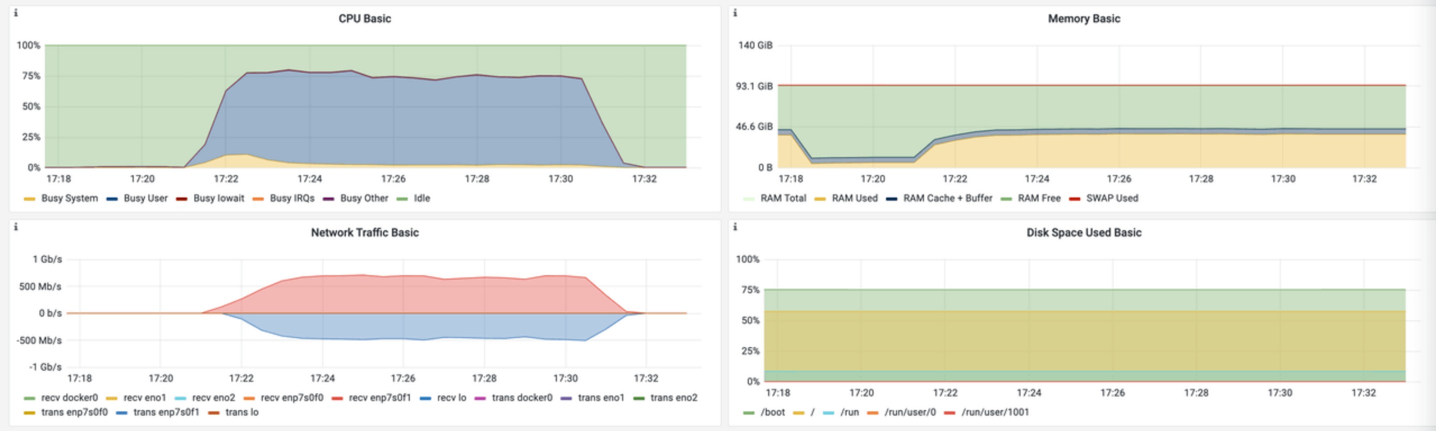
Test result: the computing node reaches the bottleneck first under the above configuration.

### Monitoring Information

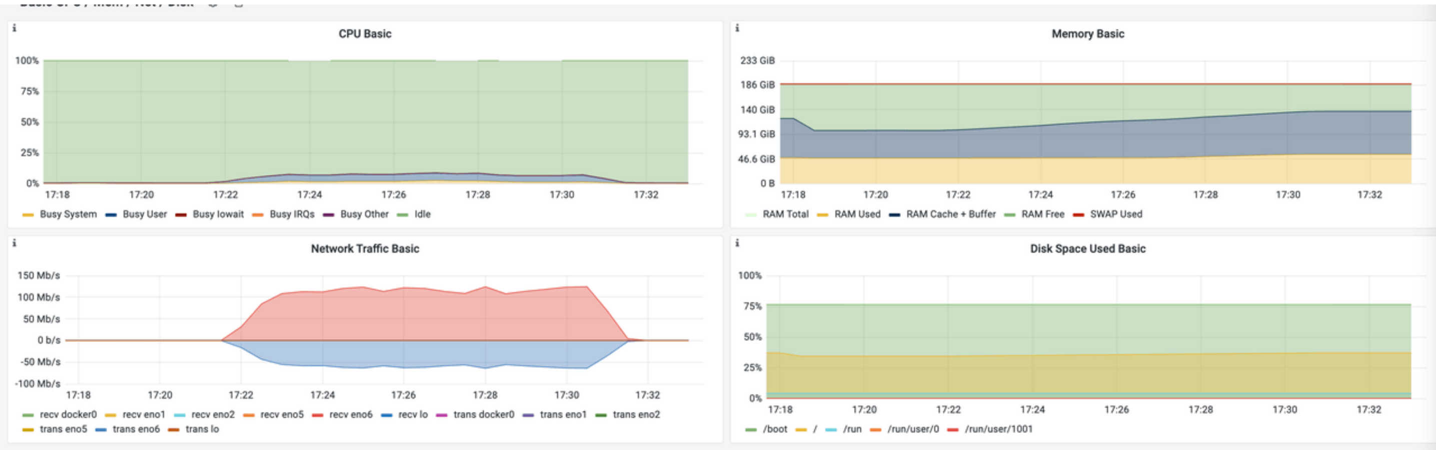
Source DB



DBPlusEngine-Proxy



One of target DB



## Glossary

Id: id of the migration task.

tables: name of the table to be migrated.

sharding_total_count: number of the source tables.

active: indicates whether the migration task is running. If it' s false, the migration task is disabled and will not run again.

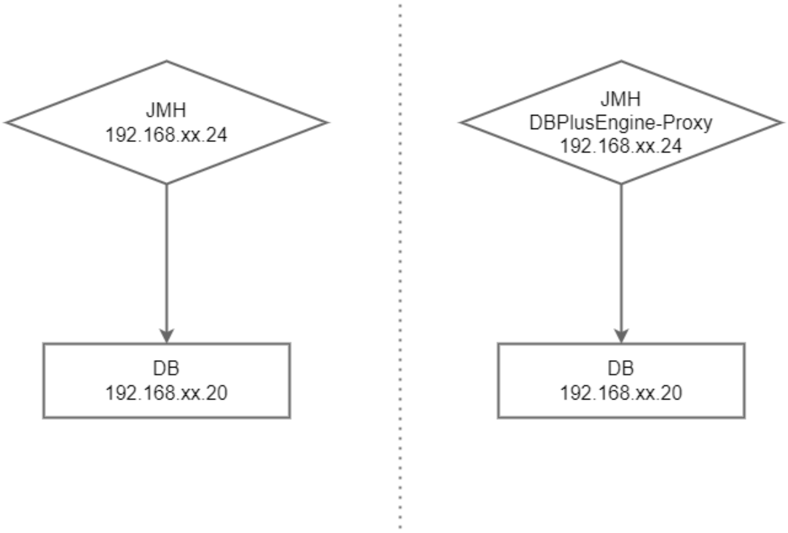
create_time: the start time of the migration task.

stop_time: the end time of the migration task.

### 5.2.2 Test Plan B: Data Import

Use JMH and DBPlusEngine-Driver to insert data into databases and tables. The insertion duration is 5 mins and the amount of data inserted is counted.

Use JMH to insert data into MySQL directly. The insertion duration is 5 mins and the amount of data inserted is counted.



## Test Objective

By comparing the two ways of importing data, we can:

- Work out the performance indicator of DBPlusEngine-Driver.
- Compare this with the performance loss of MySQL native import.

## Test Tool

JMH

## Test Environment

| Application | IP Address | Port | Version |
|---|---|---|---|
| DBPlusEngine-Driver | 192.168.xx.24 | 3307 | 1.2 |
| MySQL | 192.168.xx.20 | 13306 | 8.0.29 |

Server Configuration

| Item | Configuration |
|---|---|
| CPU | 48 C |
| Memory | 96 G |
| Hard Disk | SSD 820 G |
| JDK | 17.0.2 |

Key parameters - –Dtables=1: quantity. - -f 1: executions. - -r 300: execution time. - -t 200: number of threads used. - -w 0: number of warm-ups.

### Test Procedure

1. Create a test database.

```
mysql -utest -h192.168.xx.20 -P13306 -p
```

```
DROP SCHEMA IF EXISTS insert_db;

CREATE TABLE `sbtest1` (
 `id` int unsigned NOT NULL AUTO_INCREMENT,
 `k` int NOT NULL DEFAULT '0',
 `c` char(255) NOT NULL DEFAULT '',
 `pad` char(60) NOT NULL DEFAULT '',
 PRIMARY KEY (`id`),
 KEY `k_1` (`k`)
) ENGINE=InnoDB AUTO_INCREMENT=1;
```

2. Install test tools.

Note: users have to configure JAVA and MAVEN environments in advance. openjdk 17.0.2 and Maven 3.6.3 are used in the following example.

Establish DBPlusEngine-Driver.

Users can write their own test cases based on the JMH framework or compile the test case set using the following codes.

```
git clone https://github.com/SphereEx-QE/database-jmh.git
cd database-jmh/jmh-shardingsphere5
mvn clean package && cd target
```

testssj.yaml configuration:

```
dataSources:
 write_ds:
  dataSourceClassName: com.zaxxer.hikari.HikariDataSource
  driverClassName: com.mysql.jdbc.Driver
  jdbcUrl: jdbc:mysql://192.168.xx.20:13306/insert_db?serverTimezone=UTC&useSSL=false&useUnicode=true&
characterEncoding=UTF-8
  username: root
  password: root
  maximumPoolSize: 300
  minimumIdle: 1

rules: []
props:
```

mysql.properties configuration:

```
jdbc-url=jdbc:mysql://192.168.xx.21:13306/pure_mysql?serverTimezone=UTC&useSSL=false&useUnicode=true&
characterEncoding=UTF-8
username=root
password=root
```

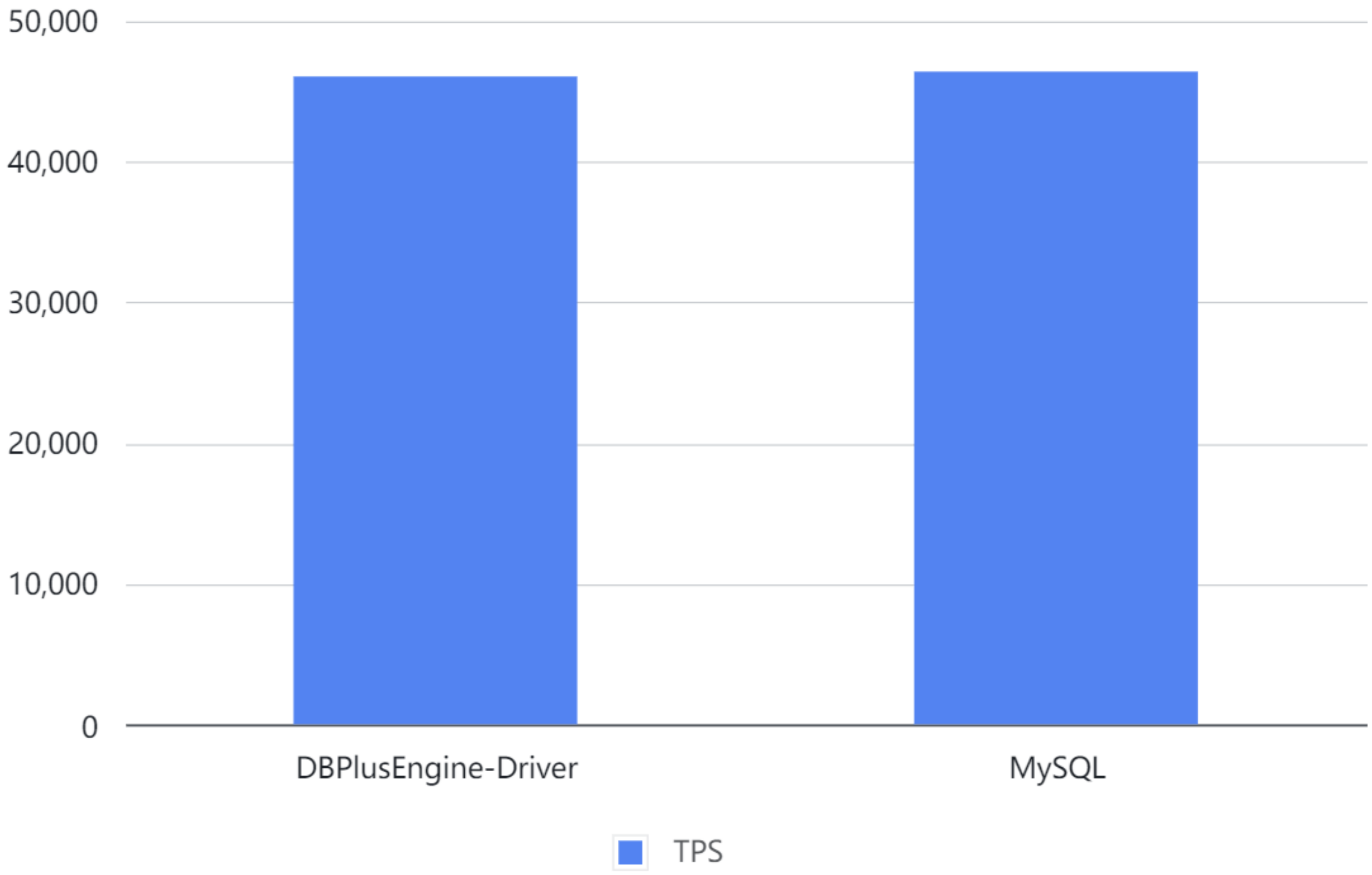
3.  3. Perform the test.

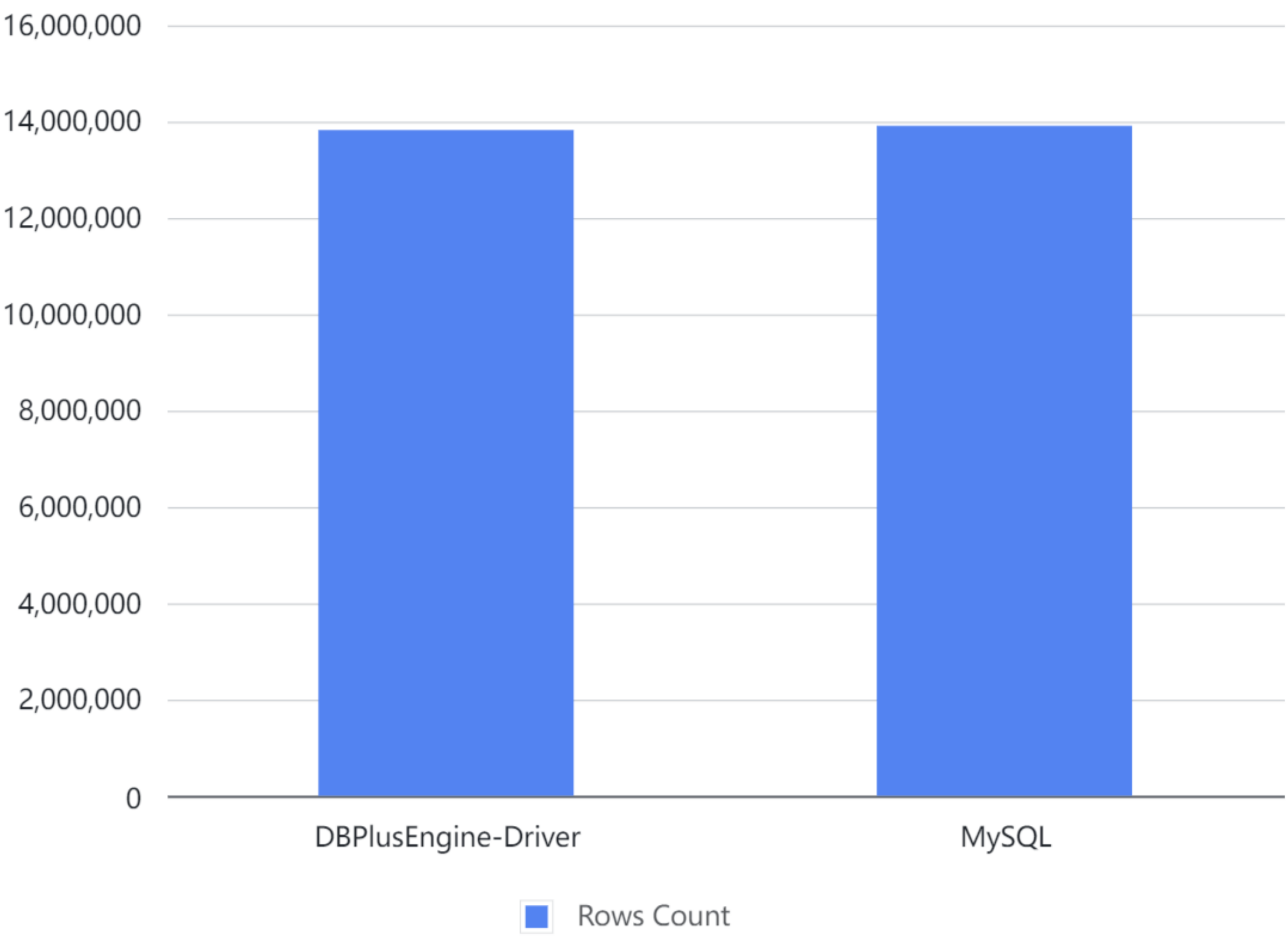
Transfer the path parameter of test.yaml and other JMH parameters using the following commands (Appendix).

```
# If you'd like to learn about JMH parameters, you can run -help after the following commands.
java -classpath 'dependency/*:jmh-shardingsphere5-1.0-SNAPSHOT.jar'  -Dshardingsphere.configurationFile=/root/testssj.yaml  -
Dtables=1  org.openjdk.jmh.Main "com.sphereex.jmh.shardingsphere5.ShardingSphereInsertOnlyBenchmark" -f 1 -i1 -r200 -t  100
-wi 0
java -classpath 'dependency/*:jmh-jdbc-1.0-SNAPSHOT.jar'  -Dconf=/root/mysql.properties  -Dtables=1  org.openjdk.jmh.Main
"com.sphereex.jmh.jdbc.UnpooledReadOnlyBenchmarkJDBC" -f 1 -i1 -r200 -t  100 -wi 0
```

## Test Result

| Test Object | TPS | Rows Count |
|---|---|---|
| DBPlusEngine-Driver | 46,055 | 13,821,323 |
| MySQL | 46,402 | 13,909,701 |



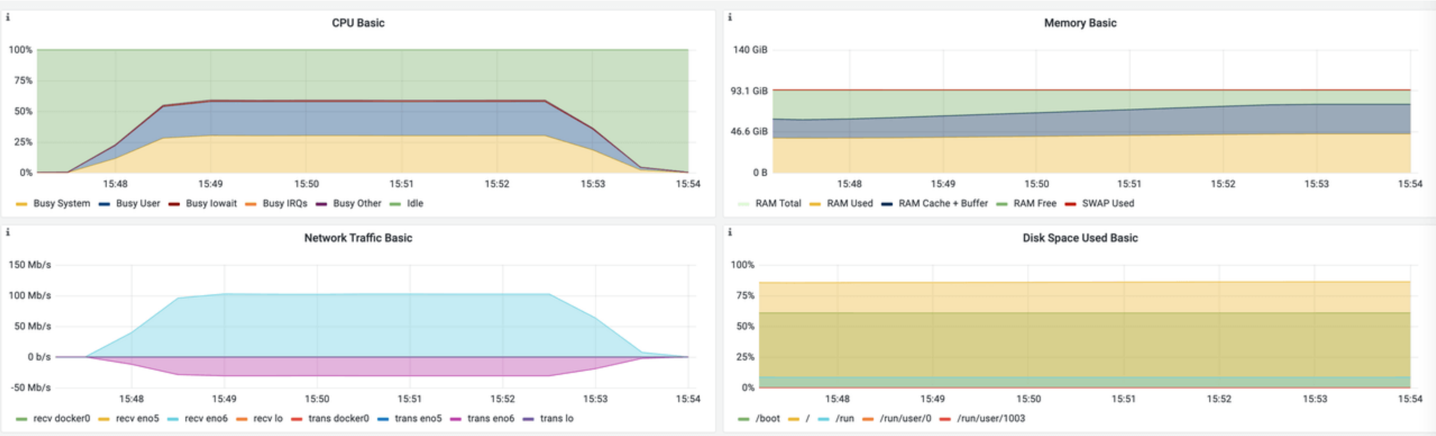


Result: by comparing the imports of DBPlusEngine-Driver and MySQL, it turns out that there’s hardly any performance loss for DBPlusEngine-Driver.

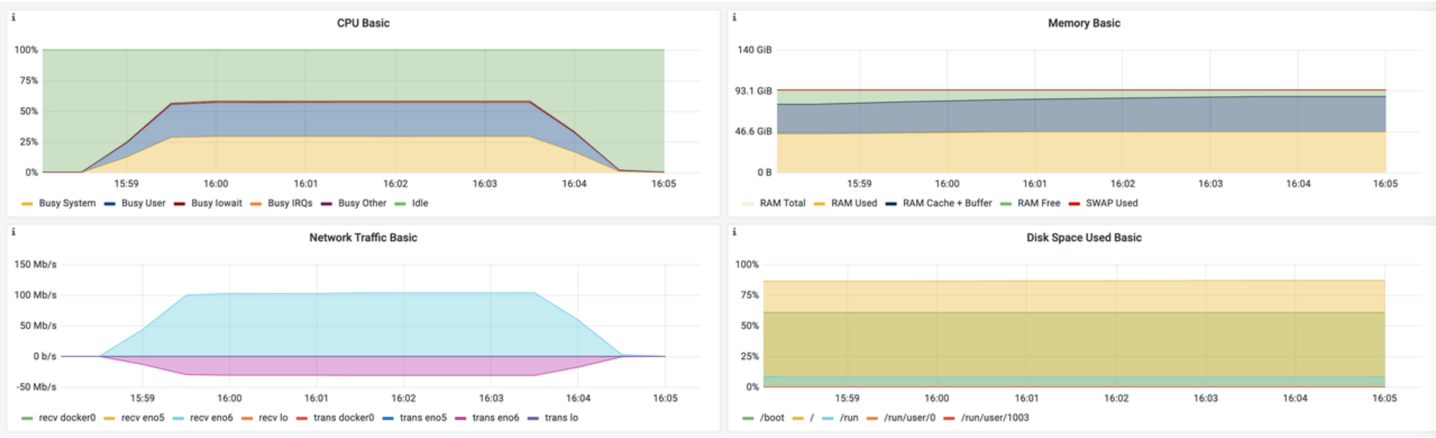
The result is for reference only. Users can carry out a test based on actual business scenarios.

## Monitoring Information

DBPlusEngine-Driver



MySQL



### Glossary

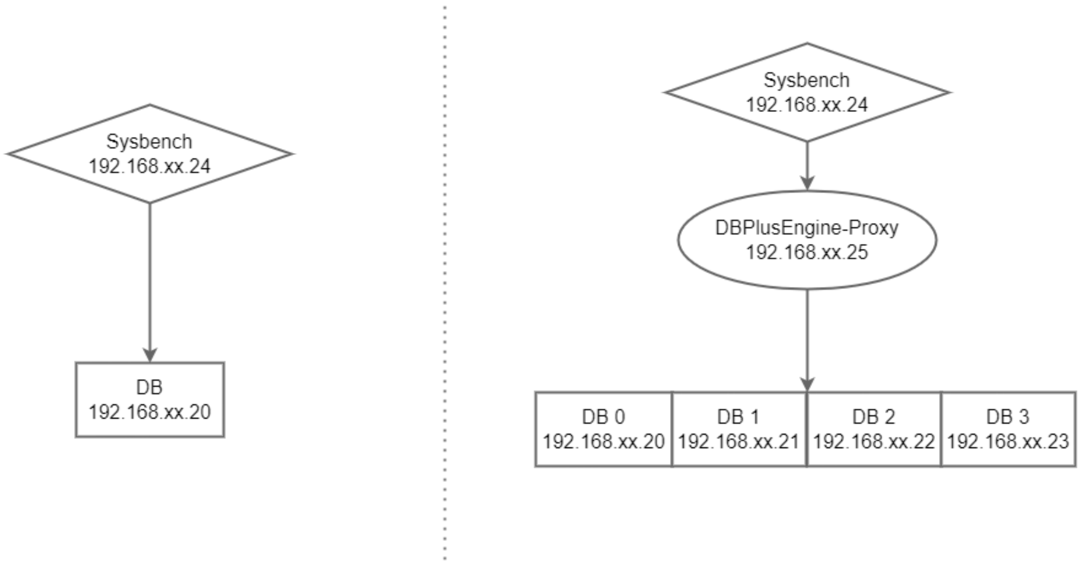
TPS: transaction per second

## 5.3 Business Performance Test

Provide a universal method for the user-end test.

### 5.3.1 Data Sharding

Data sharding test scenarios.



## Test Objective

Use DBPlusEngine-Proxy' s sharding rules to route and distribute data to corresponding databases and tables. Also use the sysbench tool to carry out stress testing on read/write scenarios, to obtain the performance indicator of the sharding feature.

## Test Tool

Sysbench

## Test Environment

| Application | IP Address | Port | Version |
|---|---|---|---|
| DBPlusEngine-Proxy | 192.168.xx.25 | 3307 | 1.2 |
| MySQL | 192.168.xx.20 | 13306 | 8.0.29 |
| MySQL | 192.168.xx.21 | 13306 | 8.0.29 |
| MySQL | 192.168.xx.22 | 13306 | 8.0.29 |
| MySQL | 192.168.xx.23 | 13306 | 8.0.29 |

Server Configuration

| Item | Configuration |
|---|---|
| CPU | 48 C |
| Memory | 96 G |
| Hard Disk | SSD 820 G |
| JDK | 17.0.2 |

### Test Procedure

1. Create a test database.

```
mysql -utest -h192.168.xx.20 -P13306 -p
```

```
DROP DATABASE IF EXISTS sbtest;

CREATE DATABASE IF NOT EXISTS sbtest;
```

```
mysql -utest -h192.168.xx.21 -P13306 -p
```

```
DROP DATABASE IF EXISTS sbtest;

CREATE DATABASE IF NOT EXISTS sbtest;
```

```
mysql -utest -h192.168.xx.22 -P13306 -p
```

```
DROP DATABASE IF EXISTS sbtest;

CREATE DATABASE IF NOT EXISTS sbtest;
```

```
mysql -utest -h192.168.xx.23 -P13306 -p
```

```
DROP DATABASE IF EXISTS sbtest;

CREATE DATABASE IF NOT EXISTS sbtest;
```

Create a logical database.

```
mysql -uroot -h192.168.xx.25 -P3307 -proot
```

```
DROP DATABASE IF EXISTS sharding_db;

CREATE DATABASE IF NOT EXISTS sharding_db;
```

Associate the data source through DistSQL.

```
REGISTER STORAGE UNIT ds_0 (
  HOST="192.168.xx.20",
  PORT=13306,
  DB="sbtest",
  USER="test",
  PASSWORD="test"
), ds_1 (
  HOST="192.168.xx.21",
  PORT=13306,
  DB="sbtest",
  USER="test",
  PASSWORD="test"
) , ds_2 (
  HOST="192.168.xx.22",
  PORT=13306,
  DB="sbtest",
  USER="test",
  PASSWORD="test"
) , ds_3 (
  HOST="192.168.xx.23",
  PORT=13306,
  DB="sbtest",
  USER="test",
  PASSWORD="test"
);
```

Create sharding rules through DistSQL.

```
CREATE SHARDING TABLE RULE t_user (
STORAGE_UNITS(ds_0, ds_1, ds_2, ds_3),
SHARDING_COLUMN=id,TYPE(NAME="MOD",PROPERTIES("sharding-count"=10))
);
```

2. Install test tools.

```
tar -xzvf sysbench-1.0.20.tar.gz -C /usr/local
cd /usr/local/sysbench-1.0.20
./autogen.sh
./configure
make && make install
```

3. Obtain data.

```
sysbench oltp_read_write --mysql-host='192.168.xx.25' --mysql-port=3307 --mysql-user=root --mysql-password='root' --mysql-db=sharding_db --tables=1 --table-size=140000000 --report-interval=5  --time=600 --threads=128 --max-requests=0 --percentile=99  --mysql-ignore-errors="all" --range_selects=off --rand-type=uniform --auto_inc=off cleanup

sysbench oltp_read_write --mysql-host='192.168.xx.25' --mysql-port=3307 --mysql-user=root --mysql-password='root' --mysql-db=sharding_db --tables=1 --table-size=140000000 --report-interval=5  --time=600 --threads=128 --max-requests=0 --percentile=99  --mysql-ignore-errors="all" --range_selects=off --rand-type=uniform --auto_inc=off prepare
```

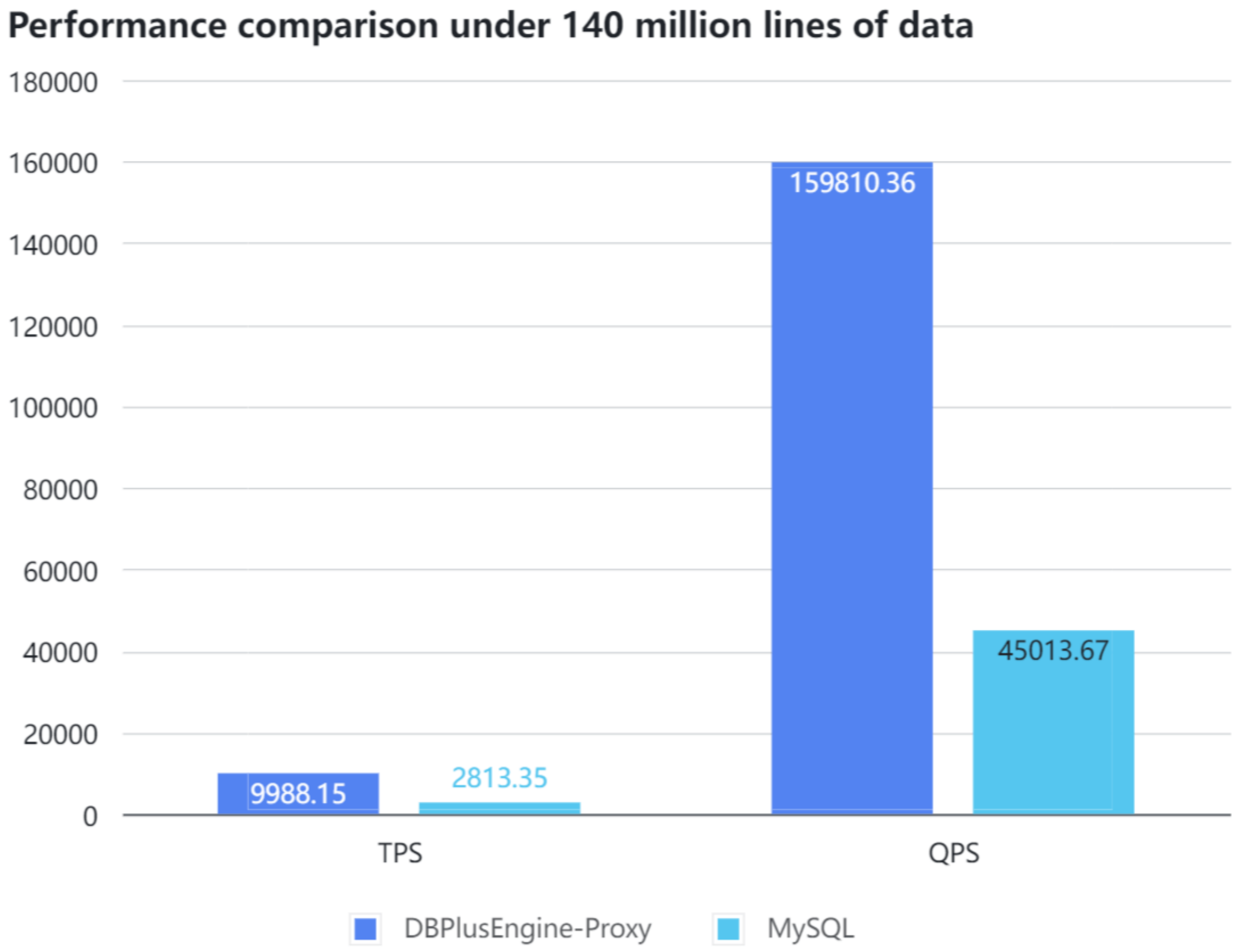
4. Carry out stress testing.

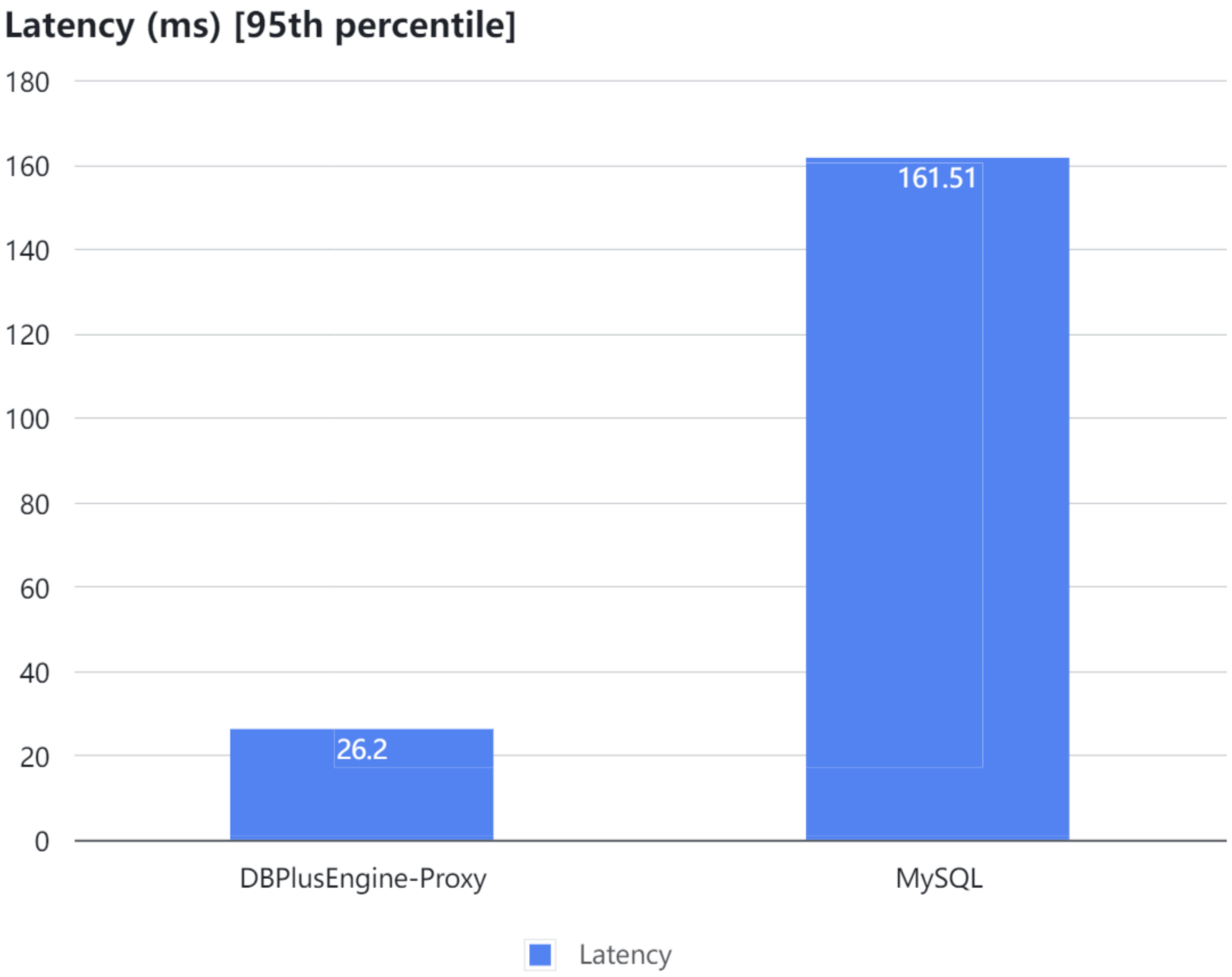
- Read/write Test.

```
sysbench oltp_read_write --mysql-host='192.168.xx.25' --mysql-port=3307 --mysql-user=root --mysql-password='root' --mysql-db=sharding_db --tables=1 --table-size=140000000 --report-interval=5  --time=180 --threads=200 --max-requests=0   --mysql-ignore-errors="all" --range_selects=off --rand-type=uniform --auto_inc=off run
```

## Test Results

| Application | Threads | TPS | Latency99%(ms) |
|---|---|---|---|
| DBPlusEngine-Proxy | 256 | 10,710 | 43.39 |
| MySQL | 256 | 1,827 | 475.79 |


Performance comparison under 140 million lines of data

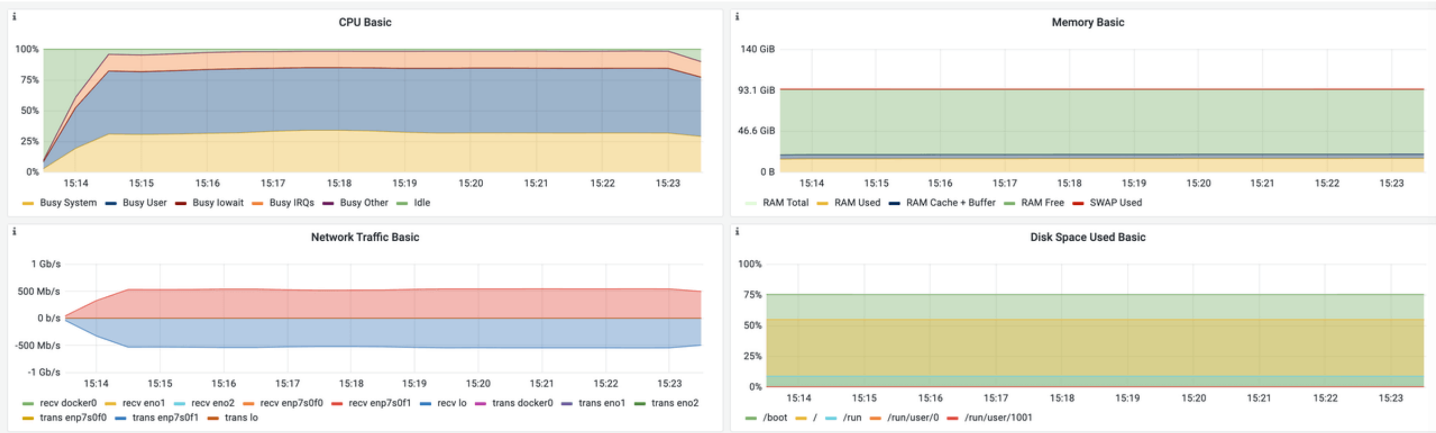


Result: under sharding scenarios, scaling out computing nodes can improve the throughput linearly. The result is for reference only. Users can carry out a test based on actual business scenarios.

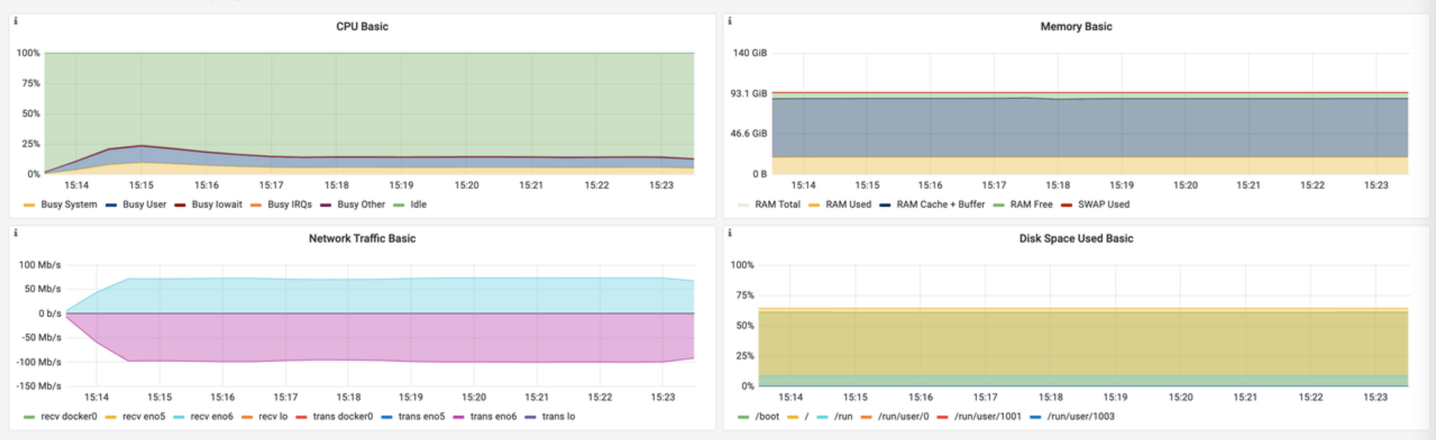
### Monitoring Information

#### Sysbench ——> Proxy ——> MySQL(4)

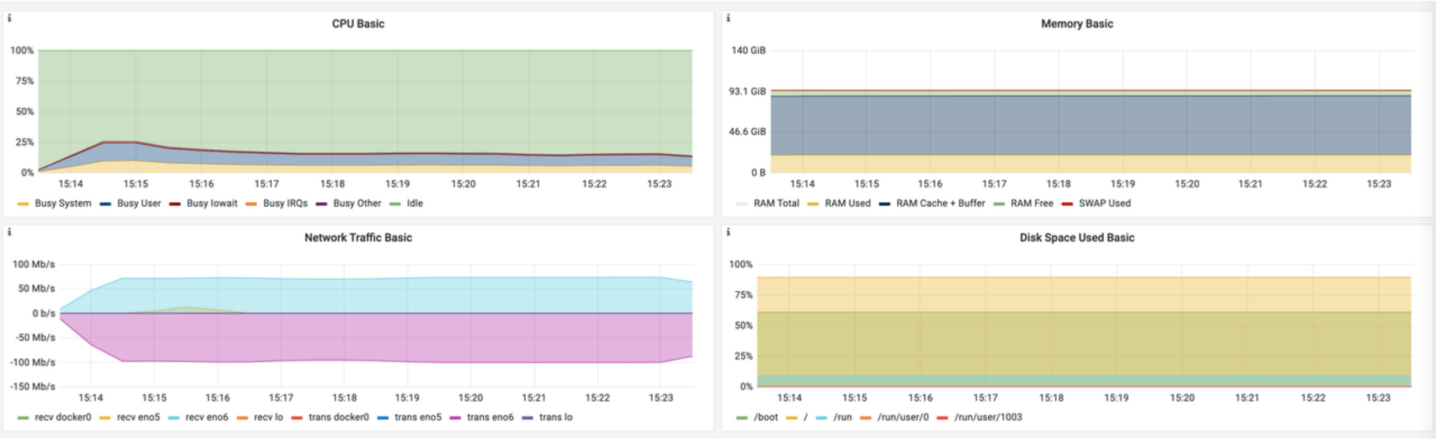
- Proxy



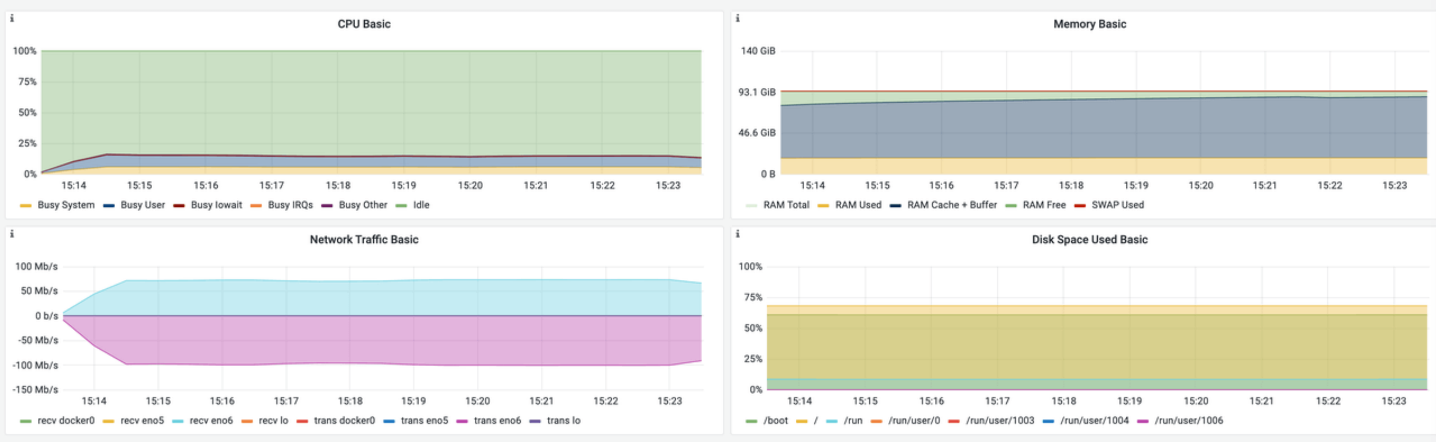
- MySQL(1)



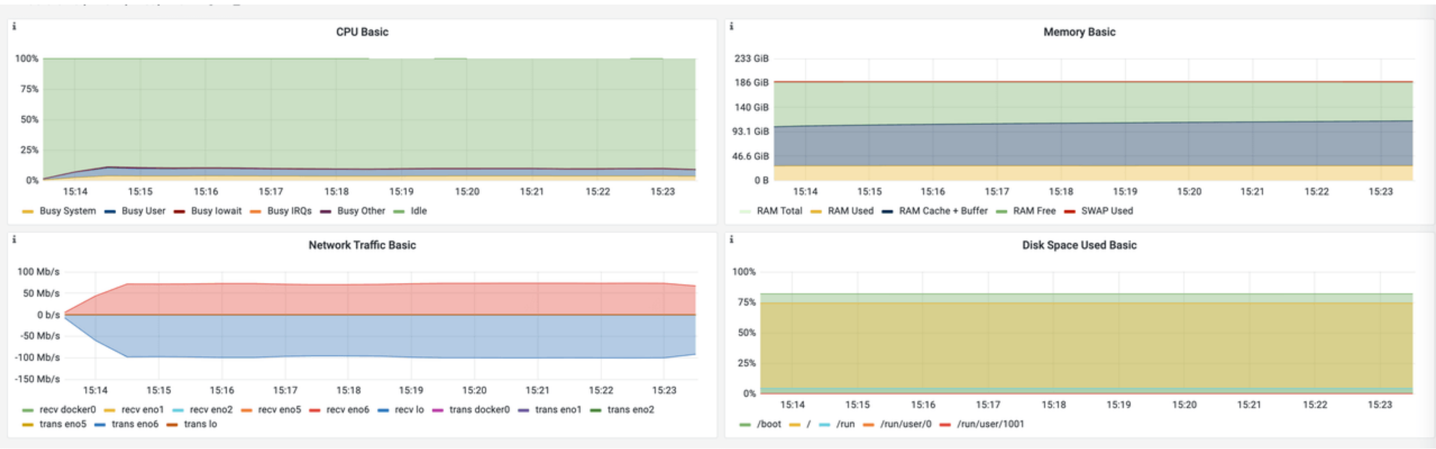
- MySQL(2)



- MySQL(3)

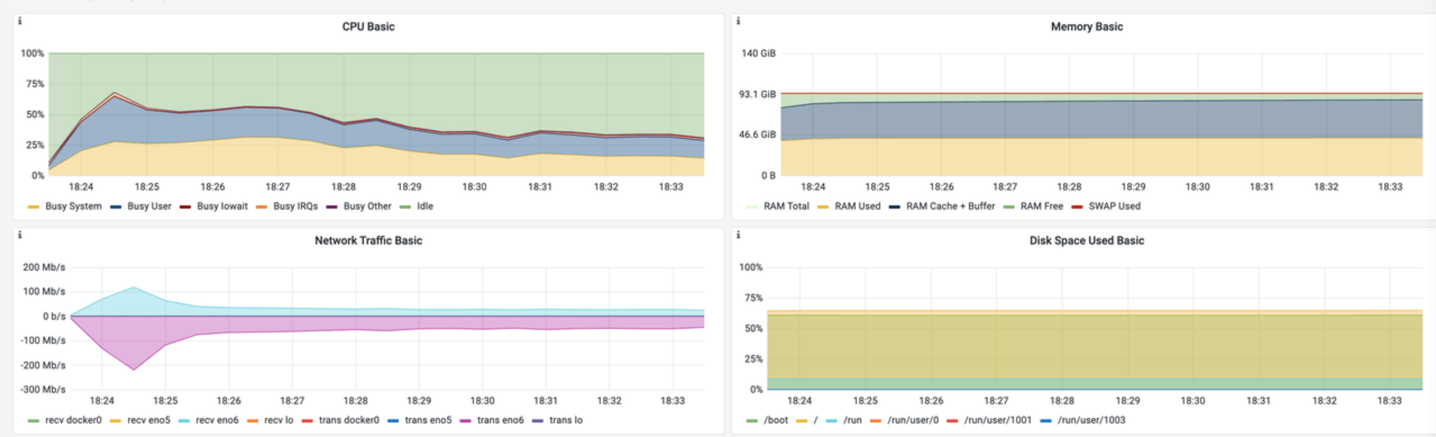


- MySQL(4)



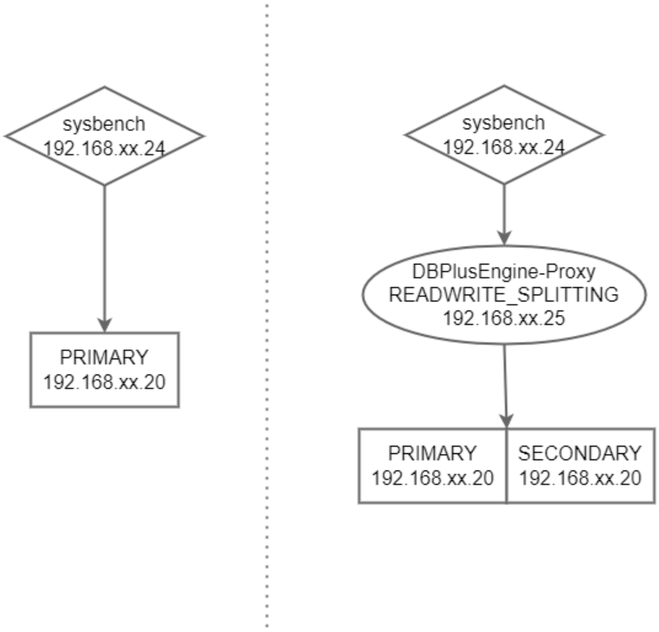
## Sysbench ——> MySQL

- MySQL



## 5.3.2 Read/Write Splitting

Read/Write Test Scenario.



### Test Objective

According to read/write splitting rules, DBPlusEngine-Proxy routes the read and write requests into the primary database and secondary database respectively. We use the sysbench tool to carry out stress testing on read/write scenarios, to obtain the performance indicator of the read/write splitting feature.

### Test Tool

Sysbench

### Test Environment

| Application | IP Address | Port | Version |
|---|---|---|---|
| DBPlusEngine-Proxy | 192.168.xx.25 | 3307 | 1.2 |
| MySQL | 192.168.xx.20 | 13308 | 8.0.29 |
| MySQL | 192.168.xx.20 | 13309 | 8.0.29 |

Server Configuration

| Item | Configuration |
|---|---|
| CPU | 48 C |
| Memory | 96 G |
| Hard Disk | SSD 820 G |
| JDK | 17.0.2 |

### Test Procedure

1. Create a test database.

```
#Primary database
mysql -utest -h192.168.xx.20 -P13308 -p
```

```
CREATE DATABASE IF NOT EXISTS test;
```

Create a logical database.

```
mysql -uroot -h192.168.xx.25 -P3307 -proot
```

```
DROP DATABASE IF EXISTS sphereex_demo;

CREATE DATABASE IF NOT EXISTS sphereex_demo;
```

Associate the data source through DistSQL.

```
use sphereex_demo;

REGISTER STORAGE UNIT write_ds (
   HOST="192.168.xx.20",
   PORT=13308,
   DB="test",
   USER="test",
   PASSWORD="test"
), read_ds (
   HOST="192.168.xx.20",
   PORT=13309,
   DB="test",
   USER="test",
   PASSWORD="test"
);
```

Create read/write splitting rules through DistSQL.

```
CREATE READWRITE_SPLITTING RULE readwrite_ds (
WRITE_STORAGE_UNIT=write_ds,
READ_STORAGE_UNITS(read_ds),
TYPE(NAME="FIXED_REPLICA_RANDOM")
);
```

2. Install test tools.

```
tar -xzvf sysbench-1.0.20.tar.gz -C /usr/local
cd /usr/local/sysbench-1.0.20
./autogen.sh
./configure
make && make install
```

3. Obtain data.

```
sysbench oltp_point_select --mysql-host='192.168.xx.25' --mysql-port=3307 --mysql-user=test --mysql-password='test' --mysql-db=sphereex_demo --tables=10 --table-size=1000000 --report-interval=10 --time=3600 --threads=256 --max-requests=0 --percentile=99 --mysql-ignore-errors="all" --rand-type=uniform --range_selects=off --auto_inc=off cleanup

sysbench oltp_point_select --mysql-host='192.168.xx.25' --mysql-port=3307 --mysql-user=test --mysql-password='test' --mysql-db=sphereex_demo --tables=1 --table-size=1000000 --report-interval=10 --time=3600 --threads=256 --max-requests=0 --percentile=99 --mysql-ignore-errors="all" --rand-type=uniform --range_selects=off --auto_inc=off prepare
```

4. Carry out stress testing.

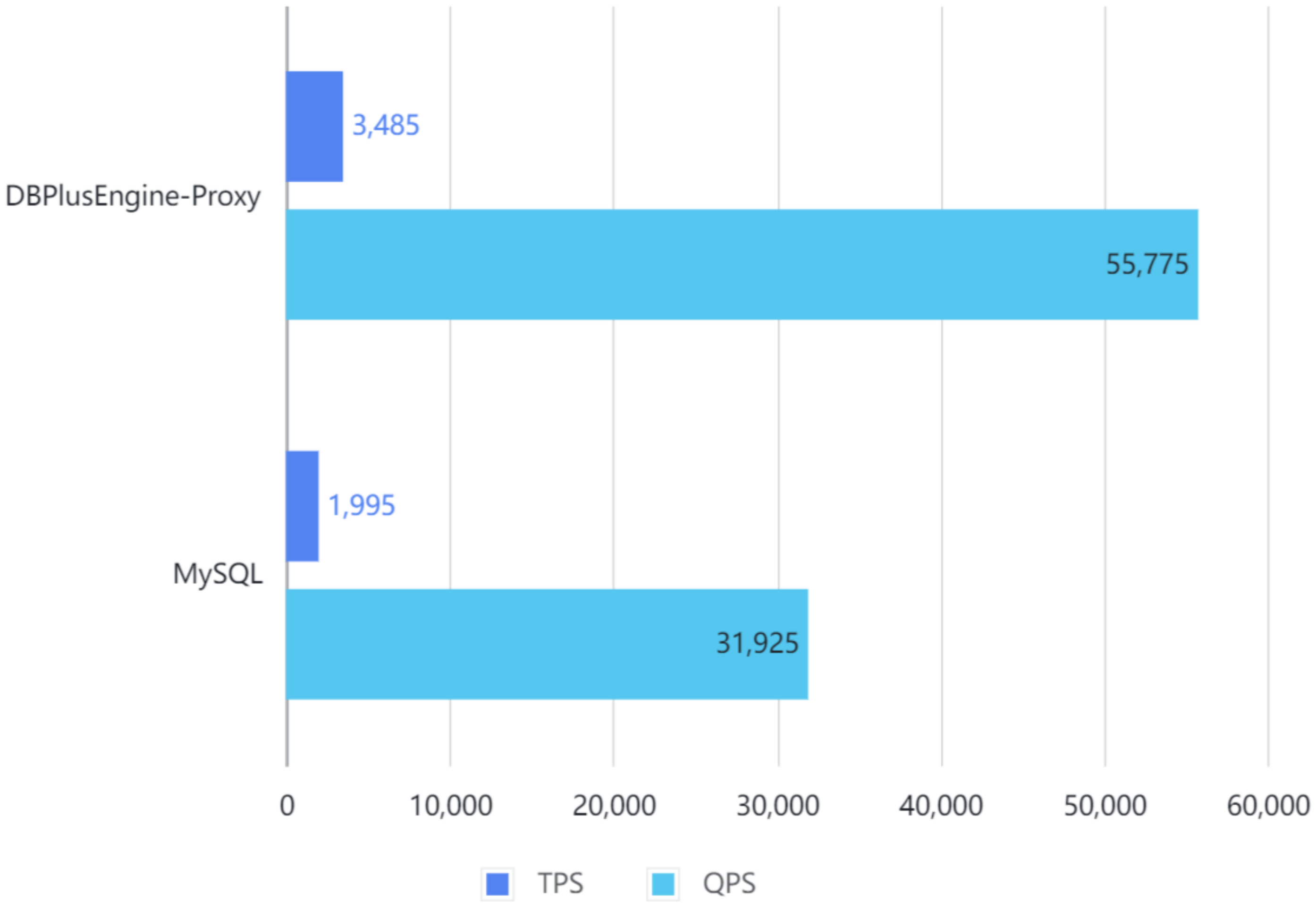
- Read/Write Test

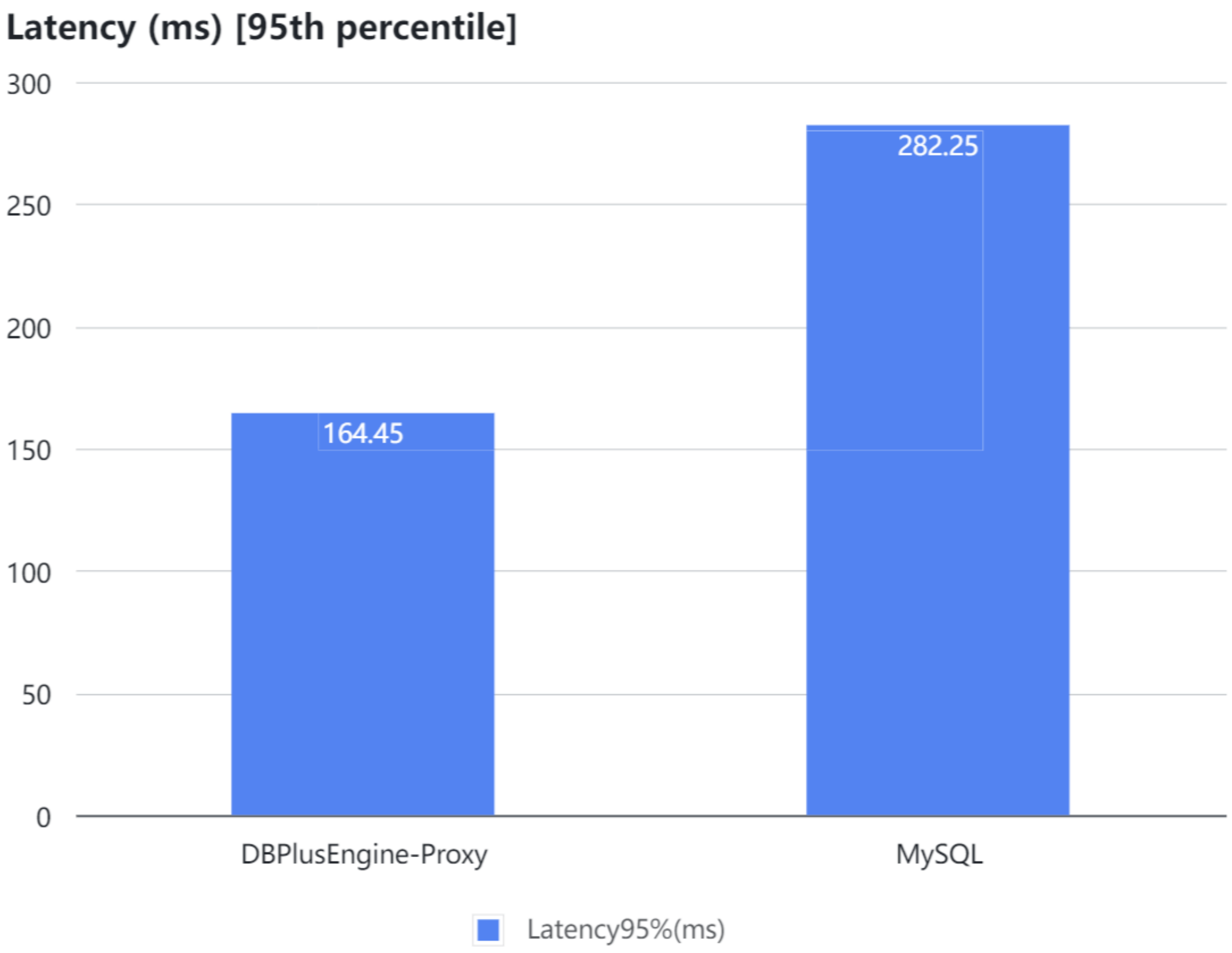
```
sysbench oltp_read_write --mysql-host='192.168.xx.25' --mysql-port=3307 --mysql-user=root --mysql-password='root' --mysql-db=sphereex_demo --tables=1 --table-size=1000000 --report-interval=5  --time=120 --threads=200 --max-requests=0  --mysql-ignore-errors="all" --range_selects=off --rand-type=uniform --auto_inc=off run

sysbench oltp_read_write --mysql-host='192.168.xx.20' --mysql-port=13308 --mysql-user=root --mysql-password='root' --mysql-db=test --tables=1 --table-size=1000000 --report-interval=5  --time=120 --threads=200 --max-requests=0  --mysql-ignore-errors="all" --range_selects=off --rand-type=uniform --auto_inc=off run
```

### Test Result

```
# sysbench ——> Proxy ——> MySQL
SQL statistics:
    queries performed:
        read:                            4185530
        write:                           1674212
        other:                           837106
        total:                           6696848
    transactions:                        418553 (3485.96 per sec.)
    queries:                             6696848 (55775.38 per sec.)
    ignored errors:                      0      (0.00 per sec.)
    reconnects:                          0      (0.00 per sec.)

General statistics:
    total time:                          120.0640s
    total number of events:              418553

Latency (ms):
         min:                                    7.61
         avg:                                   57.35
         max:                                 1035.85
         95th percentile:                      164.45
         sum:                              24005109.36

Threads fairness:
    events (avg/stddev):           2092.7650/22.11
    execution time (avg/stddev):   120.0255/0.01

# sysbench ——> MySQL
SQL statistics:
    queries performed:
        read:                            2398650
        write:                           959460
        other:                           479730
        total:                           3837840
    transactions:                        239865 (1995.35 per sec.)
    queries:                             3837840 (31925.58 per sec.)
    ignored errors:                      0      (0.00 per sec.)
    reconnects:                          0      (0.00 per sec.)

General statistics:
    total time:                          120.2079s
    total number of events:              239865

Latency (ms):
         min:                                    2.93
         avg:                                  100.16
         max:                                 1735.34
         95th percentile:                      282.25
         sum:                             24023941.63

Threads fairness:
    events (avg/stddev):           1199.3250/13.23
    execution time (avg/stddev):   120.1197/0.08
```

| Application | TPS | QPS | Latency95%(ms) |
|---|---|---|---|
| DBPlusEngine-Proxy | 3,485 | 55,775 | 164.45 |
| MySQL | 1,995 | 31,925 | 282.25 |


DBPlusEngine-Proxy
3,485
55,775
MySQL
1,995
31,925
0
10,000
20,000
30,000
40,000
50,000
60,000
TPS
QPS

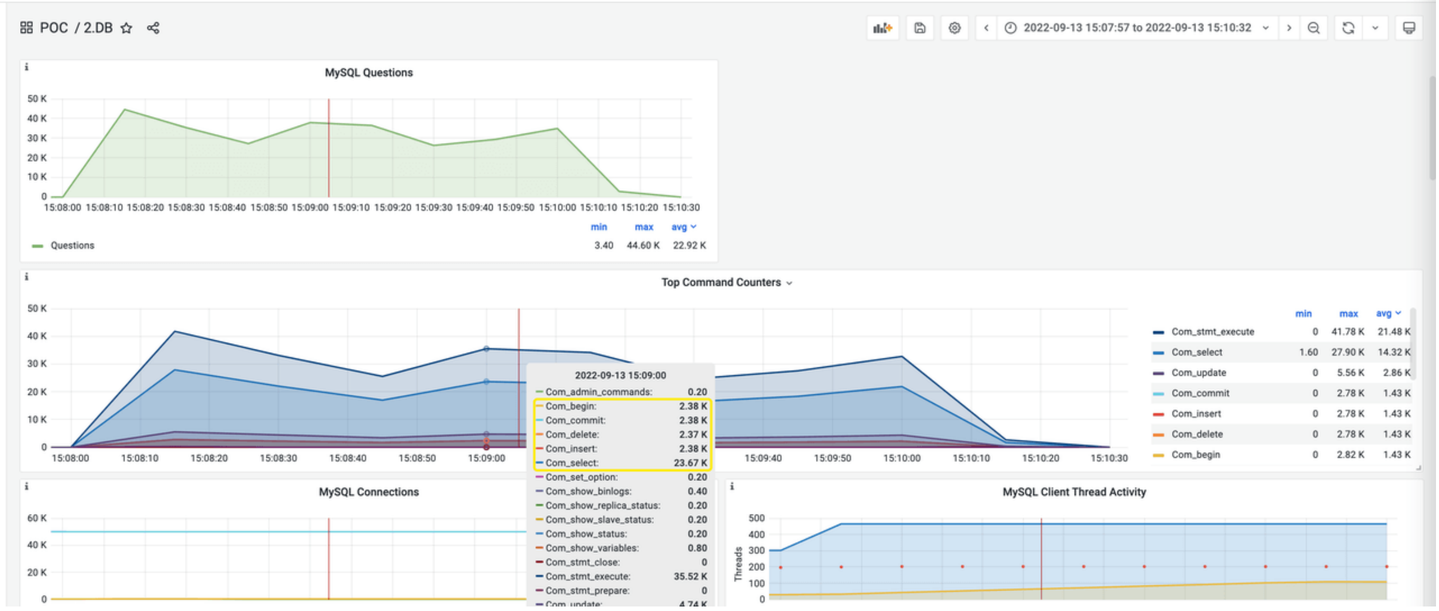


Result: under read/write splitting scenarios, the performance and throughput are improved significantly. The result is for reference only. Users can carry out a test based on actual business scenarios.

### Monitoring Information

#### sysbench ——> MySQL

PRIMARY



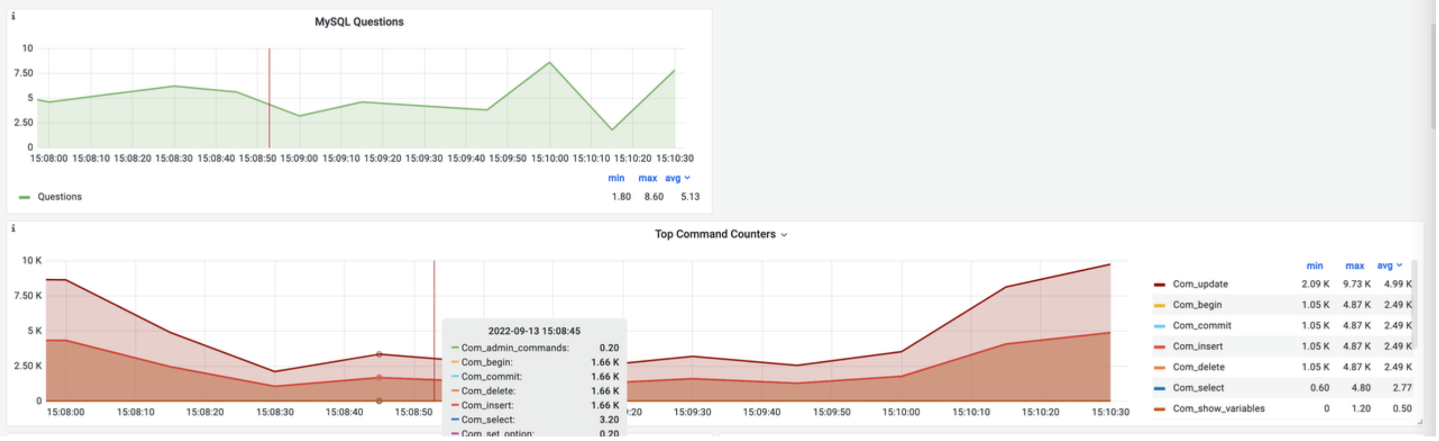
|  | min | max | avg |
|---|---|---|---|
| Questions | 3.40 | 44.60 K | 22.92 K |

|  | min | max | avg |
|---|---|---|---|
| Com_stmt_execute | 0 | 41.78 K | 21.48 K |
| Com_select | 1.60 | 27.90 K | 14.32 K |
| Com_update | 0 | 5.56 K | 2.86 K |
| Com_commit | 0 | 2.78 K | 1.43 K |
| Com_insert | 0 | 2.78 K | 1.43 K |
| Com_delete | 0 | 2.78 K | 1.43 K |
| Com_begin | 0 | 2.82 K | 1.43 K |

2022-09-13 15:09:00

|  |  |
|---|---|
| Com_admin_commands: | 0.20 |
| Com_begin: | 2.38 K |
| Com_commit: | 2.38 K |
| Com_delete: | 2.37 K |
| Com_insert: | 2.38 K |
| Com_select: | 23.67 K |
| Com_set_option: | 0.20 |
| Com_show_binlogs: | 0.40 |
| Com_show_replica_status: | 0.20 |
| Com_show_slave_status: | 0.20 |
| Com_show_status: | 0.20 |
| Com_show_variables: | 0.80 |
| Com_stmt_close: | 0 |
| Com_stmt_execute: | 35.52 K |
| Com_stmt_prepare: | 0 |
| Com_update: | 4.74 K |

SECONDARY



|  | min | max | avg |
|---|---|---|---|
| Questions | 1.80 | 8.60 | 5.13 |

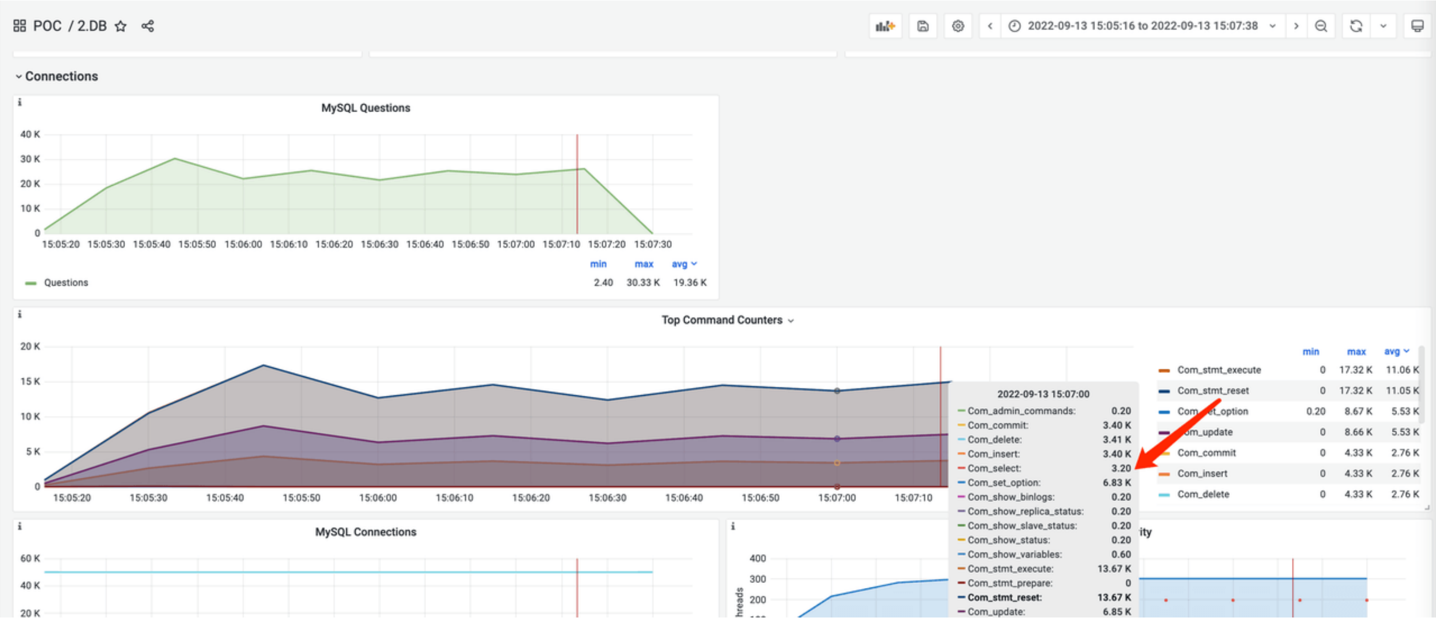
|  | min | max | avg |
|---|---|---|---|
| Com_update | 2.09 K | 9.73 K | 4.99 K |
| Com_begin | 1.05 K | 4.87 K | 2.49 K |
| Com_commit | 1.05 K | 4.87 K | 2.49 K |
| Com_insert | 1.05 K | 4.87 K | 2.49 K |
| Com_delete | 1.05 K | 4.87 K | 2.49 K |
| Com_select | 0.60 | 4.80 | 2.77 |
| Com_show_variables | 0 | 1.20 | 0.50 |

2022-09-13 15:08:45

|  |  |
|---|---|
| Com_admin_commands: | 0.20 |
| Com_begin: | 1.66 K |
| Com_commit: | 1.66 K |
| Com_delete: | 1.66 K |
| Com_insert: | 1.66 K |
| Com_select: | 3.20 |
| Com_set_option: | 0.20 |

## sysbench ——> Proxy ——> MySQL

PRIMARY

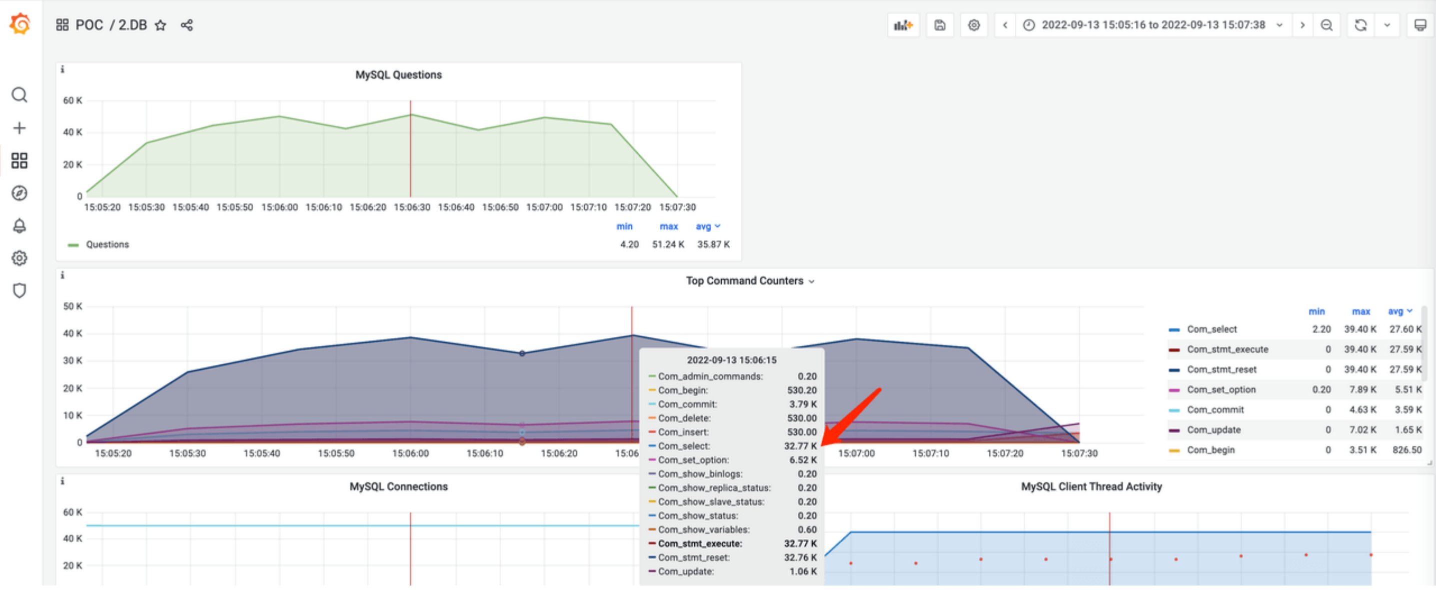


| | min | max | avg |
|---|---|---|---|
| Questions | 2.40 | 30.33 K | 19.36 K |

| 2022-09-13 15:07:00 | |
|---|---|
| Com_admin_commands: | 0.20 |
| Com_commit: | 3.40 K |
| Com_delete: | 3.41 K |
| Com_insert: | 3.40 K |
| Com_select: | 3.20 |
| Com_set_option: | 6.83 K |
| Com_show_binlogs: | 0.20 |
| Com_show_replica_status: | 0.20 |
| Com_show_slave_status: | 0.20 |
| Com_show_status: | 0.20 |
| Com_show_variables: | 0.60 |
| Com_stmt_execute: | 13.67 K |
| Com_stmt_prepare: | 0 |
| **Com_stmt_reset:** | **13.67 K** |
| Com_update: | 6.85 K |

| | min | max | avg |
|---|---|---|---|
| Com_stmt_execute | 0 | 17.32 K | 11.06 K |
| Com_stmt_reset | 0 | 17.32 K | 11.05 K |
| Com_set_option | 0.20 | 8.67 K | 5.53 K |
| Com_update | 0 | 8.66 K | 5.53 K |
| Com_commit | 0 | 4.33 K | 2.76 K |
| Com_insert | 0 | 4.33 K | 2.76 K |
| Com_delete | 0 | 4.33 K | 2.76 K |

SECONDARY



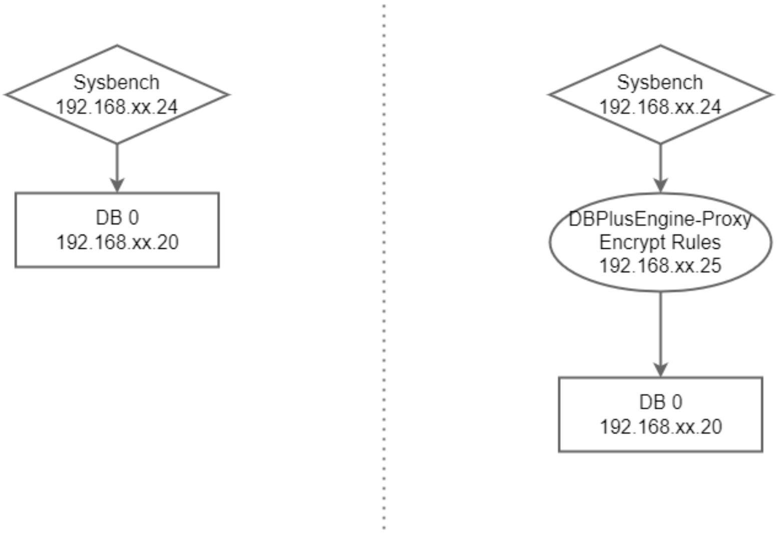
| | min | max | avg |
|---|---|---|---|
| Questions | 4.20 | 51.24 K | 35.87 K |

| 2022-09-13 15:06:15 | |
|---|---|
| Com_admin_commands: | 0.20 |
| Com_begin: | 530.20 |
| Com_commit: | 3.79 K |
| Com_delete: | 530.00 |
| Com_insert: | 530.00 |
| Com_select: | 32.77 K |
| Com_set_option: | 6.52 K |
| Com_show_binlogs: | 0.20 |
| Com_show_replica_status: | 0.20 |
| Com_show_slave_status: | 0.20 |
| Com_show_status: | 0.20 |
| Com_show_variables: | 0.60 |
| **Com_stmt_execute:** | **32.77 K** |
| Com_stmt_reset: | 32.76 K |
| Com_update: | 1.06 K |

| | min | max | avg |
|---|---|---|---|
| Com_select | 2.20 | 39.40 K | 27.60 K |
| Com_stmt_execute | 0 | 39.40 K | 27.59 K |
| Com_stmt_reset | 0 | 39.40 K | 27.59 K |
| Com_set_option | 0.20 | 7.89 K | 5.51 K |
| Com_commit | 0 | 4.63 K | 3.59 K |
| Com_update | 0 | 7.02 K | 1.65 K |
| Com_begin | 0 | 3.51 K | 826.50 |

## 5.3.3 Data Encryption

Data Encryption Test Scenario.



## Test Objective

According to the encryption rules, DBPlusEngine-Proxy inserts data into databases. We use the sysbench tool to carry out stress testing under read/write scenarios, to obtain the performance indicator of the encryption feature.

## Test Tool

Sysbench

## Test Environment

| Application | IP Address | Port | Version |
|---|---|---|---|
| DBPlusEngine-Proxy | 192.168.xx.25 | 3307 | 1.2 |
| MySQL | 192.168.xx.20 | 13306 | 8.0.29 |

Server Configuration

| Item | Configuration |
|---|---|
| CPU | 48 C |
| Memory | 96 G |
| Hard Disk | SSD 820 G |
| JDK | 17.0.2 |

## Test Procedure

1. Create a test database.

```
mysql -utest -h192.168.xx.20 -P13306 -p
```

```
DROP DATABASE IF EXISTS encrypt_db;

CREATE DATABASE IF NOT EXISTS encrypt_db;
```

Create a logical database.

```
mysql -uroot -h192.168.xx.25 -P3307 -proot
```

```
DROP DATABASE IF EXISTS sphereex_demo;

CREATE DATABASE IF NOT EXISTS sphereex_demo;
```

Associate the data source through DistSQL.

```
USE sphereex_demo;

REGISTER STORAGE UNIT sphereex_demo(
  HOST="192.168.xx.20",
  PORT=13306,
  DB="encrypt_db",
  USER="test",
  PASSWORD="test"
  );
```

Create encryption rules through DistSQL.

```
CREATE ENCRYPT RULE t_user (
COLUMNS(
(NAME=idCard,CIPHER=idcard_cipher,TYPE(NAME="AES",PROPERTIES("aes-key-value"="123456abc"))),
(NAME=mobile, CIPHER =mobile_cipher,TYPE(NAME="AES",PROPERTIES("aes-key-value"="123456abc")))));
```

2. Install test tools.

```
tar -xzvf sysbench-1.0.20.tar.gz -C /usr/local
cd /usr/local/sysbench-1.0.20
./autogen.sh
./configure
make && make install
```

3. Obtain data.

```
sysbench oltp_read_only --mysql-host='192.168.xx.25' --mysql-port=3307 --mysql-user=root --mysql-password='root' --mysql-db=sphereex_demo --tables=10 --table-size=1000000 --report-interval=10 --time=3600 --threads=128 --max-requests=0 --percentile=99 --mysql-ignore-errors="all" --rand-type=uniform --range_selects=off --auto_inc=off cleanup

sysbench oltp_read_only --mysql-host='192.168.xx.25' --mysql-port=3307 --mysql-user=root --mysql-password='root' --mysql-db=sphereex_demo --tables=10 --table-size=1000000 --report-interval=10 --time=3600 --threads=128 --max-requests=0 --percentile=99 --mysql-ignore-errors="all" --rand-type=uniform --range_selects=off --auto_inc=off prepare
```

4. Carry out stress testing.

Read/Write Test

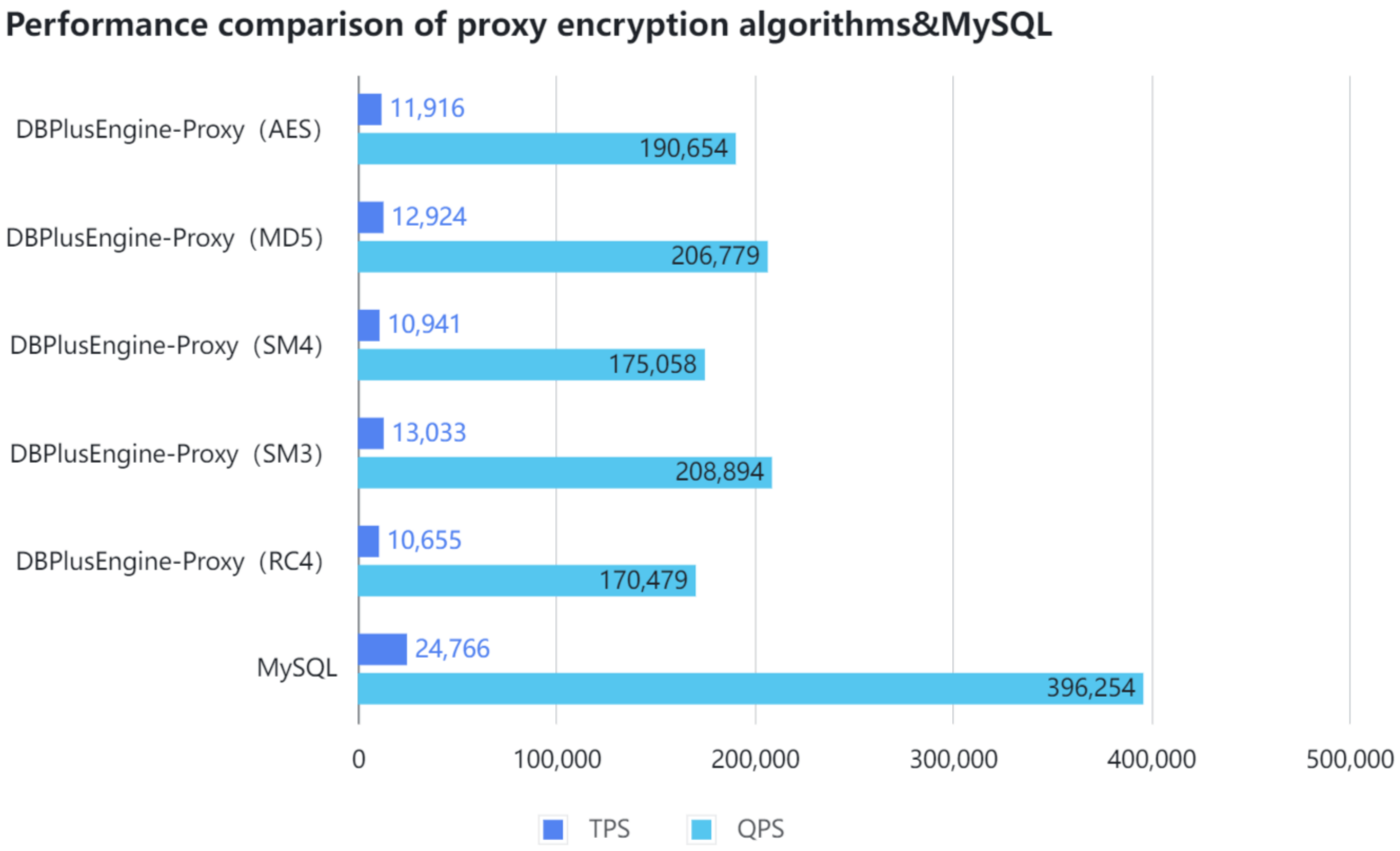
```
sysbench oltp_read_write --mysql-host='192.168.xx.25' --mysql-port=3307 --mysql-user=root --mysql-password='root' --mysql-db=sphereex_demo --tables=1 --table-size=1000000 --report-interval=5  --time=180 --threads=200 --max-requests=0  --mysql-ignore-errors="all" --range_selects=off --rand-type=uniform --auto_inc=off run
```

### Test Result

| Application | TPS | QPS |
|---|---|---|
| DBPlusEngine-Proxy（AES） | 11,916 | 190,654 |
| DBPlusEngine-Proxy（MD5） | 12,924 | 206,779 |
| DBPlusEngine-Proxy（SM4） | 10,941 | 175,058 |
| DBPlusEngine-Proxy（SM3） | 13,033 | 208,894 |
| DBPlusEngine-Proxy（RC4） | 10,655 | 170,479 |
| MySQL | 24,766 | 396,254 |

Result: under data encryption scenarios, the throughput decreased notably, and there are slight differences when using different encryption algorithms.

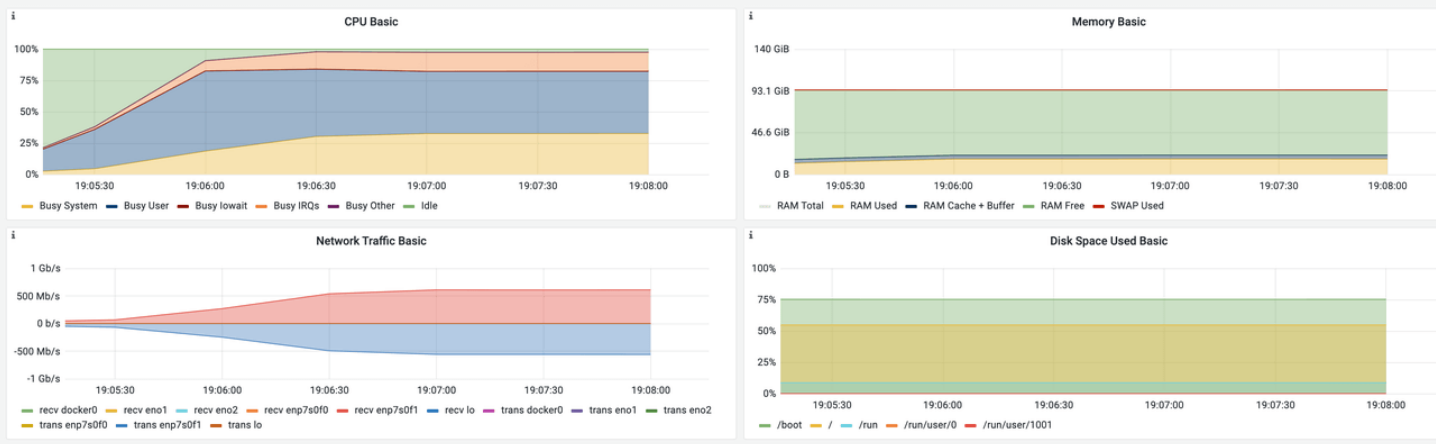
The result is for reference only. Users can carry out a test based on actual business scenarios.



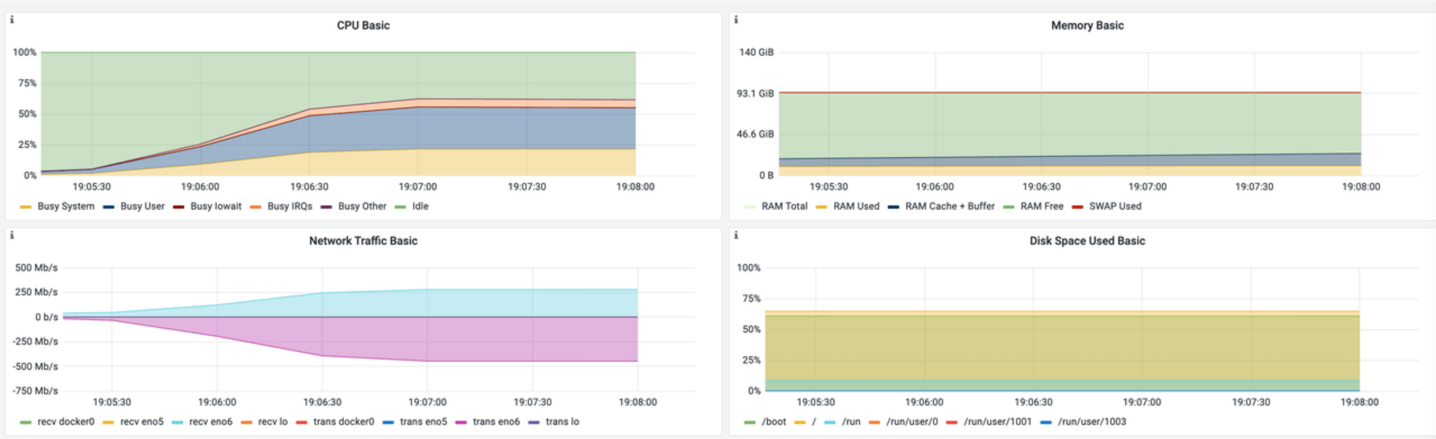
## Monitoring Information

### DBPlusEngine-Proxy（AES）

- Proxy

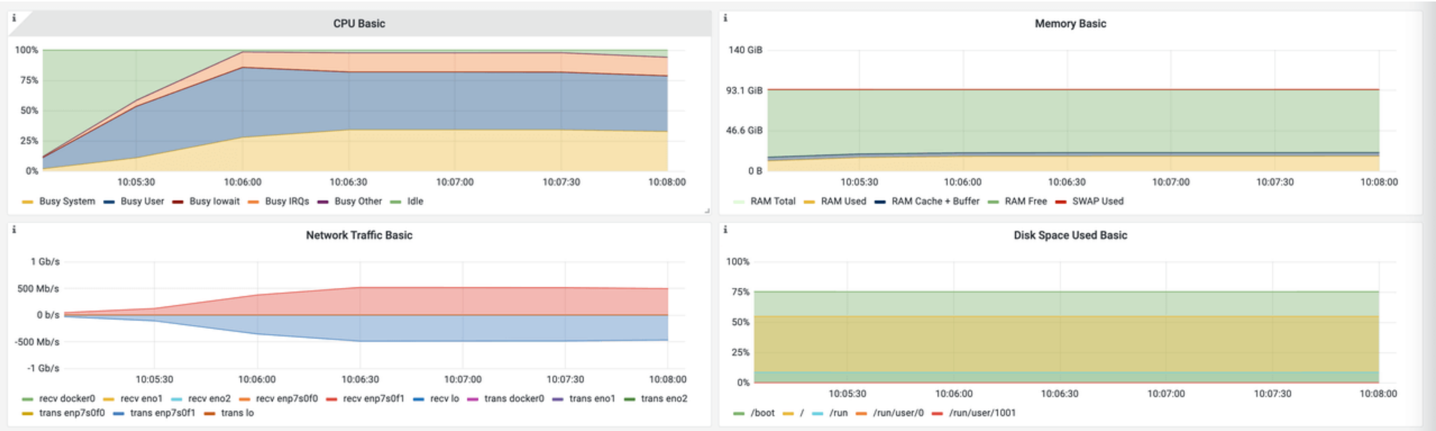


- MySQL

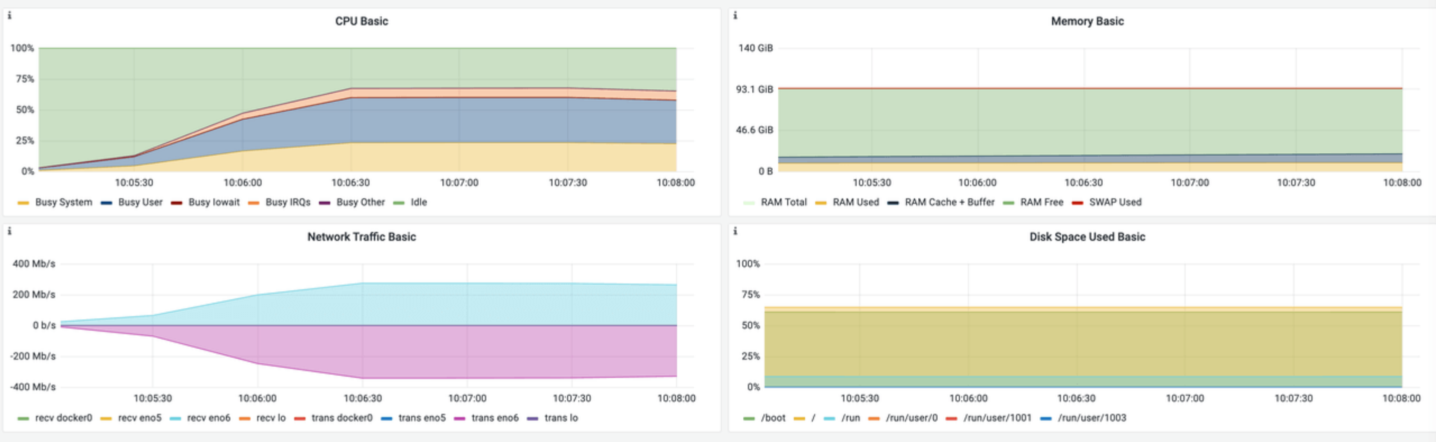


### DB Plus Engine-Proxy（MD5）

- Proxy

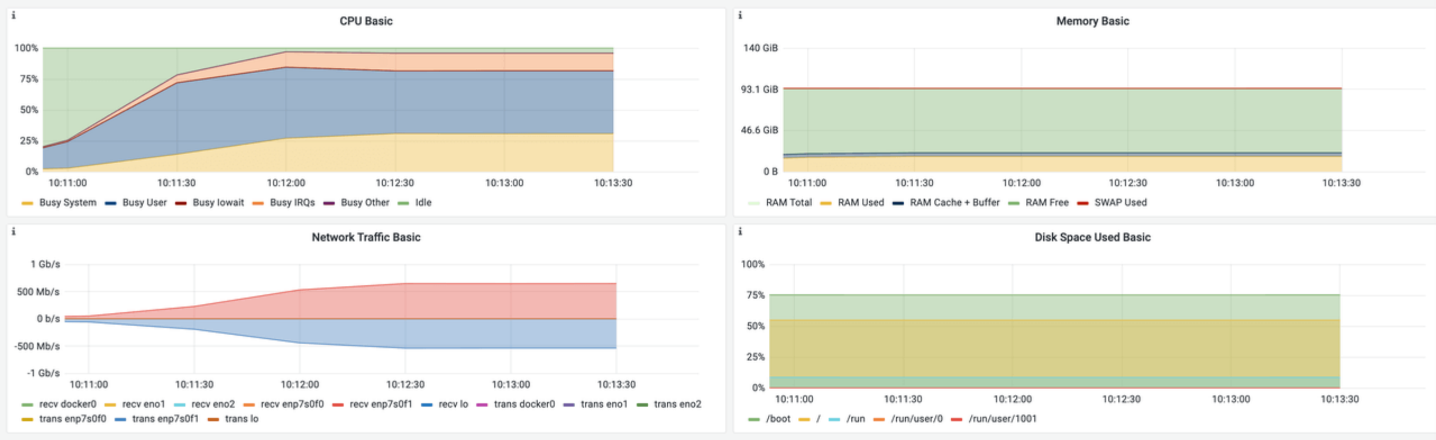


- MySQL

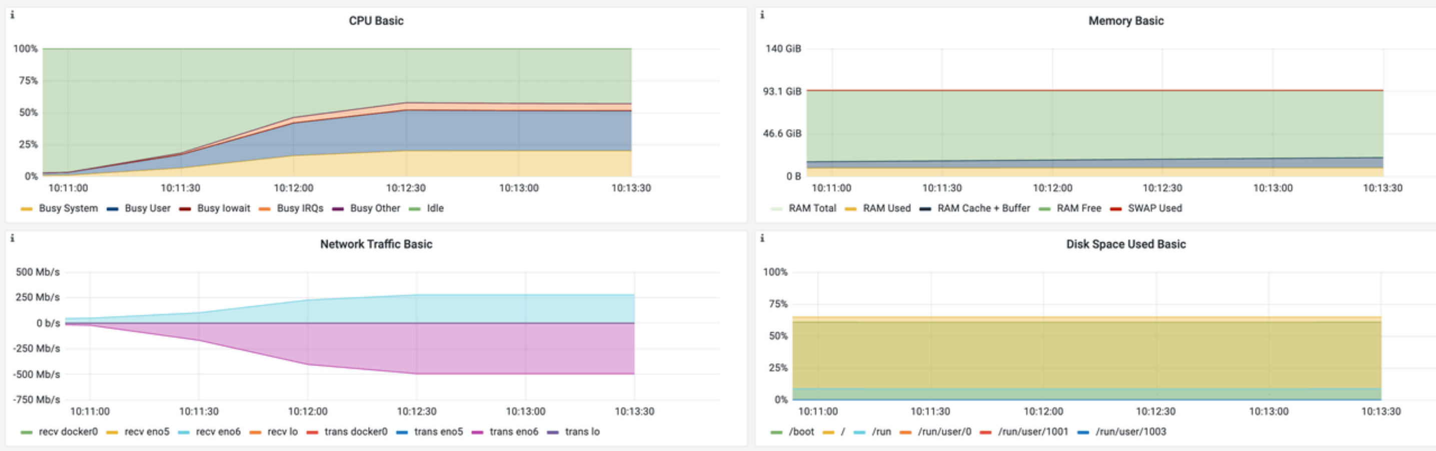


## DB Plus Engine-Proxy（SM4）

- Proxy

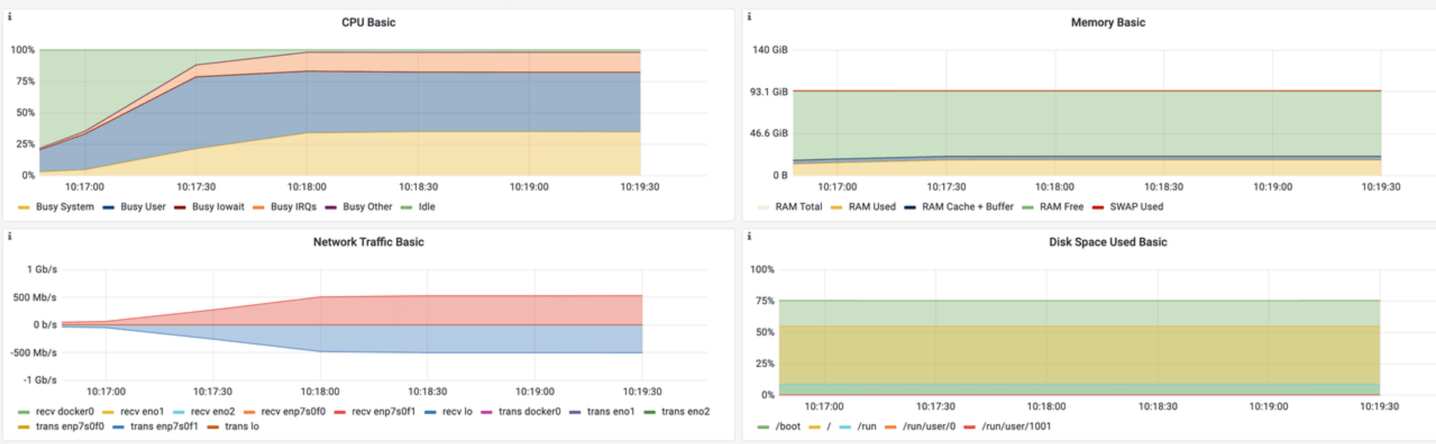


- MySQL

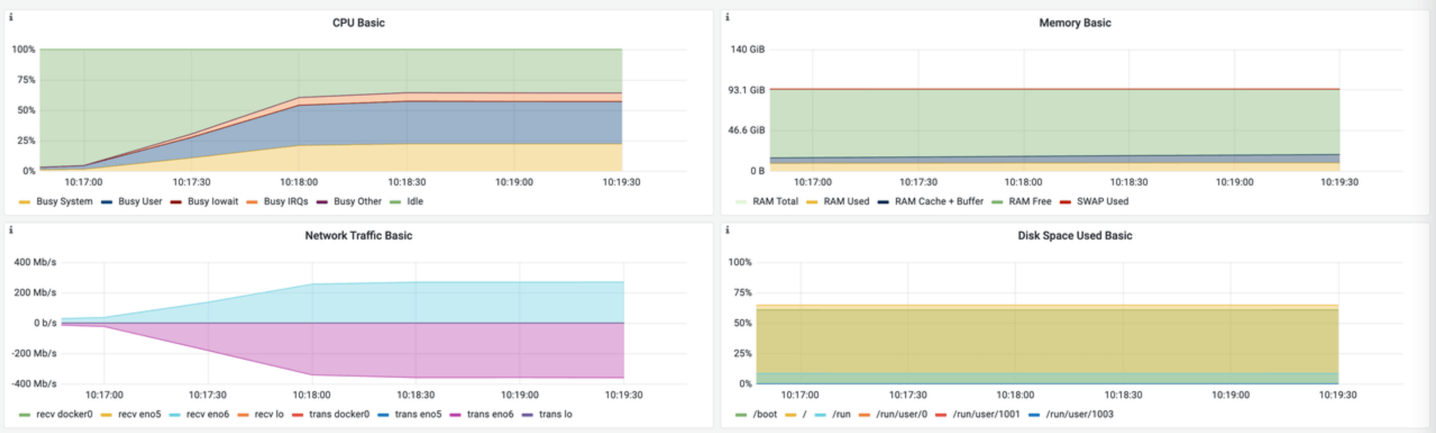


## DB Plus Engine-Proxy（SM3）

- Proxy

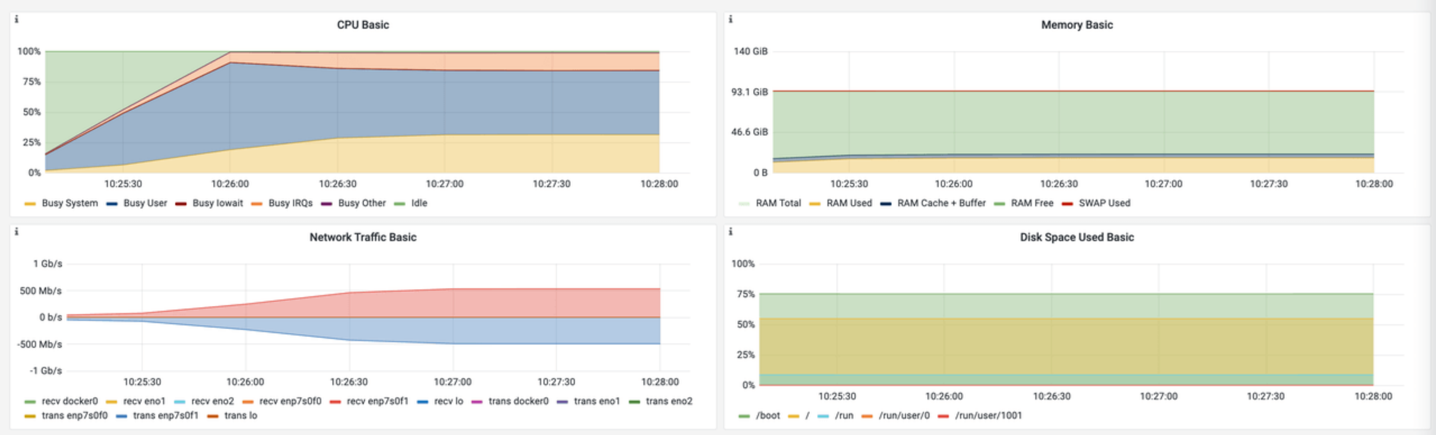


- MySQL

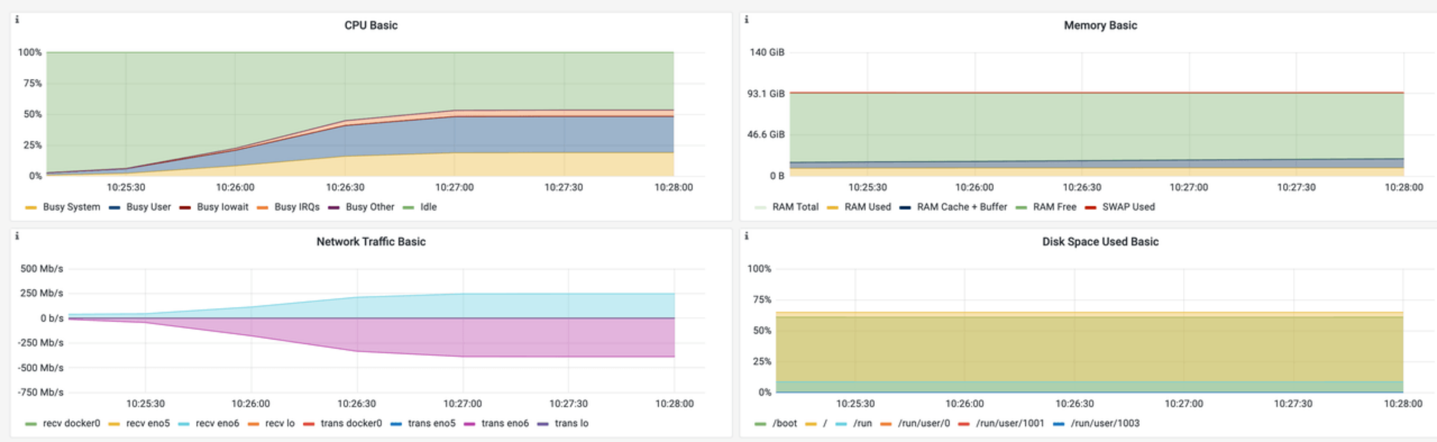


DB Plus Engine-Proxy（RC4）

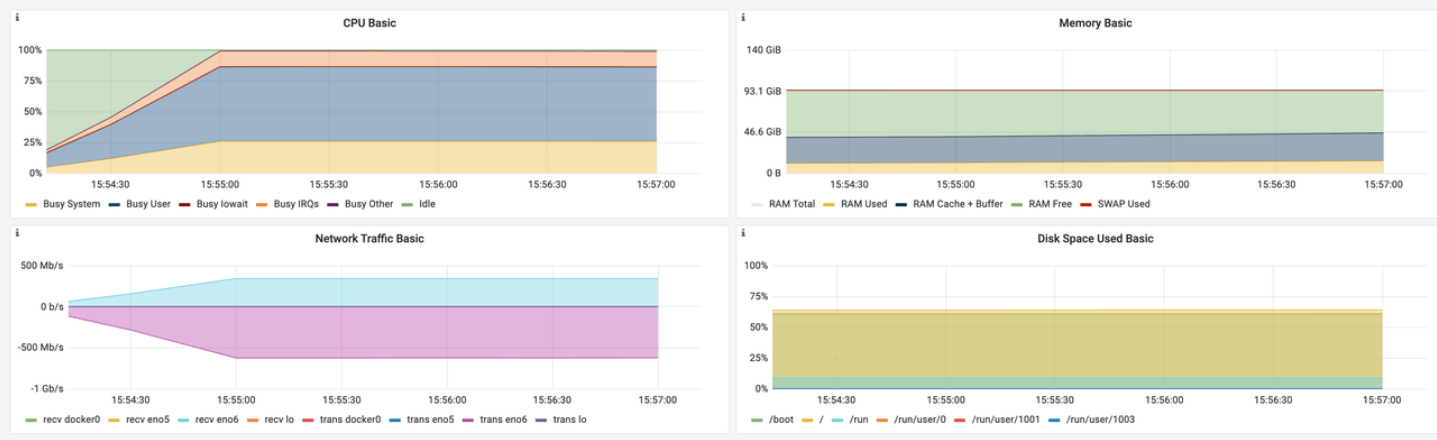
- Proxy



- MySQL

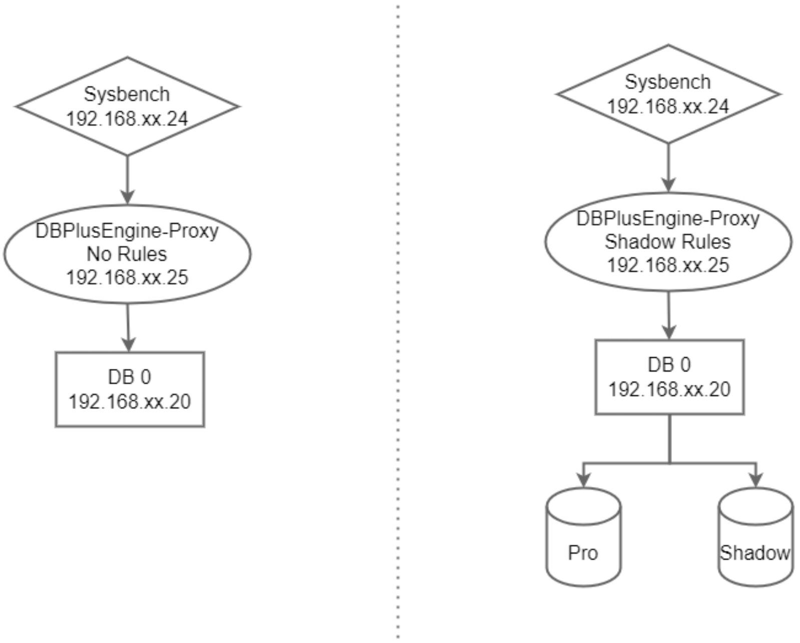


MySQL



## 5.3.4 Shadow Database

Shadow Database Test Scenario.



## Test Objective

Create data for the production database and the shadow database. According to routing rules, make requests for the production database and the shadow database respectively by matching an algorithm for values. We use the sysbench stress testing tool to carry out point queries so as to obtain the performance indicator of the shadow DB feature.

## Test Tool

Sysbench

## Test Environment

| Application | IP Address | Port | Version |
|---|---|---|---|
| DBPlusEngine-Proxy | 192.168.xx.25 | 3307 | 1.2 |
| MySQL | 192.168.xx.20 | 13306 | 8.0.29 |
| MySQL | 192.168.xx.21 | 13306 | 8.0.29 |

Server Configuration

| Item | Configuration |
|---|---|
| CPU | 48 C |
| Memory | 96 G |
| Hard Disk | SSD 820 G |
| JDK | 17.0.2 |

### Test Procedure

1. Create a test database.

Create a production database and a shadow database.

```
mysql -utest -h192.168.xx.20 -P13306 -p
```

```
DROP DATABASE IF EXISTS prod_ds;

CREATE DATABASE IF NOT EXISTS prod_ds;
```

```
mysql -utest -h192.168.xx.21 -P13306 -p
```

```
DROP DATABASE IF EXISTS shadow_ds;

CREATE DATABASE IF NOT EXISTS shadow_ds;
```

Create a logical database.

```
mysql -uroot -h192.168.xx.25 -P3307 -proot
```

```
DROP DATABASE IF EXISTS sphereex_demo;

CREATE DATABASE IF NOT EXISTS sphereex_demo;
```

Associate the data source through DistSQL.

```
USE sphereex_demo;

REGISTER STORAGE UNIT ds (
   HOST="192.168.xx.20",
   PORT=13306,
   DB="prod_ds",
   USER="test",
   PASSWORD="test"
),ds_shadow (
   HOST="192.168.xx.21",
   PORT=13306,
   DB="shadow_ds",
   USER="test",
   PASSWORD="test"
);
```

Create shadow DB rules through DistSQL.

```
CREATE SHADOW RULE shadowDataSource(
SOURCE=ds,
SHADOW=ds_shadow,
t_user(
TYPE(NAME="VALUE_MATCH", PROPERTIES("operation"="insert","column"="status", "value"=1)),
TYPE(NAME="SIMPLE_HINT", PROPERTIES("shadow"=true)))
);
```

2. Install test tools.

```
tar -xzvf sysbench-1.0.20.tar.gz -C /usr/local
cd /usr/local/sysbench-1.0.20
./autogen.sh
./configure
make && make install
```

Transform Sysbench’ s oltp_common.lua scripts. The Shadow’ s hint algorithm is supported.

```
local stmt_defs = {
  point_selects = {
    "SELECT c FROM sbtest%u WHERE id=? /*shadow:true*/",
    t.INT},
  simple_ranges = {
    "SELECT c FROM sbtest%u WHERE id BETWEEN ? AND ?",
    t.INT, t.INT},
  sum_ranges = {
    "SELECT SUM(k) FROM sbtest%u WHERE id BETWEEN ? AND ?",
     t.INT, t.INT},
  order_ranges = {
    "SELECT c FROM sbtest%u WHERE id BETWEEN ? AND ? ORDER BY c",
     t.INT, t.INT},
  distinct_ranges = {
    "SELECT DISTINCT c FROM sbtest%u WHERE id BETWEEN ? AND ? ORDER BY c",
    t.INT, t.INT},
  index_updates = {
    "UPDATE sbtest%u SET k=k+1 WHERE id=? /*shadow:true*/",
    t.INT},
  non_index_updates = {
    "UPDATE sbtest%u SET c=? WHERE id=? /*shadow:true*/",
    {t.CHAR, 120}, t.INT},
  deletes = {
    "DELETE FROM sbtest%u WHERE id=? /*shadow:true*/",
    t.INT},
  inserts = {
    "INSERT INTO sbtest%u (id, k, c, pad) VALUES (?, ?, ?, ?)/*shadow:true*/",
    t.INT, t.INT, {t.CHAR, 120}, {t.CHAR, 60}},
}
```

3. Obtain data.

```
sysbench oltp_read_only --mysql-host='192.168.xx.20' --mysql-port=13306 --mysql-user=test --mysql-password='test' --mysql-
db=prod_ds --tables=10 --table-size=1000000 --report-interval=10 --time=3600 --threads=128 --max-requests=0 --percentile=99 --
mysql-ignore-errors="all" --rand-type=uniform --range_selects=off --auto_inc=off cleanup

sysbench oltp_read_only --mysql-host='192.168.xx.20' --mysql-port=13306 --mysql-user=test --mysql-password='test' --mysql-
db=prod_ds --tables=10 --table-size=1000000 --report-interval=10 --time=3600 --threads=128 --max-requests=0 --percentile=99 --
mysql-ignore-errors="all" --rand-type=uniform --range_selects=off --auto_inc=off prepare

sysbench oltp_read_only --mysql-host='192.168.xx.21' --mysql-port=13306 --mysql-user=test --mysql-password='test' --mysql-
db=shadow_ds --tables=10 --table-size=1000000 --report-interval=10 --time=3600 --threads=128 --max-requests=0 --percentile=99
--mysql-ignore-errors="all" --rand-type=uniform --range_selects=off --auto_inc=off cleanup

sysbench oltp_read_only --mysql-host='192.168.xx.21' --mysql-port=13306 --mysql-user=test --mysql-password='test' --mysql-
db=shadow_ds --tables=10 --table-size=1000000 --report-interval=10 --time=3600 --threads=128 --max-requests=0 --percentile=99
--mysql-ignore-errors="all" --rand-type=uniform --range_selects=off --auto_inc=off prepare
```

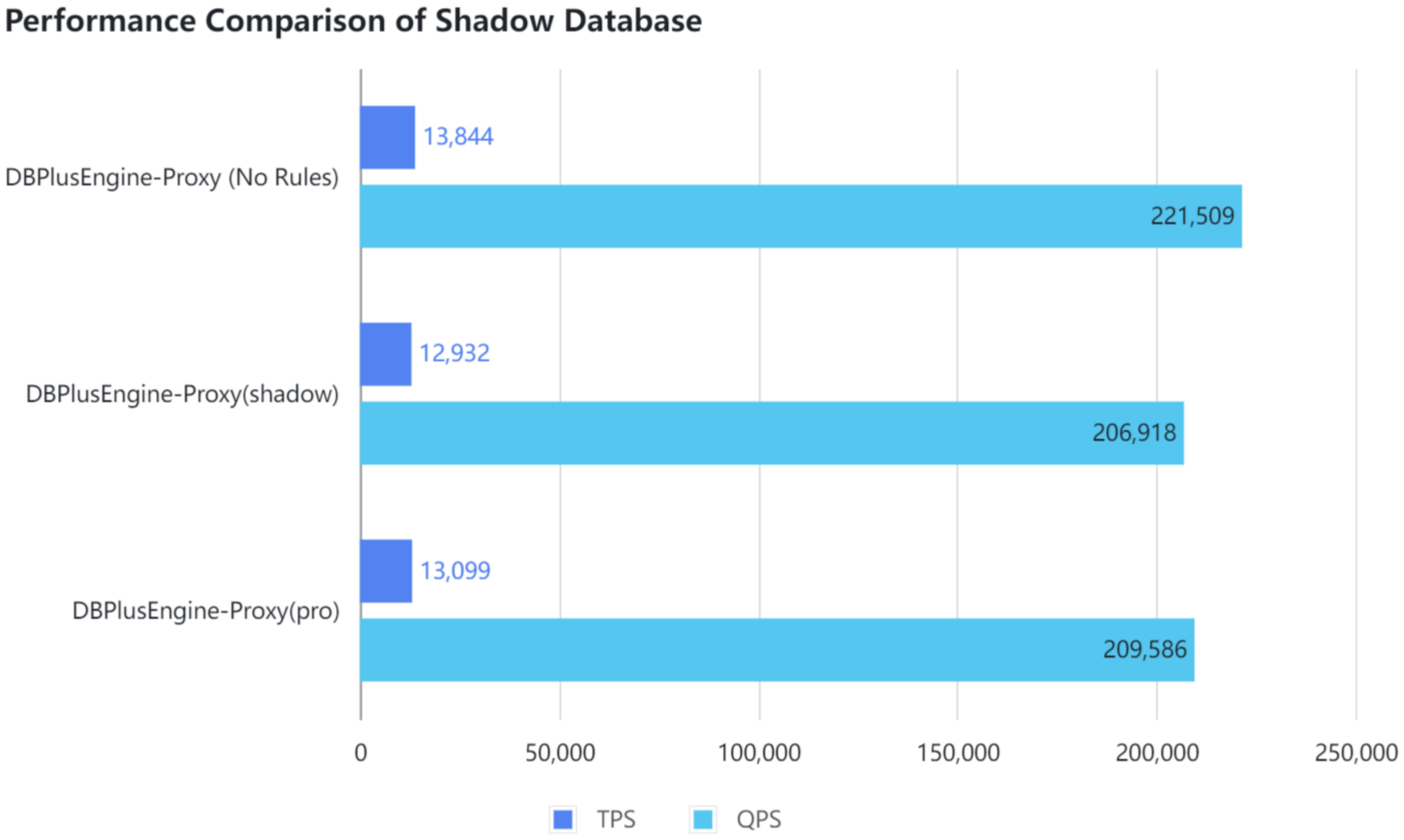
4. Carry out stress testing.

- Read-only Test.

```
sysbench oltp_point_select --mysql-host='192.168.xx.25' --mysql-port=3307 --mysql-user=root --mysql-password='root' --mysql-
db=sphereex_demo --tables=10 --table-size=1000000 --report-interval=5  --time=600 --threads=128 --max-requests=0 --
percentile=99  --mysql-ignore-errors="all" --range_selects=off --rand-type=uniform --auto_inc=off run
```

## Test Result

| Application | TPS | QPS |
|---|---|---|
| DB Plus Engine-Proxy (No Rules) | 13,844 | 221,509 |
| DB Plus Engine-Proxy(shadow) | 12,932 | 206,918 |
| DB Plus Engine-Proxy(pro) | 13,099 | 209,586 |

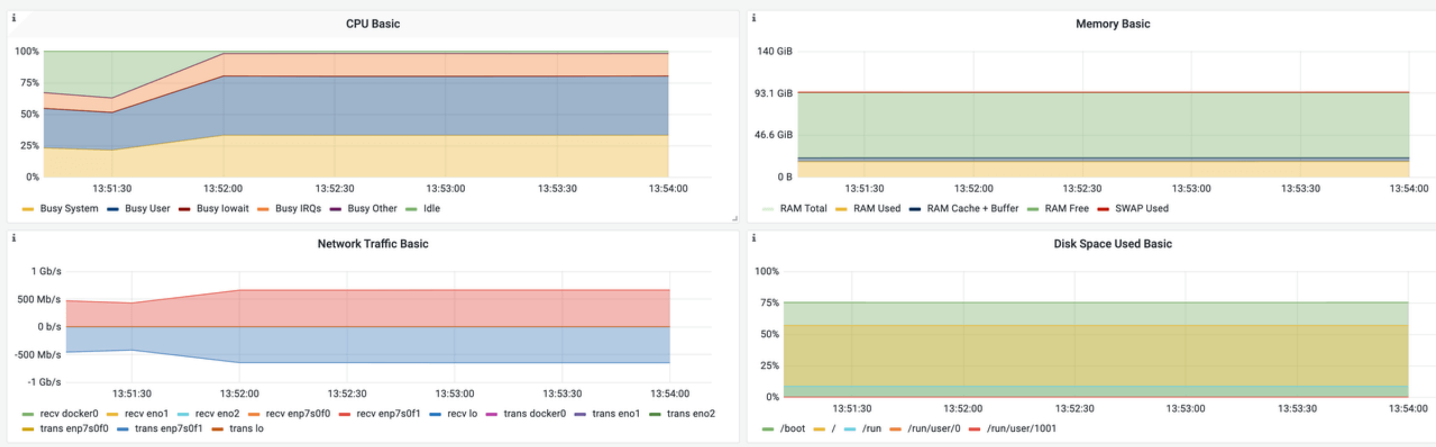


Result: under shadow database scenarios, there’s little difference when routing traffic to a production database or a shadow database. Throughput only decreases slightly.

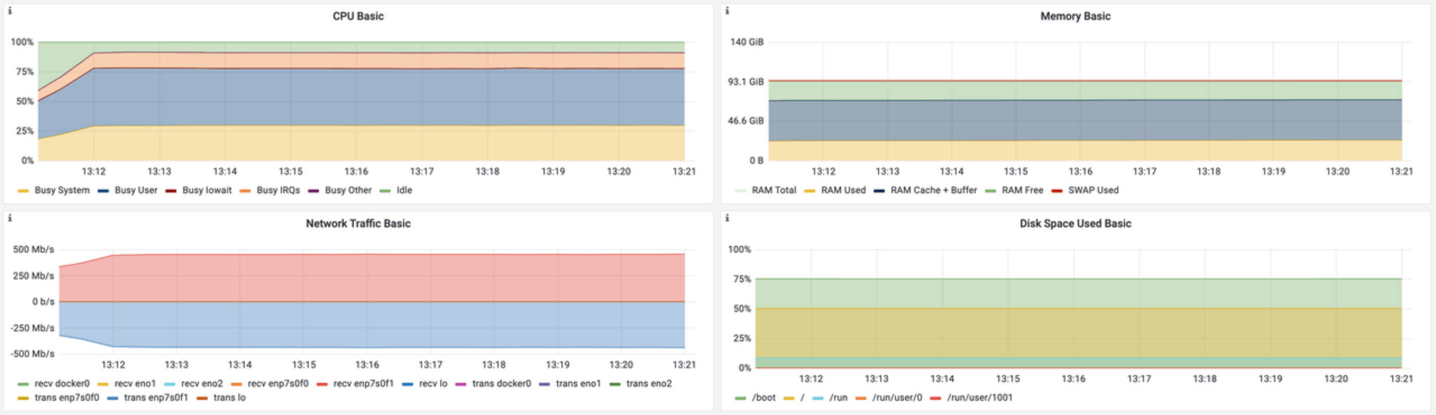
The result is for reference only. Users can carry out a test based on actual business scenarios.

## Monitoring Information

DBPlusEngine-Proxy(No Rules)

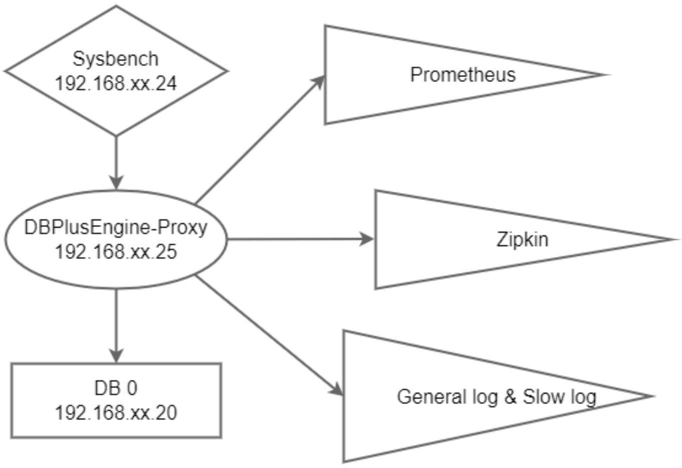


DB Plus Engine-Proxy(shadow / pro)



## 5.3.5 Probe

Probe Test Scenario.



## Test Objective

DBPlusEngine-Proxy can obtain observability through performance loss after integrating with Prometheus/Zipkin/General log & Slow log by plugins, or integrating with other components by plugins. We use the sysbench stress testing tool to perform read/write operations, to obtain the performance of DBPlusEngine-Proxy after the plugin has enabled features.

## Test Tool

Sysbench

## Test Environment

| Application | IP Address | Port | Version |
|---|---|---|---|
| DBPlusEngine-Proxy | 192.168.xx.25 | 3307 | 1.2 |
| DBPlusEngine-Plugin | 192.168.xx.25 | - | 1.2 |
| MySQL | 192.168.xx.20 | 13306 | 8.0.29 |
| Prometheus | - | - | - |
| Zipkin | - | - | - |
| General log & Slow log | - | - | - |

Server Configuration

| Item | Configuration |
|---|---|
| CPU | 48 C |
| Memory | 96 G |
| Hard Disk | SSD_SATA3 820 G |
| JDK | 17.0.2 |

Configure agent.yaml and enable related monitoring.

```yaml
# Copyright © 2022，Beijing Sifei Software Technology Co., LTD.
# All Rights Reserved.
# Unauthorized copying of this file, via any medium is strictly prohibited.
# Proprietary and confidential

plugins:
  logging:
    BaseLogging:
      props:
        slow-query-log: true
        long-query-time: 5000
        general-query-log: true
#  metrics:
#    Prometheus:
#      host: "0.0.0.0"
#      port: 9090
#      props:
#        jvm-information-collector-enabled: "true"
#  tracing:
#    Jaeger:
#      host: "localhost"
#      port: 5775
#      props:
#        service-name: "shardingsphere"
#        jaeger-sampler-type: "const"
#        jaeger-sampler-param: "1"
#    Zipkin:
#      host: "localhost"
#      port: 9411
#      props:
#        service-name: "shardingsphere"
#        url-version: "/api/v2/spans"
#        sampler-type: "const"
#        sampler-param: "1"
#    SkyWalking:
```

```
#    props:
#      opentracing-tracer-class-name: "org.apache.skywalking.apm.toolkit.opentracing.SkywalkingTracer"
#   OpenTelemetry:
#    props:
#      otel-resource-attributes: "service.name=shardingsphere"
#      otel-traces-exporter: "zipkin"
```

Use start-with-agent.sh scripts to start DBPlusEngine-Proxy.

### Test Procedure

1. Create a test database.

```
mysql -utest -h192.168.xx.20 -P13306 -p
```

```
DROP DATABASE IF EXISTS agent_ds;

CREATE DATABASE IF NOT EXISTS agent_ds;
```

Create a logical database.

```
mysql -uroot -h192.168.xx.25 -P3307 -proot
```

```
DROP DATABASE IF EXISTS sphereex_demo;

CREATE DATABASE IF NOT EXISTS sphereex_demo;
```

Associate the data source through DistSQL.

```
USE sphereex_demo;

REGISTER STORAGE UNIT ds_0 (
   HOST="192.168.xx.20",
   PORT=13306,
   DB="agent_ds",
   USER="test",
   PASSWORD="test"
);
```

2. Install test tools.

```
tar -xzvf sysbench-1.0.20.tar.gz -C /usr/local
cd /usr/local/sysbench-1.0.20
./autogen.sh
./configure
make && make install
```

3. Obtain data.

```
sysbench oltp_read_only --mysql-host='192.168.xx.25' --mysql-port=3307 --mysql-user=root --mysql-password='root' --mysql-db=sphereex_demo --tables=10 --table-size=1000000 --report-interval=10 --time=3600 --threads=128 --max-requests=0 --percentile=99 --mysql-ignore-errors="all" --rand-type=uniform --range_selects=off --auto_inc=off cleanup

sysbench oltp_read_only --mysql-host='192.168.xx.25' --mysql-port=3307 --mysql-user=root --mysql-password='root' --mysql-db=sphereex_demo --tables=10 --table-size=1000000 --report-interval=10 --time=3600 --threads=128 --max-requests=0 --percentile=99 --mysql-ignore-errors="all" --rand-type=uniform --range_selects=off --auto_inc=off prepare
```

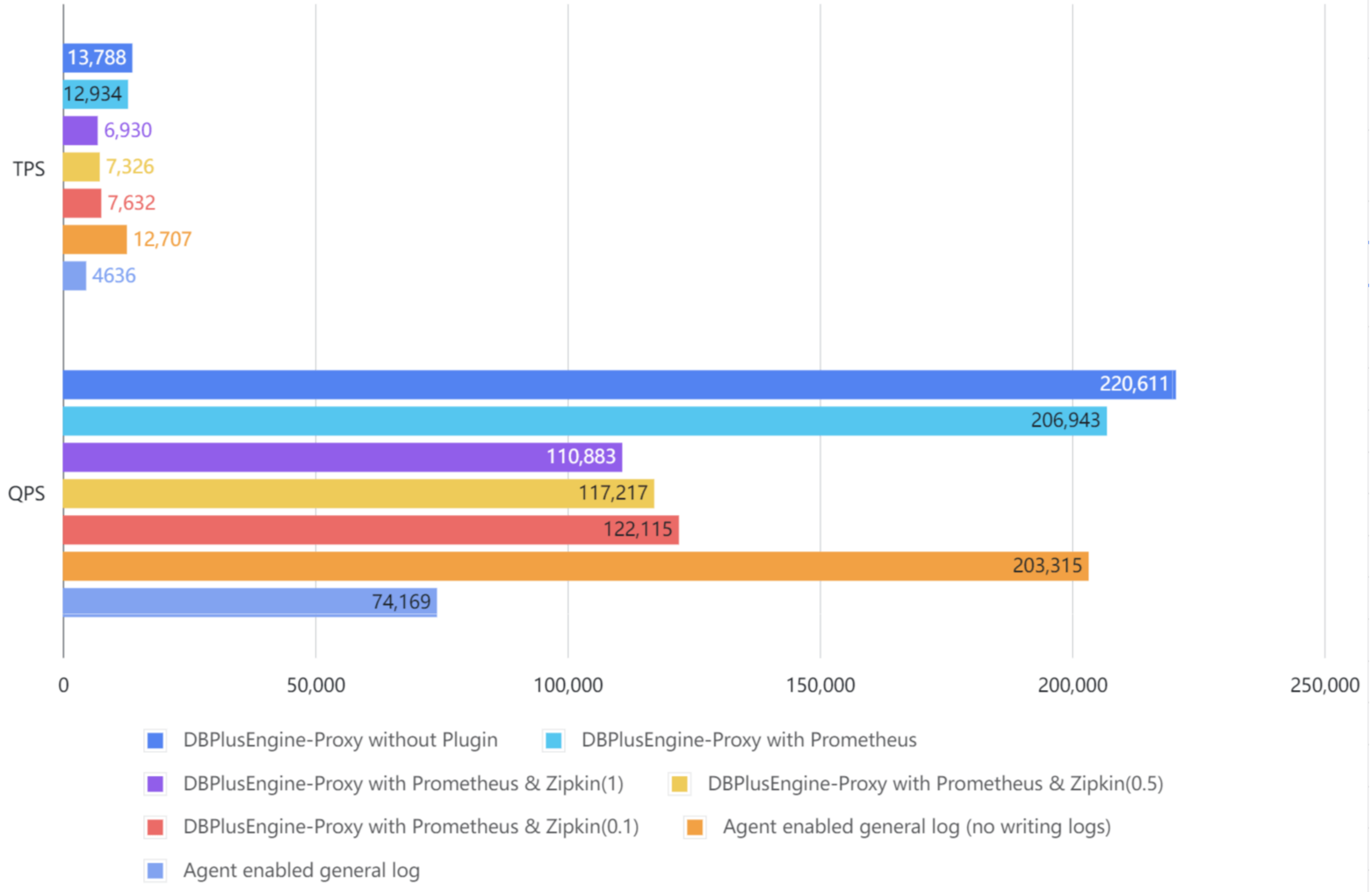
4. Carry out stress testing.

- Read-only Test

```
sysbench oltp_read_write --mysql-host='192.168.xx.25' --mysql-port=3307 --mysql-user=root --mysql-password='root' --mysql-db=sphereex_demo --tables=1 --table-size=1000000 --report-interval=5  --time=180 --threads=200 --max-requests=0  --mysql-ignore-errors="all" --range_selects=off --rand-type=uniform --auto_inc=off run
```

### Test Result

| Application | TPS | QPS |
|---|---|---|
| DBPlusEngine-Proxy without Plugin | 13,788 | 220,611 |
| DBPlusEngine-Proxy with Prometheus | 12,934 | 206,943 |
| DBPlusEngine-Proxy with Prometheus & Zipkin(1) | 6,930 | 110,883 |
| DBPlusEngine-Proxy with Prometheus & Zipkin(0.5) | 7,326 | 117,217 |
| DBPlusEngine-Proxy with Prometheus & Zipkin(0.1) | 7,632 | 122,115 |
| Agent enabled general log (no writing logs) | 12,707 | 203,315 |
| Agent enabled general log | 4,636 | 74,169 |

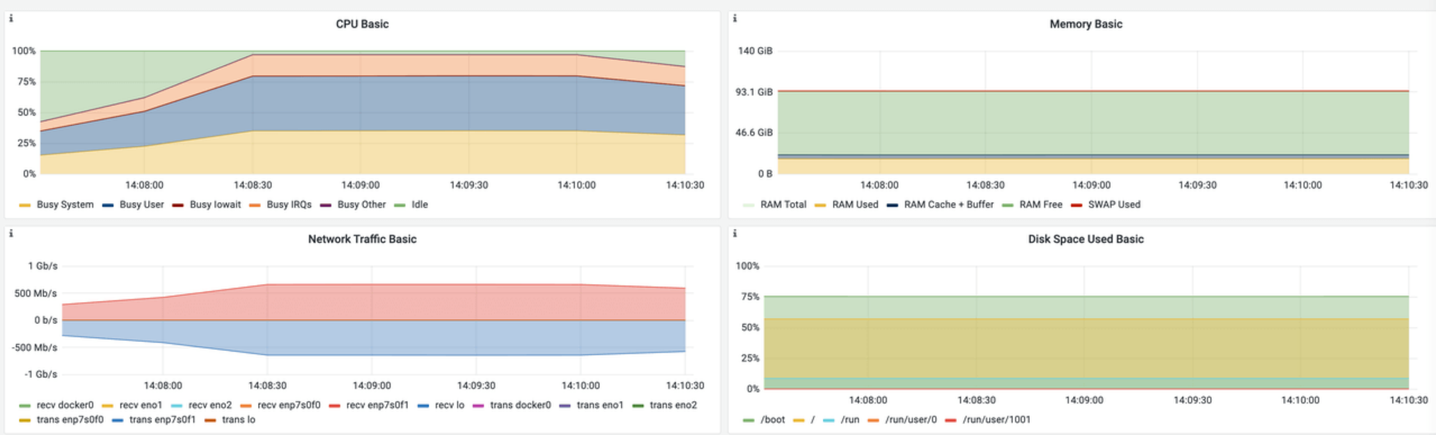


Result: when using a probe to improve observability, the performance slightly decreases after collecting the monitoring indicators. After tracing the performance, it can be seen that the performance decreases notably, which is slightly different facing different sampling rates.

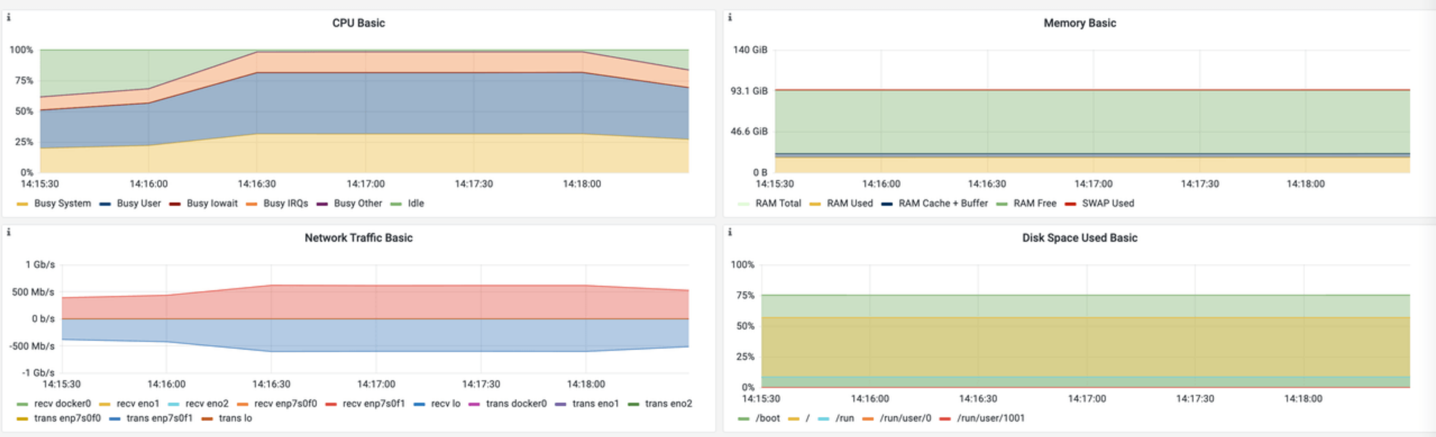
The result is for reference only. Users can carry out a test based on actual business scenarios.

## Monitoring Information

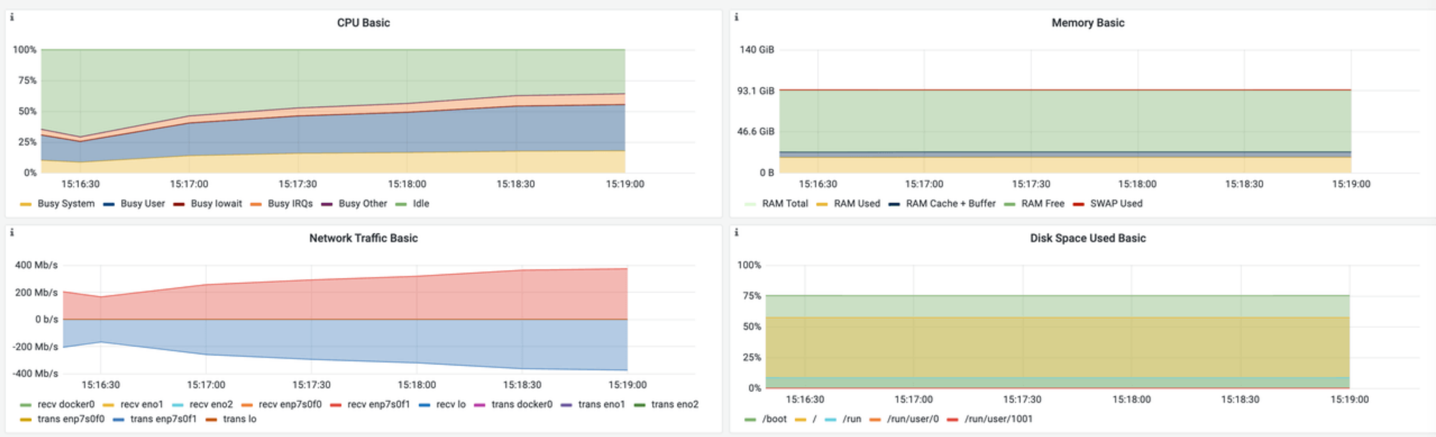
### DBPlusEngine-Proxy without Plugin



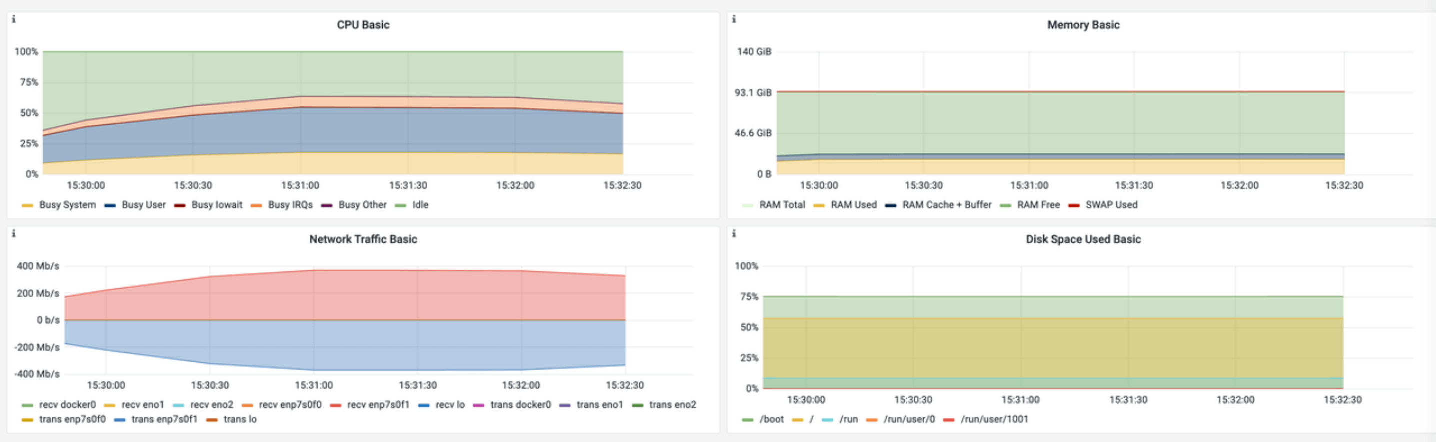
### DBPlusEngine-Proxy with Prometheus



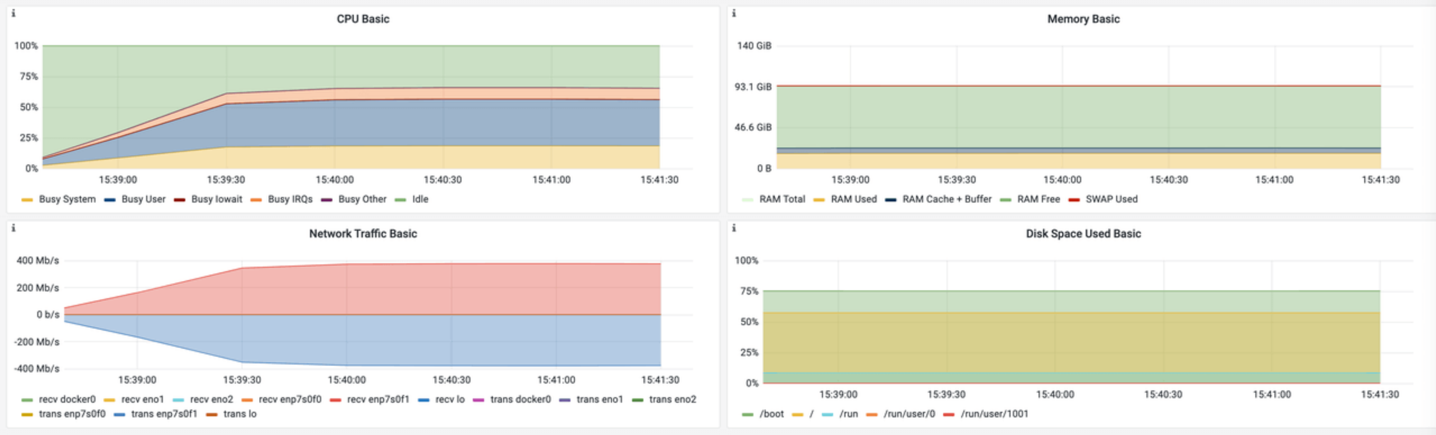
### DBPlusEngine-Proxy with Prometheus & Zipkin(1)



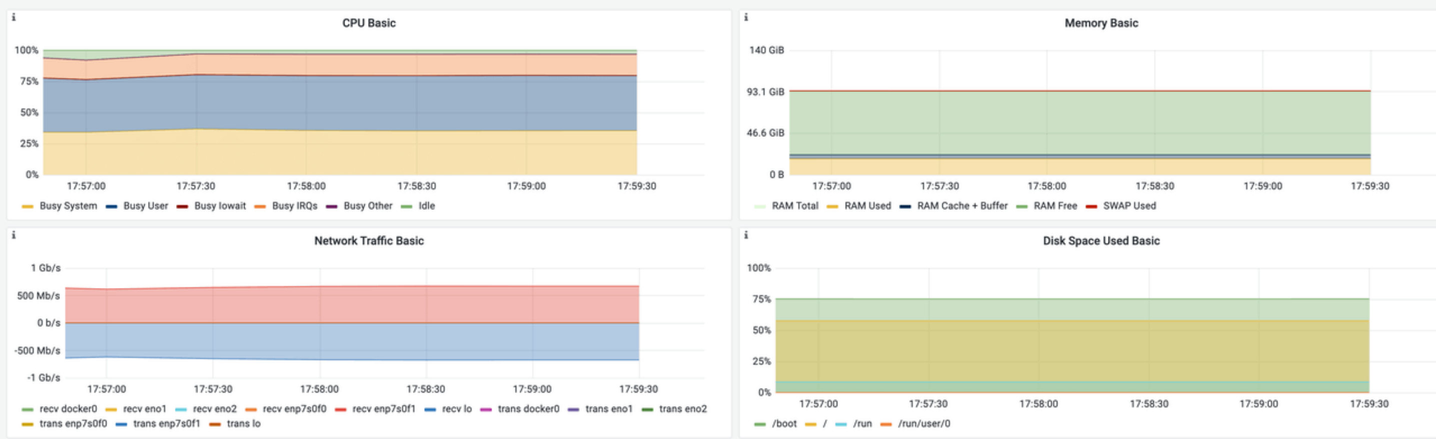
### DBPlusEngine-Proxy with Prometheus & Zipkin(0.5)



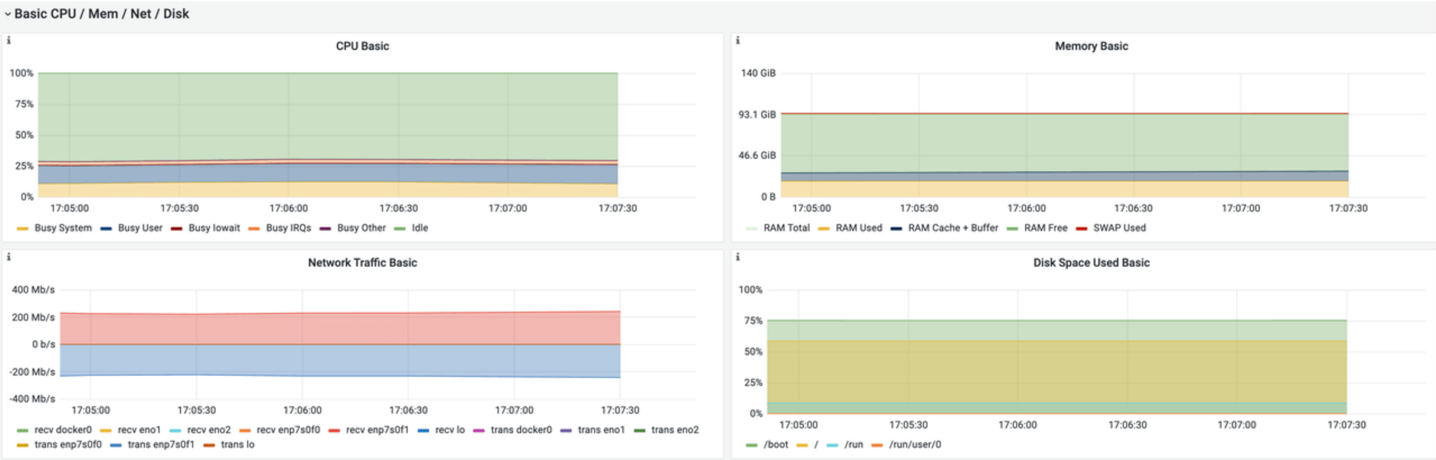
## DBPlusEngine-Proxy with Prometheus & Zipkin(0.1)



## Agent enabled general log (no writing logs)



## Agent enabled general log



## 5.4 Appendix

### 5.4.1 JMH Instructions

```
Usage: java -jar ... [regexp*] [options]
 [opt] means optional argument.
 <opt> means required argument.
 "+" means comma-separated list of values.
 "time" arguments accept time suffixes, like "100ms".

Command line options usually take precedence over annotations.

  [arguments]              Benchmarks to run (regexp+). (default: .*)

  -bm <mode>               Benchmark mode. Available modes are: [Throughput/thrpt,
                           AverageTime/avgt, SampleTime/sample, SingleShotTime/ss,
                           All/all]. (default: Throughput)

  -bs <int>                Batch size: number of benchmark method calls per
                           operation. Some benchmark modes may ignore this
                           setting, please check this separately. (default:
                           1)

  -e <regexp+>             Benchmarks to exclude from the run.

  -f <int>                 How many times to fork a single benchmark. Use 0 to
                           disable forking altogether. Warning: disabling
                           forking may have detrimental impact on benchmark
                           and infrastructure reliability, you might want
                           to use different warmup mode instead. (default:
                           5)

  -foe <bool>              Should JMH fail immediately if any benchmark had
                           experienced an unrecoverable error? This helps
                           to make quick sanity tests for benchmark suites,
                           as well as make the automated runs with checking error
                           codes. (default: false)

  -gc <bool>               Should JMH force GC between iterations? Forcing
                           the GC may help to lower the noise in GC-heavy benchmarks,
                           at the expense of jeopardizing GC ergonomics decisions.
                           Use with care. (default: false)

  -h                       Display help, and exit.
```

```
-i <int>                    Number of measurement iterations to do. Measurement
                            iterations are counted towards the benchmark score.
                            (default: 1 for SingleShotTime, and 5 for all other
                            modes)

-jvm <string>               Use given JVM for runs. This option only affects forked
                            runs.

-jvmArgs <string>           Use given JVM arguments. Most options are inherited
                            from the host VM options, but in some cases you want
                            to pass the options only to a forked VM. Either single
                            space-separated option line, or multiple options
                            are accepted. This option only affects forked runs.

-jvmArgsAppend <string>     Same as jvmArgs, but append these options after the
                            already given JVM args.

-jvmArgsPrepend <string>    Same as jvmArgs, but prepend these options before
                            the already given JVM arg.

-l                          List the benchmarks that match a filter, and exit.

-lp                         List the benchmarks that match a filter, along with
                            parameters, and exit.

-lprof                      List profilers, and exit.

-lrf                        List machine-readable result formats, and exit.

-o <filename>               Redirect human-readable output to a given file.

-opi <int>                  Override operations per invocation, see @OperationsPerInvocation
                            Javadoc for details. (default: 1)

-p <param={v,}*>            Benchmark parameters. This option is expected to
                            be used once per parameter. Parameter name and parameter
                            values should be separated with equals sign. Parameter
                            values should be separated with commas.

-prof <profiler>            Use profilers to collect additional benchmark data.
                            Some profilers are not available on all JVMs and/or
                            all OSes. Please see the list of available profilers
                            with -lprof.

-r <time>                   Minimum time to spend at each measurement iteration.
                            Benchmarks may generally run longer than iteration
                            duration. (default: 10 s)

-rf <type>                  Format type for machine-readable results. These
                            results are written to a separate file (see -rff).
                            See the list of available result formats with -lrf.
                            (default: CSV)

-rff <filename>             Write machine-readable results to a given file.
                            The file format is controlled by -rf option. Please
                            see the list of result formats for available formats.
                            (default: jmh-result.<result-format>)

-si <bool>                  Should JMH synchronize iterations? This would significantly
                            lower the noise in multithreaded tests, by making
                            sure the measured part happens only when all workers
                            are running. (default: true)
```

```
-t <int>                    Number of worker threads to run with. 'max' means
                       the maximum number of hardware threads available
                       on the machine, figured out by JMH itself. (default:
                       1)

-tg <int+>                  Override thread group distribution for asymmetric
                       benchmarks. This option expects a comma-separated
                       list of thread counts within the group. See @Group/@GroupThreads
                       Javadoc for more information.

-to <time>                  Timeout for benchmark iteration. After reaching
                       this timeout, JMH will try to interrupt the running
                       tasks. Non-cooperating benchmarks may ignore this
                       timeout. (default: 10 min)

-tu <TU>                    Override time unit in benchmark results. Available
                       time units are: [m, s, ms, us, ns]. (default: SECONDS)

-v <mode>                   Verbosity mode. Available modes are: [SILENT, NORMAL,
                       EXTRA]. (default: NORMAL)

-w <time>                   Minimum time to spend at each warmup iteration. Benchmarks
                       may generally run longer than iteration duration.
                       (default: 10 s)

-wbs <int>                  Warmup batch size: number of benchmark method calls
                       per operation. Some benchmark modes may ignore this
                       setting. (default: 1)

-wf <int>                   How many warmup forks to make for a single benchmark.
                       All iterations within the warmup fork are not counted
                       towards the benchmark score. Use 0 to disable warmup
                       forks. (default: 0)

-wi <int>                   Number of warmup iterations to do. Warmup iterations
                       are not counted towards the benchmark score. (default:
                       0 for SingleShotTime, and 5 for all other modes)

-wm <mode>                  Warmup mode for warming up selected benchmarks.
                       Warmup modes are: INDI = Warmup each benchmark individually,
                       then measure it. BULK = Warmup all benchmarks first,
                       then do all the measurements. BULK_INDI = Warmup
                       all benchmarks first, then re-warmup each benchmark
                       individually, then measure it. (default: INDI)

-wmb <regexp+>              Warmup benchmarks to include in the run in addition
                       to already selected by the primary filters. Harness
                       will not measure these benchmarks, but only use them
                       for the warmup.
```

# 6 Quick Start

This chapter provides users with the simplest and quickest way to get started with DBPlusEngine.

## 6.1 DBPlusEngine-Driver

### 6.1.1 Scenarios

DBPlusEngine-Driver can be configured by four methods, Java, YAML, Spring namespace and Spring boot starter. Developers can choose the most suitable method according to different situations.

### 6.1.2 Restrictions

Currently only Java language is supported.

### 6.1.3 Prerequisites

The development environment requires Java JRE 8 or later.

### 6.1.4 Procedure

1. Rules configuration.

Please refer to User Manual for more details.

2. Import Maven dependency.

```
<dependency>
   <groupId>org.apache.shardingsphere</groupId>
   <artifactId>shardingsphere-jdbc-core-spring-boot-starter</artifactId>
   <version>${latest.release.version}</version>
</dependency>
```

Notice: Please change ${latest.release.version} to the actual version.

3. Edit application.yml.

```
spring:
  shardingsphere:
    datasource:
      names: ds_0, ds_1
      ds_0:
        type: com.zaxxer.hikari.HikariDataSource
        driverClassName: com.mysql.cj.jdbc.Driver
        jdbcUrl: jdbc:mysql://localhost:3306/demo_ds_0?serverTimezone=UTC&useSSL=false&useUnicode=true&characterEncoding=UTF-8
        username: root
        password:
      ds_1:
        type: com.zaxxer.hikari.HikariDataSource
        driverClassName: com.mysql.cj.jdbc.Driver
        jdbcUrl: jdbc:mysql://localhost:3306/demo_ds_1?serverTimezone=UTC&useSSL=false&useUnicode=true&characterEncoding=UTF-8
        username: root
        password:
    rules:
      sharding:
        tables:
          ...
```

## 6.2 DBPlusEngine-Proxy

### 6.2.1 Application Scenarios

DBPlusEngine-Proxy is positioned as a transparent database proxy. Any client that uses MySQL, PostgreSQL, and openGauss protocols are supported to operate data, which is more friendly to heterogeneous languages and operation & maintenance scenarios.

### 6.2.2 Limitations

DBPlusEngine-Proxy has limited support for system databases or tables (such as information_schema and pg_catalog).

When connecting to Proxy through graphical database clients, an error may be displayed in the clients or Proxy. You can use command-line clients (such as mysql, psql, and gsql) to connect to the Proxy authentication function.

### 6.2.3 Prerequisite

Additional dependencies are not required when using Docker to start DBPlusEngine-Proxy. When it comes to using binary distribution, Java JRE 8 or later versions are required for the environment.

### 6.2.4 Procedure

1. Get DBPlusEngine-Proxy

Currently, DBPlusEngine-Proxy can be obtained through the following methods:

- Using Binary Distribution
- Using Docker
- Using Operator

2. Rules Configuration

Edit %SHARDINGSPHERE_PROXY_HOME%/conf/server.yaml.

Edit %SHARDINGSPHERE_PROXY_HOME%/conf/config-xxx.yaml.

> %SHARDINGSPHERE_PROXY_HOME% is the path after the Proxy is decompressed. For example, /opt/shardingsphere-proxy-bin/

Please refer to the Configuration Manual for more details.

3. Import Dependencies

If the backend database is PostgreSQL or Oracle, there’ s no need for additional dependencies.

If the backend database is MySQL, please download: mysql-connector-java-5.1.47.jar or mysql-connector-java-8.0.11.jar, and put it into the %SHARDINGSPHERE_PROXY_HOME%/ext-lib directory.

4. Start the server

- Use default configuration items.

```
sh %SHARDINGSPHERE_PROXY_HOME%/bin/start.sh
```

The default port is 3307, and the default configuration file directory is %SHARDINGSPHERE_PROXY_HOME%/conf/.

- Customize port and configure file directory.

```
sh %SHARDINGSPHERE_PROXY_HOME%/bin/start.sh ${proxy_port} ${proxy_conf_directory}
```

- Mandatory Startup.

```
sh %SHARDINGSPHERE_PROXY_HOME%/bin/start.sh -f
```

-f parameter is used to forcibly enable Proxy and it will ignore abnormal data sources during startup. Users can remove abnormal data sources through DistSQL after the Proxy is started.

5. Use DBPlusEngine-Proxy

Execute MySQL, PostgreSQL, or openGauss client command to run DBPlusEngine-Proxy.

```
mysql -h${proxy_host} -P${proxy_port} -u${proxy_username} -p${proxy_password}
```

Use PostgreSQL client to connect to DBPlusEngine-Proxy:

```
psql -h ${proxy_host} -p ${proxy_port} -U ${proxy_username}
```

Use openGauss client to connect to DBPlusEngine-Proxy:

```
gsql -r -h ${proxy_host} -p ${proxy_port} -U ${proxy_username} -W ${proxy_password}
```

# 7 User Manual

This chapter describes how to use components of DBPlusEngine.

## 7.1 DBPlusEngine-Driver

Configuration is the only module in DBPlusEngine-JDBC that interacts with application developers, through which developers can quickly and clearly understand the functions provided by DBPlusEngine-JDBC.

This chapter is a configuration manual for DBPlusEngine-JDBC, which can also be referred to as a dictionary if necessary.

DBPlusEngine-JDBC has provided 4 kinds of configuration methods for different situations. By configuration, application developers can flexibly use data sharding, readwrite-splitting, data encryption, shadow database or the combination of them.

Mixed rule configurations are very similar to single rule configuration, except for the differences from single rule to multiple rules.

It should be noted that the superposition between rules are data source and table name related. If the previous rule is data source oriented aggregation, the next rule needs to use the aggregated logical data source name configured by the previous rule when configuring the data source; Similarly, if the previous rule is table oriented aggregation, the next rule needs to use the aggregated logical table name configured by the previous rule when configuring the table.

### 7.1.1 Use of License in DBPlusEngine-Driver

#### Background

| Function of License | Description |
|---|---|
| limit duration | Limit the duration of DBPlusEngine. |
| limit resources | Limit the number of storage nodes created by the DBPlusEngine. |
| limit instances | Limit the number of instances created by [DBPlusEngine-Proxy]+[DBPlusEngine-Driver] in a single cluster. |
| limit versions | Limit the version of the DBPlusEngine instance in a stand-alone or cluster. |

## Configuration Method

### JAVA API

### Import Maven Dependency

```
<dependency>
  <groupId>com.sphere-ex</groupId>
  <artifactId>shardingsphere-jdbc-core</artifactId>
  <version>${dbplusengine.version}</version>
</dependency>
```

### Configuration Example

Class name: com.spherex.dbplusengine.license.config.LicenseRuleConfiguration

Attributes:

| Name | DataType | Description |
|---|---|---|
| license | String | license registration code |

### YAML

### Import Maven Dependency

```
<dependency>
  <groupId>com.sphere-ex</groupId>
  <artifactId>shardingsphere-jdbc-core</artifactId>
  <version>${dbplusengine.version}</version>
</dependency>
```

### Configuration Example

```
databaseName:

mode:

dataSources:

rules:

license: xxx
```

## SpringBoot

### Import Maven Dependency

```
<dependency>
  <groupId>com.sphere-ex</groupId>
  <artifactId>shardingsphere-jdbc-core-spring-boot-starter</artifactId>
  <version>${dbplusengine.version}</version>
</dependency>
```

### Configuration Example

```
spring.dbplusengine.rules.license: xxx
```

## Spring Name Space

### Import Maven Dependency

```
<dependency>
  <groupId>com.sphere-ex</groupId>
  <artifactId>shardingsphere-jdbc-core-spring-namespace</artifactId>
  <version>${dbplusengine.version}</version>
</dependency>
```

### Configuration Example

```
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:license="https://www.sphere-ex.com/schema/dbplusengine/license"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
               http://www.springframework.org/schema/beans/spring-beans.xsd
               https://www.sphere-ex.com/schema/dbplusengine/license
               https://www.sphere-ex.com/schema/dbplusengine/license/license.xsd
               ">
  <license:license id="license" license="xxx"/>
</beans>
```

## Post-processing

If the following error occurs, please obtain the latest license and update the license.

- No license prompt

```
FATAL:  License not registered
```

- Incomplete license prompt

```
FATAL:  Incomplete license
```

- License expiration prompt

```
FATAL:  License is expired
```

- Illegal license prompt

```
ERROR:  The xxx exceeds the limit of the license
```

## 7.1.2 Java API

### Overview

Java API is the basic configuration methods in DBPlusEngine-Driver, and other configurations will eventually be transformed into Java API configuration methods.

The Java API is the most complex and flexible configuration method, which is suitable for the scenarios requiring dynamic configuration through programming.

### Usage

### Import Maven Dependency

```
<dependency>
  <groupId>org.apache.shardingsphere</groupId>
  <artifactId>shardingsphere-jdbc-core</artifactId>
  <version>${shardingsphere.version}</version>
</dependency>
```

### Create Data Source

DBPlusEngine-Driver Java API consists of schema name, mode configuration, data source map, rule configurations and properties.

The ShardingSphereDataSource created by ShardingSphereDataSourceFactory implements the standard JDBC DataSource interface.

```
String schemaName = "foo_schema"; // Indicate logic schema name
ModeConfiguration modeConfig = ... // Build mode configuration
Map<String, DataSource> dataSourceMap = ... // Build actual data sources
Collection<RuleConfiguration> ruleConfigs = ... // Build concentrate rule configurations
Properties props = ... // Build properties
DataSource dataSource = ShardingSphereDataSourceFactory.createDataSource(schemaName, modeConfig, dataSourceMap,
ruleConfigs, props);
```

Please refer to Mode Confiugration for more mode details.

Please refer to Data Source Confiugration for more data source details.

Please refer to Rules Confiugration for more rule details.

### Use Data Source

Developer can choose to use native JDBC or ORM frameworks such as JPA, Hibernate or MyBatis through the DataSource.

Take native JDBC usage as an example:

```
// Create ShardingSphereDataSource
DataSource dataSource = ShardingSphereDataSourceFactory.createDataSource(schemaName, modeConfig, dataSourceMap,
ruleConfigs, props);

String sql = "SELECT i.* FROM t_order o JOIN t_order_item i ON o.order_id=i.order_id WHERE o.user_id=? AND o.order_id=?";
```

```
try (
    Connection conn = dataSource.getConnection();
    PreparedStatement ps = conn.prepareStatement(sql)) {
  ps.setInt(1, 10);
  ps.setInt(2, 1000);
  try (ResultSet rs = preparedStatement.executeQuery()) {
    while(rs.next()) {
      // ...
    }
  }
}
```

## Mode Configuration

### Root Configuration

Class name: org.apache.shardingsphere.infra.config.mode.ModeConfiguration

Attributes:

| Name | DataType | Description | DefaultValue |
|---|---|---|---|
| type | String | Type of mode configurationValues could be: Cluster | Memory |
| repository | PersistRepositoryConfiguration | Persist repository configurationMemory type does not need persist, could be nullStandalone type uses StandalonePersistRepositoryConfigurationCluster type uses ClusterPersistRepositoryConfiguration | |
| overwrite | boolean | Whether overwrite persistent configuration with local configuration | false |

### Cluster Persist Configuration

Class name: org.apache.shardingsphere.mode.repository.cluster.ClusterPersistRepositoryConfiguration

Attributes:

| Name | DataType | Description |
|---|---|---|
| type | String | Type of persist repository |
| namespace | String | Namespace of registry center |
| serverLists | String | Server lists of registry center |
| props | Properties | Properties of persist repository |

Please refer to Builtin Persist Repository List for more details about type of repository.

### Data Source

DBPlusEngine-Driver Supports all JDBC drivers and database connection pools.

### Example

In this example, the database driver is MySQL, and connection pool is HikariCP, which can be replaced with other database drivers and connection pools.

```
Map<String, DataSource> dataSourceMap = new HashMap<>();

// Configure the 1st data source
HikariDataSource dataSource1 = new HikariDataSource();
dataSource1.setDriverClassName("com.mysql.jdbc.Driver");
dataSource1.setJdbcUrl("jdbc:mysql://localhost:3306/ds_1");
dataSource1.setUsername("root");
dataSource1.setPassword("");
dataSourceMap.put("ds_1", dataSource1);

// Configure the 2nd data source
HikariDataSource dataSource2 = new HikariDataSource();
dataSource2.setDriverClassName("com.mysql.jdbc.Driver");
dataSource2.setJdbcUrl("jdbc:mysql://localhost:3306/ds_2");
dataSource2.setUsername("root");
dataSource2.setPassword("");
dataSourceMap.put("ds_2", dataSource2);

// Configure other data sources
...
```

### Rules

Rules are pluggable part of DBPlusEngine. This chapter is a java rule configuration manual for DBPlusEngine-Driver.

### Sharding

### Root Configuration

Class name: org.apache.shardingsphere.sharding.api.config.ShardingRuleConfiguration

Attributes:

| Name | DataType | Description | Def ault V alue |
|---|---|---|---|
| tables (+) | Collection<ShardingTableRuleConfiguration> | Sharding table rules | - |
| autoTables (+) | Coll ection<ShardingAutoTableRuleConfiguration> | Sharding automatic table rules | - |
| bind ingTableGroups (*) | Collection<String> | Binding table rules | E mpty |
| b roadcastTables (*) | Collection<String> | Broadcast table rules | E mpty |
| def aultDatabaseSh ardingStrategy (?) | Sharding StrategyConfiguration | Default database sharding strategy | Not shar ding |
| defaultTableSh ardingStrategy (?) | Sharding StrategyConfiguration | Default table sharding strategy | Not shar ding |
| defaultKeyGe nerateStrategy (?) | KeyG eneratorConfiguration | Default key generator | S nowf lake |
| default ShardingColumn (?) | String | Default sharding column name | None |
| shar dingAlgorithms (+) | Map<String, ShardingSphereAl gorithmConfiguration> | Sharding algorithm name and configurations | None |
| keyGenerators (?) | Map<String, ShardingSphereAl gorithmConfiguration> | Key generate algorithm name and configurations | None |

### Sharding Table Configuration

Class name: org.apache.shardingsphere.sharding.api.config.ShardingTableRuleConfiguration

Attributes:

| Name | Da taType | Description | Default Value |
|---|---|---|---|
| logic Table | String | Name of sharding logic table | - |
| actua lData Nodes (?) | String | Describe data source names and actual tables, delimiter as point. Multiple data nodes split by comma, support inline expression | Broadcast table or databases sharding only |
| data baseS hardi ngStr ategy (?) | Shard ingStr ategyC onfigu ration | Databases sharding strategy | Use default databases sharding strategy |
| t ableS hardi ngStr ategy (?) | Shard ingStr ategyC onfigu ration | Tables sharding strategy | Use default tables sharding strategy |
| keyG enera teStr ategy (?) | K eyGene ratorC onfigu ration | Key generator configuration | Use default key generator |

### Sharding Automatic Table Configuration

Class name: org.apache.shardingsphere.sharding.api.config.ShardingAutoTableRuleConfiguration

Attributes:

| Name | DataType | Description | Default Value |
|---|---|---|---|
| lo gicTable | String | Name of sharding logic table | - |
| a ctualDat aSources (?) | String | Data source names. Multiple data nodes split by comma | Use all configured data sources |
| sharding Strategy (?) | Shardin gStrategyCo nfiguration | Sharding strategy | Use default sharding strategy |
| key Generate Strategy (?) | Key GeneratorCo nfiguration | Key generator configuration | Use default key generator |

### Sharding Strategy Configuration

#### Standard Sharding Strategy Configuration

Class name: org.apache.shardingsphere.sharding.api.config.strategy.sharding.StandardShardingStrategyConfiguration

Attributes:

| Name | DataType | Description |
|---|---|---|
| shardingColumn | String | Sharding column name |
| shardingAlgorithmName | String | Sharding algorithm name |

#### Complex Sharding Strategy Configuration

Class name: org.apache.shardingsphere.sharding.api.config.strategy.sharding.ComplexShardingStrategyConfiguration

Attributes:

| Name | DataType | Description |
|---|---|---|
| shardingColumns | String | Sharding column name, separated by commas |
| shardingAlgorithmName | String | Sharding algorithm name |

#### Hint Sharding Strategy Configuration

Class name: org.apache.shardingsphere.sharding.api.config.strategy.sharding.HintShardingStrategyConfiguration

Attributes:

| Name | DataType | Description |
|---|---|---|
| shardingAlgorithmName | String | Sharding algorithm name |

### None Sharding Strategy Configuration

Class name: org.apache.shardingsphere.sharding.api.config.strategy.sharding.NoneShardingStrategyConfiguration

Attributes: None

Please refer to Built-in Sharding Algorithm List for more details about type of algorithm.

### Key Generate Strategy Configuration

Class name: org.apache.shardingsphere.sharding.api.config.strategy.keygen.KeyGenerateStrategyConfiguration

Attributes:

| Name | DataType | Description |
|---|---|---|
| column | String | Column name of key generate |
| keyGeneratorName | String | key generate algorithm name |

Please refer to Built-in Key Generate Algorithm List for more details about type of algorithm.

## Readwrite-splitting

### Root Configuration

Class name: org.apache.shardingsphere.readwritesplitting.api.ReadwriteSplittingRuleConfiguration

Attributes:

| Name | DataType | Description |
|---|---|---|
| d ataSo urces (+) | Collectio n<ReadwriteSplittingData SourceRuleConfiguration> | Data sources of write and reads |
| loa dBala ncers (*) | Map<String, ShardingSpher eAlgorithmCon- figuration> | Load balance algorithm name and configurations of replica data sources |

### Readwrite-splitting Data Source Configuration

Class name: org.apache.shardingsphere.readwritesplitting.api.rule.ReadwriteSplittingDataSourceRuleConfiguration

Configurable Properties:

| Name | Dat aTy pe | Description | Default Value |
|---|---|---|---|
| name | Str ing | Readwrite-splitting data source name | - |
| stat icStrategy | Str ing | Static Readwrite-splitting configuration | - |
| dynam icStrategy | P rop ert ies | Dynamic Readwrite-splitting configuration | - |
| loadBa lancerName (?) | Str ing | Load balance algorithm name of replica sources | Round robin load balance algorithm |

Class name：org.apache.shardingsphere.readwritesplitting.api.strategy.StaticReadwriteSplittingStrategyConfiguration

Configurable Properties:

| Name | DataType | Description |
|---|---|---|
| writeDataSourceName | String | Write data source name |
| readDataSourceNames | List<String> | Read data sources list |

Class name：org.apache.shardingsphere.readwritesplitting.api.strategy.DynamicReadwriteSplittingStrategyConfiguration

Configurable Properties:

| Name | DataType | Description | Default Value |
|---|---|---|---|
| autoAwareDataSourceName | String | Database discovery logic data source name | - |
| writeDataSourceQueryEnabled (?) | String | All read data source are offline, write data source whether the data source is responsible for read traffic | true |

Please refer to Built-in Load Balance Algorithm List for more details about type of algorithm. Please refer to Use Norms for more details about query consistent routing.

### HA

#### Root Configuration

Class name： org.apache.shardingsphere.dbdiscovery.api.config.DatabaseDiscoveryRuleConfiguration

Attributes：

| Name | DataType | Description |
|---|---|---|
| dataSources (+) | Collection<DatabaseDiscoveryDataSourceRuleConfiguration> | Data source configuration |
| discoveryHeartbeats (+) | Map<String, DatabaseDiscoveryHeartBeatConfiguration> | Detect heartbeat configuration |
| discoveryTypes (+) | Map<String, ShardingSphereAlgorithmConfiguration> | Database discovery type configuration |

#### Data Source Configuration

Class name：org.apache.shardingsphere.dbdiscovery.api.config.rule.DatabaseDiscoveryDataSourceRuleConfiguration

Attributes：

| Name | DataType | Description | Default Value |
|---|---|---|---|
| groupName (+) | String | Database discovery group name | - |
| dataSourceNames (+) | Collection<String> | Data source names, multiple data source names separated with comma. Such as: ds_0, ds_1 | - |
| discoveryHeartbeatName (+) | String | Detect heartbeat name | - |
| discoveryTypeName (+) | String | Database discovery type name | - |

### Detect Heartbeat Configuration

Class name: org.apache.shardingsphere.dbdiscovery.api.config.rule.DatabaseDiscoveryHeartBeatConfiguration

Attributes:

| Name | DataType | Description | Default Value |
|---|---|---|---|
| props (+) | Properties | Detect heartbeat attribute configuration, keep-alive-cron configuration, cron expression. Such as: '0/5 * * * * ?' | - |

### Database Discovery Type Configuration

Class name: org.apache.shardingsphere.infra.config.algorithm.ShardingSphereAlgorithmConfiguration

Attributes:

| Name | DataType | Description | Default Value |
|---|---|---|---|
| type (+) | String | Database discovery type, such as: MGR、openGauss | - |
| props (?) | Properties | Required parameters for high-availability types, such as MGR's group-name | - |

## Encryption

### Configuration Entry

Class name: org.apache.shardingsphere.encrypt.api.config.EncryptRuleConfiguration

Attributes:

| Name | DataType | Description | Default Value |
|---|---|---|---|
| tables (+) | Collection<EncryptTableRuleConfiguration> | Encrypt table rule configurations | |
| encryptors (+) | Map<String, ShardingSphereAlgorithmConfiguration> | Encrypt algorithm name and configurations | |
| queryWithCipherColumn (?) | boolean | Whether query with cipher column for data encrypt. User you can use plaintext to query if have | true |

### Encrypt Table Rule Configuration

Class name: org.apache.shardingsphere.encrypt.api.config.rule.EncryptTableRuleConfiguration

Attributes:

| Name | DataType | Description |
|---|---|---|
| name | String | Table name |
| columns (+) | Collection <EncryptColumnRuleConfiguration> | Encrypt column rule configurations |
| queryWithCipherColumn (?) | boolean | The current table whether query with cipher column for data encrypt. |

### Encrypt Column Rule Configuration

Class name: org.apache.shardingsphere.encrypt.api.config.rule.EncryptColumnRuleConfiguration

Attributes:

| Name | DataType | Description |
|---|---|---|
| logicColumn | String | Required field. Logic column name. |
| dataType | String | Logical column type, optional (required if other encrypted column, auxiliary query column or plain text column types are configured) |
| cipherColumn | String | Required field. Cipher column name. |
| cipherDataType | String | Encrypted column type, optional (required if logical column type is configured) |
| assistedQueryColumn | String | Auxiliary query column name, optional |
| assistedQueryDataType | String | Auxiliary query column type, optional (required if auxiliary query column and logical column types are configured) |
| plainColumn | String | Name in plain text, optional |
| plainDataType | String | Plain text column type, optional (required if plain text column and logical column types are configured) |
| encryptorName | String | Encryption algorithm name, required |

### Encrypt Algorithm Configuration

Class name: org.apache.shardingsphere.infra.config.algorithm.ShardingSphereAlgorithmConfiguration

Attributes:

| Name | DataType | Description |
|---|---|---|
| name | String | Encrypt algorithm name |
| type | String | Encrypt algorithm type |
| properties | Properties | Encrypt algorithm properties |

Please refer to Built-in Encrypt Algorithm List for more details about type of algorithm.

### Shadow DB

#### Root Configuration

Class name: org.apache.shardingsphere.shadow.api.config.ShadowRuleConfiguration

Attributes:

| Name | DataType | Description | Default Value |
|---|---|---|---|
| dataSources | Map<String, ShadowDataSourceConfiguration> | Shadow data source mapping name and configuration | |
| tables | Map<String, ShadowTableConfiguration> | Shadow table name and configuration | |
| defaultShadowAlgorithmName | String | Default shadow algorithm name | |
| shadowAlgorithms | Map<String, ShardingSphere AlgorithmConfiguration> | Shadow algorithm name and configuration | |

#### Shadow Data Source Configuration

Class name: org.apache.shardingsphere.shadow.api.config.datasource.ShadowDataSourceConfiguration

Attributes:

| Name | DataType | Description |
|---|---|---|
| sourceDataSourceName | String | Production data source name |
| shadowDataSourceName | String | Shadow data source name |

#### Shadow Table Configuration

Class name: org.apache.shardingsphere.shadow.api.config.table.ShadowTableConfiguration

Attributes:

| Name | DataType | Description |
|---|---|---|
| dataSourceNames | Collection<String> | Shadow table location shadow data source mapping names |
| shadowAlgorithmNames | Collection<String> | Shadow table location shadow algorithm names |

#### Shadow Algorithm Configuration

Please refer to Built-in Shadow Algorithm List.

## SQL Parser

### Root Configuration

Class: org.apache.shardingsphere.parser.config.SQLParserRuleConfiguration

Attributes:

| name | DataType | Description |
|---|---|---|
| sqlCommentParseEnabled (?) | boolean | Whether to parse SQL comments |
| parseTreeCache (?) | CacheOption | Parse syntax tree local cache configuration |
| sqlStatementCache (?) | CacheOption | sql statement local cache configuration |

### Cache option Configuration

Class: org.apache.shardingsphere.sql.parser.api.CacheOption

Attributes:

| name | DataType | Description | Default Value |
|---|---|---|---|
| initialCapacity | int | Initial capacity of local cache | parser syntax tree local cache default value 128, SQL statement cache default value 2000 |
| maximumSize(?) | long | Maximum capacity of local cache | The default value of local cache for parsing syntax tree is 1024, and the default value of sql statement cache is 65535 |
| concurrencyLevel | int | Local cache concurrency level, the maximum number of concurrent updates allowed by threads | 4 |

## Mixed Rules

### Configuration Item Explanation

```
/* Data source configuration */
HikariDataSource writeDataSource0 = new HikariDataSource();
writeDataSource0.setDriverClassName("com.mysql.jdbc.Driver");
writeDataSource0.setJdbcUrl("jdbc:mysql://localhost:3306/db0?serverTimezone=UTC&useSSL=false&useUnicode=true&
characterEncoding=UTF-8");
writeDataSource0.setUsername("root");
writeDataSource0.setPassword("");

HikariDataSource writeDataSource1 = new HikariDataSource();
// ...Omit specific configuration.

HikariDataSource read0OfwriteDataSource0 = new HikariDataSource();
// ...Omit specific configuration.

HikariDataSource read1OfwriteDataSource0 = new HikariDataSource();
// ...Omit specific configuration.

HikariDataSource read0OfwriteDataSource1 = new HikariDataSource();
// ...Omit specific configuration.

HikariDataSource read1OfwriteDataSource1 = new HikariDataSource();
```

```
// ...Omit specific configuration.

Map<String, DataSource> datasourceMaps = new HashMap<>(6);

datasourceMaps.put("write_ds0", writeDataSource0);
datasourceMaps.put("write_ds0_read0", read0OfwriteDataSource0);
datasourceMaps.put("write_ds0_read1", read1OfwriteDataSource0);

datasourceMaps.put("write_ds1", writeDataSource1);
datasourceMaps.put("write_ds1_read0", read0OfwriteDataSource1);
datasourceMaps.put("write_ds1_read1", read1OfwriteDataSource1);

/* Sharding rule configuration */
// The enumeration value of `ds_$->{0..1}` is the name of the logical data source configured with read-query
ShardingTableRuleConfiguration tOrderRuleConfiguration = new ShardingTableRuleConfiguration("t_order", "ds_${0..1}.t_order_${[0, 1]}");
tOrderRuleConfiguration.setKeyGenerateStrategy(new KeyGenerateStrategyConfiguration("order_id", "snowflake"));
tOrderRuleConfiguration.setTableShardingStrategy(new StandardShardingStrategyConfiguration("order_id", "tOrderInlineShardingAlgorithm"));
Properties tOrderShardingInlineProps = new Properties();
tOrderShardingInlineProps.setProperty("algorithm-expression", "t_order_${order_id % 2}");
tOrderRuleConfiguration.getShardingAlgorithms().putIfAbsent("tOrderInlineShardingAlgorithm", new ShardingSphereAlgorithmConfiguration("INLINE",tOrderShardingInlineProps));

ShardingTableRuleConfiguration tOrderItemRuleConfiguration = new ShardingTableRuleConfiguration("t_order_item", "ds_${0..1}.t_order_item_${[0, 1]}");
tOrderItemRuleConfiguration.setKeyGenerateStrategy(new KeyGenerateStrategyConfiguration("order_item_id", "snowflake"));
tOrderRuleConfiguration.setTableShardingStrategy(new StandardShardingStrategyConfiguration("order_item_id", "tOrderItemInlineShardingAlgorithm"));
Properties tOrderItemShardingInlineProps = new Properties();
tOrderItemShardingInlineProps.setProperty("algorithm-expression", "t_order_item_${order_item_id % 2}");
tOrderRuleConfiguration.getShardingAlgorithms().putIfAbsent("tOrderItemInlineShardingAlgorithm", new ShardingSphereAlgorithmConfiguration("INLINE",tOrderItemShardingInlineProps));

ShardingRuleConfiguration shardingRuleConfiguration = new ShardingRuleConfiguration();
shardingRuleConfiguration.getTables().add(tOrderRuleConfiguration);
shardingRuleConfiguration.getTables().add(tOrderItemRuleConfiguration);
shardingRuleConfiguration.getBindingTableGroups().add("t_order, t_order_item");
shardingRuleConfiguration.getBroadcastTables().add("t_bank");
// Default database strategy configuration
shardingRuleConfiguration.setDefaultDatabaseShardingStrategy(new StandardShardingStrategyConfiguration("user_id", "default_db_strategy_inline"));
Properties defaultDatabaseStrategyInlineProps = new Properties();
defaultDatabaseStrategyInlineProps.setProperty("algorithm-expression", "ds_${user_id % 2}");
shardingRuleConfiguration.getShardingAlgorithms().put("default_db_strategy_inline", new ShardingSphereAlgorithmConfiguration("INLINE", defaultDatabaseStrategyInlineProps));

// Key generate algorithm configuration
Properties snowflakeProperties = new Properties();
shardingRuleConfiguration.getKeyGenerators().put("snowflake", new ShardingSphereAlgorithmConfiguration("SNOWFLAKE", snowflakeProperties));

/* Data encrypt rule configuration */
Properties encryptProperties = new Properties();
encryptProperties.setProperty("aes-key-value", "123456");
EncryptColumnRuleConfiguration columnConfigAes = new EncryptColumnRuleConfiguration("username", "username", "", "username_plain", "name_encryptor");
EncryptColumnRuleConfiguration columnConfigTest = new EncryptColumnRuleConfiguration("pwd", "pwd", "assisted_query_pwd", "", "pwd_encryptor");
EncryptTableRuleConfiguration encryptTableRuleConfig = new EncryptTableRuleConfiguration("t_user", Arrays.asList(columnConfigAes, columnConfigTest));
// Data encrypt algorithm configuration
Map<String, ShardingSphereAlgorithmConfiguration> encryptAlgorithmConfigs = new LinkedHashMap<>(2, 1);
encryptAlgorithmConfigs.put("name_encryptor", new ShardingSphereAlgorithmConfiguration("AES", encryptProperties));
encryptAlgorithmConfigs.put("pwd_encryptor", new ShardingSphereAlgorithmConfiguration("assistedTest",
encryptProperties));
EncryptRuleConfiguration encryptRuleConfiguration = new EncryptRuleConfiguration(Collections.
singleton(encryptTableRuleConfig), encryptAlgorithmConfigs);

/* Readwrite-splitting rule configuration */
Properties readwriteProps1 = new Properties();
readwriteProps1.setProperty("write-data-source-name", "write_ds0");
readwriteProps1.setProperty("read-data-source-names", "write_ds0_read0, write_ds0_read1");
ReadwriteSplittingDataSourceRuleConfiguration dataSourceConfiguration1 = new
ReadwriteSplittingDataSourceRuleConfiguration("ds_0", "Static", readwriteProps1, "roundRobin");
Properties readwriteProps2 = new Properties();
readwriteProps2.setProperty("write-data-source-name", "write_ds0");
readwriteProps2.setProperty("read-data-source-names", "write_ds1_read0, write_ds1_read1");
ReadwriteSplittingDataSourceRuleConfiguration dataSourceConfiguration2 = new
ReadwriteSplittingDataSourceRuleConfiguration("ds_1", "Static", readwriteProps2, "roundRobin");

// Load balance algorithm configuration
Map<String, ShardingSphereAlgorithmConfiguration> loadBalanceMaps = new HashMap<>(1);
loadBalanceMaps.put("roundRobin", new ShardingSphereAlgorithmConfiguration("ROUND_ROBIN", new Properties()));

ReadwriteSplittingRuleConfiguration readWriteSplittingRuleConfiguration = new ReadwriteSplittingRuleConfiguration(Arrays.
asList(dataSourceConfiguration1, dataSourceConfiguration2), loadBalanceMaps);

/* Other Properties configuration */
Properties otherProperties = new Properties();
otherProperties.setProperty("sql-show", "true");

/* The variable `shardingDataSource` is the logic data source referenced by other frameworks(such as ORM, JPA, etc.) */
DataSource shardingDataSource = ShardingSphereDataSourceFactory.createDataSource(datasourceMaps, Arrays.
asList(shardingRuleConfiguration, readWriteSplittingRuleConfiguration, encryptRuleConfiguration), otherProperties);
```

## 7.1.3 YAML Configuration

### Overview

YAML configuration provides interaction with DBPlusEngine Driver through configuration files. When used together with the governance module, the configuration of persistence in the configuration center is YAML format.

Note: the YAML configuration file supports more than 3MB of configuration content.

YAML configuration is the most common configuration mode, which can omit the complexity of programming and simplify user configuration.

### Usage

### Import Maven Dependency

```
<dependency>
  <groupId>org.apache.shardingsphere</groupId>
  <artifactId>shardingsphere-jdbc-core</artifactId>
  <version>${shardingsphere.version}</version>
</dependency>
```

### YAML Format

The DBPlusEngine-Driver YAML file consists of schema name, mode configuration, data source map, rule configurations and properties.

Note: the example connection pool is HikariCP, which can be replaced with other connection pools according to business scenarios.

```
# Alias of the datasource in JDBC.
# Through this parameter to connect, ShardingSphere-JDBC and ShardingSphere-Proxy.
# Default value: logic_db
schemaName (?):

mode:

dataSources:

rules:
- !FOO_XXX
   ...
- !BAR_XXX
   ...

props:
 key_1: value_1
 key_2: value_2
```

Please refer to Mode Confiugration for more mode details.

Please refer to Data Source Confiugration for more data source details.

Please refer to Rules Confiugration for more rule details.

### Create Data Source

The ShardingSphereDataSource created by YamlShardingSphereDataSourceFactory implements the standard JDBC DataSource interface.

```
File yamlFile = // Indicate YAML file
DataSource dataSource = YamlShardingSphereDataSourceFactory.createDataSource(yamlFile);
```

### Use Data Source

Same with Java API.

### YAML Syntax Explanation

!! means instantiation of that class

! means self-defined alias

- means one or multiple can be included

[] means array, can substitutable with - each other

## Mode Configuration

### Configuration Item Explanation

```
mode (?): # Default value is Memory
  type: # Type of mode configuration. Values could be: Cluster
  repository (?): # Persist repository configuration. Memory type does not need persist
  overwrite: # Whether overwrite persistent configuration with local configuration
```

### Cluster Mode

```
mode:
  type: Cluster
  repository:
    type: # Type of persist repository
    props: # Properties of persist repository
      namespace: # Namespace of registry center
      server-lists: # Server lists of registry center
      foo_key: foo_value
      bar_key: bar_value
  overwrite: # Whether overwrite persistent configuration with local configuration
```

Please refer to Builtin Persist Repository List for more details about type of repository.

## Data Source

It is divided into single data source configuration and multi data source configuration. DBPlusEngine-Driver Supports all JDBC drivers and database connection pools.

In this example, the database driver is MySQL, and connection pool is HikariCP, which can be replaced with other database drivers and connection pools.

### Configuration Item Explanation

```
dataSources: # Data sources configuration, multiple <data-source-name> available
  <data-source-name>: # Data source name
    dataSourceClassName: # Data source class name
    driverClassName: # Class name of database driver, ref property of connection pool
    jdbcUrl: # Database URL, ref property of connection pool
    username: # Database username, ref property of connection pool
    password: # Database password, ref property of connection pool
    # ... Other properties for data source pool
```

### Example

```
dataSources:
  ds_1:
    dataSourceClassName: com.zaxxer.hikari.HikariDataSource
    driverClassName: com.mysql.jdbc.Driver
    jdbcUrl: jdbc:mysql://localhost:3306/ds_1
    username: root
    password:
  ds_2:
    dataSourceClassName: com.zaxxer.hikari.HikariDataSource
    driverClassName: com.mysql.jdbc.Driver
    jdbcUrl: jdbc:mysql://localhost:3306/ds_2
    username: root
    password:

  # Configure other data sources
```

### Rules

Rules are pluggable part of DBPlusEngine. This chapter is a YAML rule configuration manual for DBPlusEngine-Driver.

### Sharding

#### Configuration Item Explanation

```
rules:
- !SHARDING
  tables: # Sharding table configuration
    <logic-table-name> (+): # Logic table name
      actualDataNodes (?): # Describe data source names and actual tables (refer to Inline syntax rules)
      databaseStrategy (?): # Databases sharding strategy, use default databases sharding strategy if absent. sharding strategy below can choose only one.
        standard: # For single sharding column scenario
          shardingColumn: # Sharding column name
          shardingAlgorithmName: # Sharding algorithm name
        complex: # For multiple sharding columns scenario
          shardingColumns: # Sharding column names, multiple columns separated with comma
          shardingAlgorithmName: # Sharding algorithm name
        hint: # Sharding by hint
          shardingAlgorithmName: # Sharding algorithm name
        none: # Do not sharding
      tableStrategy: # Tables sharding strategy, same as database sharding strategy
      keyGenerateStrategy: # Key generator strategy
        column: # Column name of key generator
        keyGeneratorName: # Key generator name
  autoTables: # Auto Sharding table configuration
    t_order_auto: # Logic table name
      actualDataSources (?): # Data source names
      shardingStrategy: # Sharding strategy
        standard: # For single sharding column scenario
          shardingColumn: # Sharding column name
          shardingAlgorithmName: # Auto sharding algorithm name
  bindingTables (+): # Binding tables
    - <logic_table_name_1, logic_table_name_2, ...>
    - <logic_table_name_1, logic_table_name_2, ...>
  broadcastTables (+): # Broadcast tables
    - <table-name>
    - <table-name>
  defaultDatabaseStrategy: # Default strategy for database sharding
  defaultTableStrategy: # Default strategy for table sharding
  defaultKeyGenerateStrategy: # Default Key generator strategy
  defaultShardingColumn: # Default sharding column name

  # Sharding algorithm configuration
  shardingAlgorithms:
    <sharding-algorithm-name> (+): # Sharding algorithm name
      type: # Sharding algorithm type
      props: # Sharding algorithm properties
      # ...
```

```
 # Key generate algorithm configuration
 keyGenerators:
  <key-generate-algorithm-name> (+): # Key generate algorithm name
   type: # Key generate algorithm type
   props: # Key generate algorithm properties
   # ...
```

## Readwrite-splitting

### Configuration Item Explanation

### Static Readwrite-splitting

```
rules:
- !READWRITE_SPLITTING
 dataSources:
  <data-source-name> (+): # Logic data source name of readwrite-splitting
   static-strategy: # Readwrite-splitting type
    write-data-source-name: # Write data source name
    read-data-source-names: # Read data source names, multiple data source names separated with comma
   loadBalancerName: # Load balance algorithm name

 # Load balance algorithm configuration
 loadBalancers:
  <load-balancer-name> (+): # Load balance algorithm name
   type: # Load balance algorithm type
   props: # Load balance algorithm properties
    # ...
```

### Dynamic Readwrite-splitting

```
rules:
- !READWRITE_SPLITTING
 dataSources:
  <data-source-name> (+): # Logic data source name of readwrite-splitting
   dynamic-strategy: # Readwrite-splitting type
    auto-aware-data-source-name: # Database discovery logic data source name
    write-data-source-query-enabled: # All read data source are offline, write data source whether the data source is
responsible for read traffic
   loadBalancerName: # Load balance algorithm name

 # Load balance algorithm configuration
 loadBalancers:
  <load-balancer-name> (+): # Load balance algorithm name
   type: # Load balance algorithm type
   props: # Load balance algorithm properties
    # ...
```

Please refer to Built-in Load Balance Algorithm List for more details about type of algorithm. Please refer to Use Norms for more details about query consistent routing.

## HA

```
rules:
- !DB_DISCOVERY
 dataSources:
  <data-source-name> (+): # Logic data source name
   dataSourceNames: # Data source names
    - <data-source>
    - <data-source>
   discoveryHeartbeatName: # Detect heartbeat name
   discoveryTypeName: # Database discovery type name

 # Heartbeat Configuration
 discoveryHeartbeats:
  <discovery-heartbeat-name> (+): # heartbeat name
   props:
    keep-alive-cron: # This is cron expression, such as： '0/5 * * * ?'

 # Database Discovery Configuration
 discoveryTypes:
  <discovery-type-name> (+): # Database discovery type name
   type: # Database discovery type, such as: MySQL.NORMAL_REPLICATION, MySQL.MGR, openGauss.NORMAL_REPLICATION, SphereEx:GaussDB_for_MySQL.NORMAL_REPLICATION
   props (?):
    group-name: 92504d5b-6dec-11e8-91ea-246e9612aaf1 # Required parameters for database discovery types, such as MGR's group-name
```

## Encryption

### Configuration Item Explanation

```
rules:
- !ENCRYPT
 tables:
  <table-name> (+): # Encrypt table name
   columns:
    <column-name> (+): # Encrypted column name
     dataType: # Logical column type
     cipherColumn: # Ciphertext column name
     cipherDataType: # Encrypted column type
     assistedQueryColumn (?):  # Query auxiliary column name
     assistedQueryDataType: # Query auxiliary column type
     plainColumn (?): # Plaintext column name
     plainDataType: # Plaintext type
     encryptorName: # Encryption algorithm name
   queryWithCipherColumn(?): # Whether the table uses encrypted columns for query

 # Encryption algorithm configuration
 encryptors:
  <encrypt-algorithm-name> (+): # Encryption and decryption algorithm name
   type: # Encryption and decryption algorithm type
   props: # Encryption and decryption algorithm attribute configuration
    # ...

 queryWithCipherColumn: # Whether query with cipher column for data encrypt. You can use plaintext to query if have.

 # Key Storage Configuration
 keyManagers:
   <key-manager-name> (+): # Name of key storage manager
    type: # Key storage manager type. Local storage and AWS cloud storage are supported.
    props: # Property configuration of key storage manager
     # ...
```

### Shadow DB

### Configuration Item Explanation

```
rules:
 - !SHADOW
  dataSources:
   shadowDataSource:
    sourceDataSourceName: # Production data source name
    shadowDataSourceName: # Shadow data source name
  tables:
   <table-name>:
    dataSourceNames: # Shadow table location shadow data source names
     - <shadow-data-source>
    shadowAlgorithmNames: # Shadow table location shadow algorithm names
     - <shadow-algorithm-name>
  defaultShadowAlgorithmName: # Default shadow algorithm name
  shadowAlgorithms:
   <shadow-algorithm-name> (+): # Shadow algorithm name
    type: # Shadow algorithm type
    props: # Shadow algorithm property configuration
    # ...
```

### SQL-parser

### Configuration Item Explanation

```
rules:
- !SQL_PARSER
 sqlCommentParseEnabled: # Whether to parse SQL comments
 sqlStatementCache: # SQL statement local cache
  initialCapacity: # Initial capacity of local cache
  maximumSize: # Maximum capacity of local cache
  concurrencyLevel: # Local cache concurrency level, the maximum number of concurrent updates allowed by threads
 parseTreeCache: # Parse tree local cache
  initialCapacity: # Initial capacity of local cache
  maximumSize: # Maximum capacity of local cache
  concurrencyLevel: # Local cache concurrency level, the maximum number of concurrent updates allowed by threads
```

### Mixed Rules

The overlay between rule items in a mixed configuration is associated by the data source name and the table name.

If the previous rule is aggregation-oriented, the next rule needs to use the aggregated logical data source name configured by the previous rule when configuring the data source. Similarly, if the previous rule is table aggregation-oriented, the next rule needs to use the aggregated logical table name configured by the previous rule when configuring the table.

## Configuration Item Explanation

```
dataSources: # Configure the real data source name.
 write_ds:
  # ...Omit specific configuration.
 read_ds_0:
  # ...Omit specific configuration.
 read_ds_1:
  # ...Omit specific configuration.

rules:
 - !SHARDING # Configure data sharding rules.
  tables:
   t_user:
    actualDataNodes: ds.t_user_${0..1} # Data source name 'ds' uses the logical data source name of the readwrite-splitting configuration.
    tableStrategy:
     standard:
      shardingColumn: user_id
      shardingAlgorithmName: t_user_inline
  shardingAlgorithms:
   t_user_inline:
    type: INLINE
    props:
     algorithm-expression: t_user_${user_id % 2}

 - !ENCRYPT # Configure data encryption rules.
  tables:
   t_user: # Table `t_user` is the name of the logical table that uses the data sharding configuration.
    columns:
     pwd:
      plainColumn: plain_pwd
      cipherColumn: cipher_pwd
      encryptorName: encryptor_aes
  encryptors:
   encryptor_aes:
    type: aes
    props:
     aes-key-value: 123456abc

 - !READWRITE_SPLITTING # Configure readwrite-splitting rules.
  dataSources:
   ds: # The logical data source name 'ds' for readwrite-splitting is used in data sharding.
    type: Static
    props:
     write-data-source-name: write_ds # Use the real data source name 'write_ds'.
     read-data-source-names: read_ds_0, read_ds_1 # Use the real data source name 'read_ds_0', 'read_ds_1'.
    loadBalancerName: roundRobin
  loadBalancers:
   roundRobin:
    type: ROUND_ROBIN

props:
 sql-show: true
```

## 7.1.4 JDBC Driver

### Overview

DBPlusEngine-Driver provides JDBC driver, it permits user using ShardingSphere by configuration updating only, without any code changes.

### Usage

### Import Maven Dependency

```
<dependency>
  <groupId>org.apache.shardingsphere</groupId>
  <artifactId>shardingsphere-jdbc-core</artifactId>
  <version>${shardingsphere.version}</version>
</dependency>
```

### Driver Usage

### Native Driver Usage

```
Class.forName("org.apache.shardingsphere.driver.ShardingSphereDriver");
String jdbcUrl = "jdbc:shardingsphere:classpath:config.yaml";

String sql = "SELECT i.* FROM t_order o JOIN t_order_item i ON o.order_id=i.order_id WHERE o.user_id=? AND o.order_id=?";
try (
    Connection conn = DriverManager.getConnection(jdbcUrl);
    PreparedStatement ps = conn.prepareStatement(sql)) {
  ps.setInt(1, 10);
  ps.setInt(2, 1000);
  try (ResultSet rs = preparedStatement.executeQuery()) {
    while(rs.next()) {
      // ...
    }
  }
}
```

### Database Connection Pool Usage

```
String driverClassName = "org.apache.shardingsphere.driver.ShardingSphereDriver";
String jdbcUrl = "jdbc:shardingsphere:classpath:config.yaml";

// Use HikariCP as sample
HikariDataSource dataSource = new HikariDataSource();
dataSource.setDriverClassName(driverClassName);
dataSource.setJdbcUrl(jdbcUrl);

String sql = "SELECT i.* FROM t_order o JOIN t_order_item i ON o.order_id=i.order_id WHERE o.user_id=? AND o.order_id=?";
try (
    Connection conn = dataSource.getConnection();
    PreparedStatement ps = conn.prepareStatement(sql)) {
  ps.setInt(1, 10);
  ps.setInt(2, 1000);
  try (ResultSet rs = preparedStatement.executeQuery()) {
    while(rs.next()) {
            // ...
        }
    }
}
```

### Configuration Explanation

### Driver Class Name

org.apache.shardingsphere.driver.ShardingSphereDriver

### URL Configuration Explanation

- Use jdbc:shardingsphere: as prefix
- Configuration file: xxx.yaml, keep consist format with YAML Configuration
- Configuration file loading rule:
  - No prefix for loading from absolute path
  - Prefix with classpath: for loading from java class path

## 7.1.5 Spring Boot Starter

### Overview

DBPlusEngine-Driver provides the official Spring Boot Starter to make it convenient for developers to integrate DBPlusEngine-Driver and Spring Boot.

The list of compatible SpringBoot versions is as follows:

1. SpringBoot 1.x
2. SpringBoot 2.x
3. SpringBoot 3.x (Experimental)

### Usage

### Import Maven Dependency

```
<dependency>
    <groupId>org.apache.shardingsphere</groupId>
    <artifactId>shardingsphere-jdbc-core-spring-boot-starter</artifactId>
    <version>${shardingsphere.version}</version>
</dependency>
```

### Use DBPlusEngine Data Source in Spring

Developers can use DataSource to inject to use native JDBC or ORM frameworks such as JPA, Hibernate or MyBatis. Take native JDBC usage as an example:

```
@Resource
private DataSource dataSource;
```

### Mode Configuration

Default is Memory mode.

### Configuration Item Explanation

```
spring.shardingsphere.mode.type= # Type of mode configuration. Values could be: Cluster
spring.shardingsphere.mode.repository= # Persist repository configuration. Memory type does not need persist
spring.shardingsphere.mode.overwrite= # Whether overwrite persistent configuration with local configuration
```

### Cluster Mode

```
spring.shardingsphere.mode.type=Cluster
spring.shardingsphere.mode.repository.type= # Type of persist repository
spring.shardingsphere.mode.repository.props.namespace= # Namespace of registry center
spring.shardingsphere.mode.repository.props.server-lists= # Server lists of registry center
spring.shardingsphere.mode.repository.props.<key>= # Properties of persist repository
spring.shardingsphere.mode.overwrite= # Whether overwrite persistent configuration with local configuration
```

Please refer to Builtin Persist Repository List for more details about type of repository.

### Data Source

### Use Native Data Source

### Configuration Item Explanation

```
spring.shardingsphere.datasource.names= # Actual data source name, multiple split by `,`

# <actual-data-source-name> indicate name of data source name
spring.shardingsphere.datasource.<actual-data-source-name>.type= # Full class name of database connection pool
spring.shardingsphere.datasource.<actual-data-source-name>.driver-class-name= # Class name of database driver, ref property
of connection pool
spring.shardingsphere.datasource.<actual-data-source-name>.jdbc-url= # Database URL, ref property of connection pool
spring.shardingsphere.datasource.<actual-data-source-name>.username= # Database username, ref property of connection pool
spring.shardingsphere.datasource.<actual-data-source-name>.password= # Database password, ref property of connection pool
spring.shardingsphere.datasource.<actual-data-source-name>.<xxx>= # ... Other properties for data source pool
```

### Example

In this example, the database driver is MySQL, and connection pool is HikariCP, which can be replaced with other database drivers and connection pools.

```
# Configure actual data sources
spring.shardingsphere.datasource.names=ds1,ds2

# Configure the 1st data source
spring.shardingsphere.datasource.ds1.type=com.zaxxer.hikari.HikariDataSource
spring.shardingsphere.datasource.ds1.driver-class-name=com.mysql.jdbc.Driver
spring.shardingsphere.datasource.ds1.jdbc-url=jdbc:mysql://localhost:3306/ds1
spring.shardingsphere.datasource.ds1.username=root
spring.shardingsphere.datasource.ds1.password=

# Configure the 2nd data source
spring.shardingsphere.datasource.ds2.type=com.zaxxer.hikari.HikariDataSource
spring.shardingsphere.datasource.ds2.driver-class-name=com.mysql.jdbc.Driver
spring.shardingsphere.datasource.ds2.jdbc-url=jdbc:mysql://localhost:3306/ds2
spring.shardingsphere.datasource.ds2.username=root
spring.shardingsphere.datasource.ds2.password=
```

### Use JNDI Data Source

If developer plan to use ShardingSphere-JDBC in Web Server (such as Tomcat) with JNDI data source, spring.shardingsphere.datasource.${datasourceName}.jndiName can be used as an alternative to series of configuration of data source.

### Configuration Item Explanation

```
spring.shardingsphere.datasource.names= # Actual data source name, multiple split by `,`

# <actual-data-source-name> indicate name of data source name
spring.shardingsphere.datasource.<actual-data-source-name>.jndi-name= # JNDI of data source
```

### Example

```
# Configure actual data sources
spring.shardingsphere.datasource.names=ds1,ds2

# Configure the 1st data source
spring.shardingsphere.datasource.ds1.jndi-name=java:comp/env/jdbc/ds1
# Configure the 2nd data source
spring.shardingsphere.datasource.ds2.jndi-name=java:comp/env/jdbc/ds2
```

### Rules

Rules are pluggable part of DBPlusEngine. This chapter is a Spring Boot Starter rule configuration manual for DBPlusEngine-Driver.

### Sharding

### Configuration Item Explanation

```
spring.shardingsphere.datasource.names= # Omit the data source configuration, please refer to the usage

# Standard sharding table configuration
spring.shardingsphere.rules.sharding.tables.<table-name>.actual-data-nodes= # Describe data source names and actual tables, delimiter as point, multiple data nodes separated with comma, support inline expression. Absent means sharding databases only.

# Databases sharding strategy, use default databases sharding strategy if absent. sharding strategy below can choose only one.

# For single sharding column scenario
spring.shardingsphere.rules.sharding.tables.<table-name>.database-strategy.standard.sharding-column= # Sharding column name
spring.shardingsphere.rules.sharding.tables.<table-name>.database-strategy.standard.sharding-algorithm-name= # Sharding algorithm name

# For multiple sharding columns scenario
spring.shardingsphere.rules.sharding.tables.<table-name>.database-strategy.complex.sharding-columns= # Sharding column names, multiple columns separated with comma
spring.shardingsphere.rules.sharding.tables.<table-name>.database-strategy.complex.sharding-algorithm-name= # Sharding algorithm name

# Sharding by hint
spring.shardingsphere.rules.sharding.tables.<table-name>.database-strategy.hint.sharding-algorithm-name= # Sharding algorithm name

# Tables sharding strategy, same as database sharding strategy
spring.shardingsphere.rules.sharding.tables.<table-name>.table-strategy.xxx= # Omitted

# Auto sharding table configuraiton
spring.shardingsphere.rules.sharding.auto-tables.<auto-table-name>.actual-data-sources= # data source names

spring.shardingsphere.rules.sharding.auto-tables.<auto-table-name>.sharding-strategy.standard.sharding-column= # Sharding column name
spring.shardingsphere.rules.sharding.auto-tables.<auto-table-name>.sharding-strategy.standard.sharding-algorithm-name= # Auto sharding algorithm name

# Key generator strategy configuration
spring.shardingsphere.rules.sharding.tables.<table-name>.key-generate-strategy.column= # Column name of key generator
spring.shardingsphere.rules.sharding.tables.<table-name>.key-generate-strategy.key-generator-name= # Key generator name

spring.shardingsphere.rules.sharding.binding-tables[0]= # Binding table name
spring.shardingsphere.rules.sharding.binding-tables[1]= # Binding table name
spring.shardingsphere.rules.sharding.binding-tables[x]= # Binding table name

spring.shardingsphere.rules.sharding.broadcast-tables[0]= # Broadcast tables
spring.shardingsphere.rules.sharding.broadcast-tables[1]= # Broadcast tables
spring.shardingsphere.rules.sharding.broadcast-tables[x]= # Broadcast tables

spring.shardingsphere.sharding.default-database-strategy.xxx= # Default strategy for database sharding
spring.shardingsphere.sharding.default-table-strategy.xxx= # Default strategy for table sharding
spring.shardingsphere.sharding.default-key-generate-strategy.xxx= # Default Key generator strategy
spring.shardingsphere.sharding.default-sharding-column= # Default sharding column name

# Sharding algorithm configuration
spring.shardingsphere.rules.sharding.sharding-algorithms.<sharding-algorithm-name>.type= # Sharding algorithm type
spring.shardingsphere.rules.sharding.sharding-algorithms.<sharding-algorithm-name>.props.xxx=# Sharding algorithm properties

# Key generate algorithm configuration
spring.shardingsphere.rules.sharding.key-generators.<key-generate-algorithm-name>.type= # Key generate algorithm type
spring.shardingsphere.rules.sharding.key-generators.<key-generate-algorithm-name>.props.xxx= # Key generate algorithm properties
```

Please refer to Built-in Sharding Algorithm List and Built-in Key Generate Algorithm List for more details about type of algorithm.

#### Attention

Inline expression identifier can use ${...} or $->{...}, but ${...} is conflict with spring placeholder of properties, so use $->{...} on spring environment is better.

### Readwrite splitting

#### Configuration Item Explanation

#### Static Readwrite-splitting

```
spring.shardingsphere.datasource.names= # Omit the data source configuration, please refer to the usage

spring.shardingsphere.rules.readwrite-splitting.data-sources.<readwrite-splitting-data-source-name>.static-strategy.write-data-source-name= # Write data source name
spring.shardingsphere.rules.readwrite-splitting.data-sources.<readwrite-splitting-data-source-name>.static-strategy.read-data-source-names= # Read data source names, multiple data source names separated with comma
spring.shardingsphere.rules.readwrite-splitting.data-sources.<readwrite-splitting-data-source-name>.load-balancer-name= # Load balance algorithm name

# Load balance algorithm configuration
spring.shardingsphere.rules.readwrite-splitting.load-balancers.<load-balance-algorithm-name>.type= # Load balance algorithm type
spring.shardingsphere.rules.readwrite-splitting.load-balancers.<load-balance-algorithm-name>.props.xxx= # Load balance algorithm properties
```

### Dynamic Readwrite-splitting

```
spring.shardingsphere.datasource.names= # Omit the data source configuration, please refer to the usage

spring.shardingsphere.rules.readwrite-splitting.data-sources.<readwrite-splitting-data-source-name>.dynamic-strategy.auto-aware-data-source-name= # Database discovery logic data source name
spring.shardingsphere.rules.readwrite-splitting.data-sources.<readwrite-splitting-data-source-name>.dynamic-strategy.write-data-source-query-enabled= # All read data source are offline, write data source whether the data source is responsible for read traffic
spring.shardingsphere.rules.readwrite-splitting.data-sources.<readwrite-splitting-data-source-name>.load-balancer-name= # Load balance algorithm name

# Load balance algorithm configuration
spring.shardingsphere.rules.readwrite-splitting.load-balancers.<load-balance-algorithm-name>.type= # Load balance algorithm type
spring.shardingsphere.rules.readwrite-splitting.load-balancers.<load-balance-algorithm-name>.props.xxx= # Load balance algorithm properties
```

Please refer to Built-in Load Balance Algorithm List for more details about type of algorithm. Please refer to Use Norms for more details about query consistent routing.

### HA

### Configuration Item Explanation

```
spring.shardingsphere.datasource.names= # Omit the data source configuration, please refer to the usage

spring.shardingsphere.rules.database-discovery.data-sources.<database-discovery-data-source-name>.data-source-names= # Data source names, multiple data source names separated with comma. Such as: ds_0, ds_1
spring.shardingsphere.rules.database-discovery.data-sources.<database-discovery-data-source-name>.discovery-heartbeat-name= # Detect heartbeat name
spring.shardingsphere.rules.database-discovery.data-sources.<database-discovery-data-source-name>.discovery-type-name= # Database discovery type name

spring.shardingsphere.rules.database-discovery.discovery-heartbeats.<discovery-heartbeat-name>.props.keep-alive-cron= # This is cron expression, such as：'0/5 * * * * ?'

spring.shardingsphere.rules.database-discovery.discovery-types.<discovery-type-name>.type= # Database discovery type, such as: MGR、openGauss
spring.shardingsphere.rules.database-discovery.discovery-types.<discovery-type-name>.props.group-name= # Required parameters for database discovery types, such as MGR's group-name
```

### Encryption

### Configuration Item Explanation

```
spring.shardingsphere.datasource.names= # Omit the data source configuration, please refer to the user manual

spring.shardingsphere.rules.encrypt.tables.<table-name>.query-with-cipher-column= # Whether the table uses cipher columns for query
spring.shardingsphere.rules.encrypt.tables.<table-name>.columns.<column-name>.data-type= # Logical column type
spring.shardingsphere.rules.encrypt.tables.<table-name>.columns.<column-name>.cipher-column= # Encrypted column name
spring.shardingsphere.rules.encrypt.tables.<table-name>.columns.<column-name>.cipher-data-type= # Encrypted column type
spring.shardingsphere.rules.encrypt.tables.<table-name>.columns.<column-name>.assisted-query-column= # Query auxiliary column name
spring.shardingsphere.rules.encrypt.tables.<table-name>.columns.<column-name>.assisted-query-data-type= # Query auxiliary column type
spring.shardingsphere.rules.encrypt.tables.<table-name>.columns.<column-name>.plain-column= # Plaintext column name
spring.shardingsphere.rules.encrypt.tables.<table-name>.columns.<column-name>.plain-data-type= # Plaintext column type
spring.shardingsphere.rules.encrypt.tables.<table-name>.columns.<column-name>.encryptor-name= # Encryption algorithm name

# Encryption algorithm configuration
spring.shardingsphere.rules.encrypt.encryptors.<encrypt-algorithm-name>.type= # Encryption algorithm type
spring.shardingsphere.rules.encrypt.encryptors.<encrypt-algorithm-name>.props.xxx= # Encryption algorithm attribute configuration

spring.shardingsphere.rules.encrypt.queryWithCipherColumn= # Whether query with cipher column for data encrypt. User you can use plaintext to query if have.
```

### Example

```
spring.shardingsphere.rules.encrypt.encryptors.pwd-encryptor.type=AES
spring.shardingsphere.rules.encrypt.encryptors.pwd-encryptor.props.aes-key-value=123456ab

spring.shardingsphere.rules.encrypt.tables.t_user.columns.pwd.cipher-column=pwd_encrypt
spring.shardingsphere.rules.encrypt.tables.t_user.columns.pwd.data-type=INT NOT NULL
spring.shardingsphere.rules.encrypt.tables.t_user.columns.pwd.cipher-data-type=VARCHAR(200) NOT NULL
spring.shardingsphere.rules.encrypt.tables.t_user.columns.pwd.plain-column=pwd_plain
spring.shardingsphere.rules.encrypt.tables.t_user.columns.pwd.plain-data-type=INT NOT NULL
spring.shardingsphere.rules.encrypt.tables.t_user.columns.pwd.assisted-query-column=pwd_assisted
spring.shardingsphere.rules.encrypt.tables.t_user.columns.pwd.assisted-query-data-type= VARCHAR(200) NOT NULL
spring.shardingsphere.rules.encrypt.tables.t_user.columns.pwd.encryptor-name=pwd-encryptor
```

Please refer to Built-in Encrypt Algorithm List for more details about type of algorithm.

### Shadow DB

### Configuration Item Explanation

```
spring.shardingsphere.datasource.names= # Omit the data source configuration, please refer to the usage

spring.shardingsphere.rules.shadow.data-sources.shadow-data-source.source-data-source-name= # Production data source name
spring.shardingsphere.rules.shadow.data-sources.shadow-data-source.shadow-data-source-name= # Shadow data source name

spring.shardingsphere.rules.shadow.tables.<table-name>.data-source-names= # Shadow table location shadow data source names (multiple values are separated by ",")
spring.shardingsphere.rules.shadow.tables.<table-name>.shadow-algorithm-names= # Shadow table location shadow algorithm names (multiple values are separated by ",")

spring.shardingsphere.rules.shadow.defaultShadowAlgorithmName= # Default shadow algorithm name， optional item.

spring.shardingsphere.rules.shadow.shadow-algorithms.<shadow-algorithm-name>.type= # Shadow algorithm type
spring.shardingsphere.rules.shadow.shadow-algorithms.<shadow-algorithm-name>.props.xxx= # Shadow algorithm property configuration
```

### SQL Parser

### Configuration Item Explanation

```
spring.shardingsphere.rules.sql-parser.sql-comment-parse-enabled= # Whether to parse SQL comments

spring.shardingsphere.rules.sql-parser.sql-statement-cache.initial-capacity= # Initial capacity of SQL statement local cache
spring.shardingsphere.rules.sql-parser.sql-statement-cache.maximum-size= # Maximum capacity of SQL statement local cache
spring.shardingsphere.rules.sql-parser.sql-statement-cache.concurrency-level= # SQL statement local cache concurrency level, the maximum number of concurrent updates allowed by threads

spring.shardingsphere.rules.sql-parser.parse-tree-cache.initial-capacity= # Initial capacity of parse tree local cache
spring.shardingsphere.rules.sql-parser.parse-tree-cache.maximum-size= # Maximum local cache capacity of parse tree
spring.shardingsphere.rules.sql-parser.parse-tree-cache.concurrency-level= # The local cache concurrency level of the parse tree. The maximum number of concurrent updates allowed by threads
```

### Mixed Rules

### Configuration Item Explanation

```
# data source configuration
spring.shardingsphere.datasource.names= write-ds0,write-ds1,write-ds0-read0,write-ds1-read0

spring.shardingsphere.datasource.write-ds0.url= # Database URL connection
spring.shardingsphere.datasource.write-ds0.type=  # Database connection pool type name
spring.shardingsphere.datasource.write-ds0.driver-class-name= # Database driver class name
spring.shardingsphere.datasource.write-ds0.username= # Database username
spring.shardingsphere.datasource.write-ds0.password= # Database password
spring.shardingsphere.datasource.write-ds0.xxx=  # Other properties of database connection pool

spring.shardingsphere.datasource.write-ds1.url= # Database URL connection
# ...Omit specific configuration.

spring.shardingsphere.datasource.write-ds0-read0.url= # Database URL connection
# ...Omit specific configuration.

spring.shardingsphere.datasource.write-ds1-read0.url= # Database URL connection
# ...Omit specific configuration.

# Sharding rules configuration
# Databases sharding strategy
spring.shardingsphere.rules.sharding.default-database-strategy.standard.sharding-column=user_id
spring.shardingsphere.rules.sharding.default-database-strategy.standard.sharding-algorithm-name=default-database-strategy-inline
# Binding table rules configuration ,and multiple groups of binding-tables configured with arrays
spring.shardingsphere.rules.sharding.binding-tables[0]=t_user,t_user_detail
spring.shardingsphere.rules.sharding.binding-tables[1]= # Binding table names,multiple table name are separated by commas
spring.shardingsphere.rules.sharding.binding-tables[x]= # Binding table names,multiple table name are separated by commas
# Broadcast table rules configuration
spring.shardingsphere.rules.sharding.broadcast-tables= # Broadcast table names,multiple table name are separated by commas

# Table sharding strategy
# The enumeration value of `ds_$->{0..1}` is the name of the logical data source configured with readwrite-splitting
spring.shardingsphere.rules.sharding.tables.t_user.actual-data-nodes=ds_$->{0..1}.t_user_$->{0..1}
spring.shardingsphere.rules.sharding.tables.t_user.table-strategy.standard.sharding-column=user_id
spring.shardingsphere.rules.sharding.tables.t_user.table-strategy.standard.sharding-algorithm-name=user-table-strategy-inline

# Data encrypt configuration
# Table `t_user` is the name of the logical table that uses for data sharding configuration.
spring.shardingsphere.rules.encrypt.tables.t_user.columns.username.cipher-column=username
spring.shardingsphere.rules.encrypt.tables.t_user.columns.username.encryptor-name=name-encryptor
spring.shardingsphere.rules.encrypt.tables.t_user.columns.pwd.cipher-column=pwd
spring.shardingsphere.rules.encrypt.tables.t_user.columns.pwd.encryptor-name=pwd-encryptor

# Data encrypt algorithm configuration
spring.shardingsphere.rules.encrypt.encryptors.name-encryptor.type=AES
spring.shardingsphere.rules.encrypt.encryptors.name-encryptor.props.aes-key-value=123456abc
spring.shardingsphere.rules.encrypt.encryptors.pwd-encryptor.type=AES
spring.shardingsphere.rules.encrypt.encryptors.pwd-encryptor.props.aes-key-value=123456abc

# Key generate strategy configuration
spring.shardingsphere.rules.sharding.tables.t_user.key-generate-strategy.column=user_id
spring.shardingsphere.rules.sharding.tables.t_user.key-generate-strategy.key-generator-name=snowflake

# Sharding algorithm configuration
spring.shardingsphere.rules.sharding.sharding-algorithms.default-database-strategy-inline.type=INLINE
# The enumeration value of `ds_$->{user_id % 2}` is the name of the logical data source configured with readwrite-splitting
spring.shardingsphere.rules.sharding.sharding-algorithms.default-database-strategy-inline.algorithm-expression=ds$->{user_id
% 2}
spring.shardingsphere.rules.sharding.sharding-algorithms.user-table-strategy-inline.type=INLINE
spring.shardingsphere.rules.sharding.sharding-algorithms.user-table-strategy-inline.algorithm-expression=t_user_$->{user_id %
2}

# Key generate algorithm configuration
spring.shardingsphere.rules.sharding.key-generators.snowflake.type=SNOWFLAKE

# read query configuration
# ds_0,ds_1 is the logical data source name of the readwrite-splitting
spring.shardingsphere.rules.readwrite-splitting.data-sources.ds_0.type=Static
spring.shardingsphere.rules.readwrite-splitting.data-sources.ds_0.props.write-data-source-name=write-ds0
spring.shardingsphere.rules.readwrite-splitting.data-sources.ds_0.props.read-data-source-names=write-ds0-read0
spring.shardingsphere.rules.readwrite-splitting.data-sources.ds_0.load-balancer-name=read-random
spring.shardingsphere.rules.readwrite-splitting.data-sources.ds_1.type=Static
spring.shardingsphere.rules.readwrite-splitting.data-sources.ds_1.props.write-data-source-name=write-ds1
spring.shardingsphere.rules.readwrite-splitting.data-sources.ds_1.props.read-data-source-names=write-ds1-read0
spring.shardingsphere.rules.readwrite-splitting.data-sources.ds_1.load-balancer-name=read-random

# Load balance algorithm configuration
spring.shardingsphere.rules.readwrite-splitting.load-balancers.read-random.type=RANDOM
```

### 7.1.6 Spring Namespace

#### Overview

DBPlusEngine-Driver provides official Spring Namespace to make convenient for developers to integrate DBPlusEngine-Driver and Spring.

#### Usage

#### Import Maven Dependency

```
<dependency>
  <groupId>org.apache.shardingsphere</groupId>
  <artifactId>shardingsphere-jdbc-core-spring-namespace</artifactId>
  <version>${shardingsphere.version}</version>
</dependency>
```

### Configure Spring Bean

### Configuration Item Explanation

Namespace: http://shardingsphere.apache.org/schema/shardingsphere/datasource/datasource-5.0.0.xsd

<shardingsphere:data-source />

| Name | Type | Description |
|---|---|---|
| id | Attribute | Spring Bean Id |
| schema-name (?) | Attribute | JDBC data source alias |
| data-source-names | Attribute | Data source name, multiple data source names are separated by commas |
| rule-refs | Attribute | Rule name, multiple rule names are separated by commas |
| mode (?) | Tag | Mode configuration |
| props (?) | Tag | Properties configuration, Please refer to Properties Configuration for more details |

### Example

```xml
<beans xmlns="http://www.springframework.org/schema/beans"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xmlns:shardingsphere="http://shardingsphere.apache.org/schema/shardingsphere/datasource"
     xsi:schemaLocation="http://www.springframework.org/schema/beans
                  http://www.springframework.org/schema/beans/spring-beans.xsd
                  http://shardingsphere.apache.org/schema/shardingsphere/datasource
                  http://shardingsphere.apache.org/schema/shardingsphere/datasource/datasource.xsd
                  ">
  <shardingsphere:data-source id="ds" schema-name="foo_schema" data-source-names="..." rule-refs="...">
    <shardingsphere:mode type="..." />
    <props>
      <prop key="xxx.xxx">${xxx.xxx}</prop>
    </props>
  </shardingsphere:data-source>
</beans>
```

### Use DBPlusEngine Data Source in Spring

Same with Spring Boot Starter.

### Mode Configuration

### Configuration Item Explanation

Namespace: http://shardingsphere.apache.org/schema/shardingsphere/datasource/datasource-5.1.0.xsd

<shardingsphere:mode />

| Name | Type | Description | Default Value |
|---|---|---|---|
| type | Attribute | Type of mode configuration. Values could be: Cluster | |
| repository-ref (?) | Attribute | Persist repository configuration. Memory type does not need persist | |
| overwrite (?) | Attribute | Whether overwrite persistent configuration with local configuration | false |

### Memory Mode

It is the default value.

### Example

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xmlns:shardingsphere="http://shardingsphere.apache.org/schema/shardingsphere/datasource"
     xsi:schemaLocation="http://www.springframework.org/schema/beans
                http://www.springframework.org/schema/beans/spring-beans.xsd
                http://shardingsphere.apache.org/schema/shardingsphere/datasource
                http://shardingsphere.apache.org/schema/shardingsphere/datasource/datasource.xsd">

  <shardingsphere:data-source id="ds" schema-name="foo_schema" data-source-names="..." rule-refs="..." />
</beans>
```

### Cluster Mode

### Configuration Item Explanation

Namespace: http://shardingsphere.apache.org/schema/shardingsphere/mode-repository/cluster/repository-5.0.0.xsd

| Name | Type | Description |
|---|---|---|
| id | Attribute | Name of persist repository bean |
| type | Attribute | Type of persist repository |
| namespace | Attribute | Namespace of registry center |
| server-lists | Attribute | Server lists of registry center |
| props (?) | Tag | Properties of persist repository |

### Example

```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:shardingsphere="http://shardingsphere.apache.org/schema/shardingsphere/datasource"
    xmlns:cluster="http://shardingsphere.apache.org/schema/shardingsphere/mode-repository/cluster"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
               http://www.springframework.org/schema/beans/spring-beans.xsd
               http://shardingsphere.apache.org/schema/shardingsphere/datasource
               http://shardingsphere.apache.org/schema/shardingsphere/datasource/datasource.xsd
               http://shardingsphere.apache.org/schema/shardingsphere/mode-repository/cluster
               http://shardingsphere.apache.org/schema/shardingsphere/mode-repository/cluster/repository.xsd">
  <cluster:repository id="clusterRepository" type="Zookeeper" namespace="regCenter" server-lists="localhost:3182">
    <props>
      <prop key="max-retries">3</prop>
      <prop key="operation-timeout-milliseconds">1000</prop>
    </props>
  </cluster:repository>

  <shardingsphere:data-source id="ds" schema-name="foo_schema" data-source-names="..." rule-refs="...">
    <shardingsphere:mode type="Cluster" repository-ref="clusterRepository" overwrite="true" />
  </shardingsphere:data-source>
</beans>
```

Please refer to Builtin Persist Repository List for more details about type of repository.

### Data Source

Any data source configured as spring bean can be cooperated with spring namespace.

### Example

In this example, the database driver is MySQL, and connection pool is HikariCP, which can be replaced with other database drivers and connection pools.

```xml
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:shardingsphere="http://shardingsphere.apache.org/schema/shardingsphere/datasource"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
                http://www.springframework.org/schema/beans/spring-beans.xsd
                http://shardingsphere.apache.org/schema/shardingsphere/datasource
                http://shardingsphere.apache.org/schema/shardingsphere/datasource/datasource.xsd
                ">
  <bean id="ds1" class="com.zaxxer.hikari.HikariDataSource" destroy-method="close">
    <property name="driverClassName" value="com.mysql.jdbc.Driver" />
    <property name="jdbcUrl" value="jdbc:mysql://localhost:3306/ds1" />
    <property name="username" value="root" />
    <property name="password" value="" />
  </bean>

  <bean id="ds2" class="com.zaxxer.hikari.HikariDataSource" destroy-method="close">
    <property name="driverClassName" value="com.mysql.jdbc.Driver" />
    <property name="jdbcUrl" value="jdbc:mysql://localhost:3306/ds2" />
    <property name="username" value="root" />
    <property name="password" value="" />
  </bean>

  <shardingsphere:data-source id="ds" schema-name="foo_schema" data-source-names="ds1,ds2" rule-refs="..." />
</beans>
```

### Rules

Rules are pluggable part of DBPlusEngine. This chapter is a Spring namespace rule configuration manual for DBPlusEngine-Driver.

### Sharding

### Configuration Item Explanation

Namespace: http://shardingsphere.apache.org/schema/shardingsphere/sharding/sharding-5.1.0.xsd

<sharding:rule />

| Name | T ype | Description |
|---|---|---|
| id | A ttri bute | Spring Bean Id |
| table-rules (?) | Tag | Sharding table rule configuration |
| auto-table-rules (?) | Tag | Automatic sharding table rule configuration |
| binding-table-rules (?) | Tag | Binding table rule configuration |
| broadcast-table-rules (?) | Tag | Broadcast table rule configuration |
| def ault-database-strategy-ref (?) | A ttri bute | Default database strategy name |
| default-table-strategy-ref (?) | A ttri bute | Default table strategy name |
| default -key-generate-strategy-ref (?) | A ttri bute | Default key generate strategy name |
| default-sharding-column (?) | A ttri bute | Default sharding column name |

<sharding:table-rule />

| Name | Type | Description |
|---|---|---|
| logic-table | Attribute | Logic table name |
| actual-data-nodes | Attribute | Describe data source names and actual tables, delimiter as point, multiple data nodes separated with comma, support inline expression. Absent means sharding databases only. |
| actual-data-sources | Attribute | Data source names for auto sharding table |
| database-strategy-ref | Attribute | Database strategy name for standard sharding table |
| table-strategy-ref | Attribute | Table strategy name for standard sharding table |
| sharding-strategy-ref | Attribute | sharding strategy name for auto sharding table |
| key-generate-strategy-ref | Attribute | Key generate strategy name |

<sharding:binding-table-rules />

| Name | Type | Description |
|---|---|---|
| binding-table-rule (+) | Tag | Binding table rule configuration |

<sharding:binding-table-rule />

| Name | Type | Description |
|---|---|---|
| logi c-tables | Attr ibute | Binding table name, multiple tables separated with comma |

<sharding:broadcast-table-rules />

| Name | Type | Description |
|---|---|---|
| broadcast-table-rule (+) | Tag | Broadcast table rule configuration |

<sharding:broadcast-table-rule />

| Name | Type | Description |
|---|---|---|
| table | Attribute | Broadcast table name |

<sharding:standard-strategy />

| Name | Type | Description |
|---|---|---|
| id | Attribute | Standard sharding strategy name |
| sharding-column | Attribute | Sharding column name |
| algorithm-ref | Attribute | Sharding algorithm name |

<sharding:complex-strategy />

| Name | T ype | Description |
|---|---|---|
| id | A ttri bute | Complex sharding strategy name |
| shardi ng-columns | A ttri bute | Sharding column names, multiple columns separated with comma |
| alg orithm-ref | A ttri bute | Sharding algorithm name |

<sharding:hint-strategy />

| Name | Type | Description |
|---|---|---|
| id | Attribute | Hint sharding strategy name |
| algorithm-ref | Attribute | Sharding algorithm name |

<sharding:none-strategy />

| Name | Type | Description |
|---|---|---|
| id | Attribute | Sharding strategy name |

<sharding:key-generate-strategy />

| Name | Type | Description |
|---|---|---|
| id | Attribute | Key generate strategy name |
| column | Attribute | Key generate column name |
| algorithm-ref | Attribute | Key generate algorithm name |

<sharding:sharding-algorithm />

| Name | Type | Description |
|---|---|---|
| id | Attribute | Sharding algorithm name |
| type | Attribute | Sharding algorithm type |
| props (?) | Tag | Sharding algorithm properties |

<sharding:key-generate-algorithm />

| Name | Type | Description |
|---|---|---|
| id | Attribute | Key generate algorithm name |
| type | Attribute | Key generate algorithm type |
| props (?) | Tag | Key generate algorithm properties |

Please refer to Built-in Sharding Algorithm List and Built-in Key Generate Algorithm List for more details about type of algorithm.

### Attention

Inline expression identifier can use ${...} or $->{...}, but ${...} is conflict with spring placeholder of properties, so use $->{...} on spring environment is better.

### Readwrite-splitting

#### Configuration Item Explanation

Namespace: http://shardingsphere.apache.org/schema/shardingsphere/readwrite-splitting/readwrite-splitting-5.1.2.xsd

<readwrite-splitting:rule />

| Name | ■<br>Type* | Description |
|---|---|---|
| id | Attr ibute | Spring Bean Id |
| data-source-rule (+) | Tag | Readwrite-splitting data source rule configuration |

<readwrite-splitting:data-source-rule />

| Name | Type | Description |
|---|---|---|
| id | Attribute | Readwrite-splitting data source rule name |
| static-strategy | Tag | Static Readwrite-splitting type |
| dynamic-strategy | Tag | Dynamic Readwrite-splitting type |
| l oad-balance-algorithm-ref | Attribute | Load balance algorithm name |

<readwrite-splitting:static-strategy />

| Name | Type | Description |
|---|---|---|
| id | Attribute | Static readwrite-splitting name |
| write-d ata-source-name | Attribute | Write data source name |
| read-da ta-source-names | Attribute | Read data source names, multiple data source names separated with comma |
| load-balanc e-algorithm-ref | Attribute | Load balance algorithm name |

<readwrite-splitting:dynamic-strategy />

| Name | Type | Description |
|---|---|---|
| id | Attribute | Dynamic readwrite-splitting name |
| aut o-aware-data -source-name | Attribute | Database discovery logic data source name |
| write-d ata-source-q uery-enabled | Attribute | All read data source are offline, write data source whether the data source is responsible for read traffic |
| lo ad-balance-a lgorithm-ref | Attribute | Load balance algorithm name |

<readwrite-splitting:load-balance-algorithm />

| Name | Type | Description |
|---|---|---|
| id | Attribute | Load balance algorithm name |
| type | Attribute | Load balance algorithm type |
| props (?) | Tag | Load balance algorithm properties |

Please refer to Built-in Load Balance Algorithm List for more details about type of algorithm. Please refer to Use Norms for more details about query consistent routing.

### HA

#### Configuration Item Explanation

Namespace：http://shardingsphere.apache.org/schema/shardingsphere/database-discovery/database-discovery-5.1.0.xsd

<database-discovery:rule />

| Name | Type | Description |
|---|---|---|
| id | Attribute | Spring Bean Id |
| data-source-rule (+) | tag | Data source rule configuration |
| discovery-heartbeat (+) | tag | Detect heartbeat rule configuration |

<database-discovery:data-source-rule />

| Name | Type | Description |
|---|---|---|
| id | Attribute | Data source rule Id |
| data- source-names | Attribute | Data source names, multiple data source names separated with comma. Such as: ds_0, ds_1 |
| discovery-heartbeat-name | Attribute | Detect heartbeat name |
| discovery-type-name | Attribute | Database discovery type name |

<database-discovery:discovery-heartbeat />

| Name | Type | Description |
|---|---|---|
| id | Attribute | Detect heartbeat Id |
| props | tag | Detect heartbeat attribute configuration, keep-alive-cron configuration, cron expression. Such as: ‘0/5 * * * * ?’ |

<database-discovery:discovery-type />

| Name | Type | Description |
|---|---|---|
| id | Attribute | Database discovery type Id |
| type | Attribute | Database discovery type, such as: MGR、openGauss |
| props (?) | tag | Required parameters for database discovery types, such as MGR’s group-name |

### Encryption

#### Configuration Item Explanation

Namespace: http://shardingsphere.apache.org/schema/shardingsphere/encrypt/encrypt-5.1.0.xsd

<encrypt:rule />

| Name | Type | Description | Default Value |
|---|---|---|---|
| id | Attribute | Spring Bean Id | |
| queryWithCipher-Column (?) | Attribute | Whether query with cipher column for data encrypt. User you can use plaintext to query if have | true |
| table (+) | Tag | Encrypt table configuration | |

<encrypt:table />

| Name | Type | Description |
|---|---|---|
| name | Attribute | Encrypt table name |
| column (+) | Tag | Encrypt column configuration |
| query-with-cipher-column(?) | Attribute | Whether query with cipher column for data encrypt. User you can use plain-text to query if have |

<encrypt:column />

| Name | Type | Description |
|---|---|---|
| logic-column | Attribute | Encrypted column logical name |
| data-type(?) | Attribute | Logical column type |
| cipher-column | Attribute | Encrypted column name |
| cipher-data-type(?) | Attribute | Encrypted column type |
| assisted-query-column (?) | Attribute | Query auxiliary column name |
| assisted-query-data-type(?) | Attribute | Query auxiliary column type |
| plain-column (?) | Attribute | Plaintext column name |
| plain-data-type(?) | Attribute | Plaintext column type |
| encrypt-algorithm-ref | Attribute | Encryption algorithm name |

<encrypt:encrypt-algorithm />

| Name | Type | Description |
|---|---|---|
| id | Attribute | Encrypt algorithm name |
| type | Attribute | Encrypt algorithm type |
| props (?) | Tag | Encrypt algorithm properties |

## Example

```
<encrypt:encrypt-algorithm id="name_encryptor" type="AES">
  <props>
    <prop key="aes-key-value">123456</prop>
  </props>
</encrypt:encrypt-algorithm>

<encrypt:rule id="encryptRule">
  <encrypt:table name="t_user">
    <encrypt:column logic-column="pwd" cipher-column="pwd_encrypt" data-type="VARCHAR(20) NOT NULL" cipher-data-type="VARCHAR(200) NOT NULL" plain-column="pwd_plain" plain-data-type="VARCHAR(20) NOT NULL" assisted-query-column="pwd_assisted" assisted-query-data-type="VARCHAR(20) NOT NULL" encrypt-algorithm-ref="name_encryptor" />
  </encrypt:table>
</encrypt:rule>
```

Please refer to Built-in Encrypt Algorithm List for more details about type of algorithm.

### Shadow DB

#### Configuration Item Explanation

Namespace: http://shardingsphere.apache.org/schema/shardingsphere/shadow/shadow-5.1.0.xsd

<shadow:rule />

| Name | Type | Description |
|---|---|---|
| id | Attribute | Spring Bean Id |
| d ata-source(?) | Tag | Shadow data source configuration |
| defaul t-shadow-algo rithm-name(?) | Tag | Default shadow algorithm configuration |
| sh adow-table(?) | Tag | Shadow table configuration |

<shadow:data-source />

| Name | Type | Description |
|---|---|---|
| id | Attribute | Spring Bean Id |
| source-data-source-name | Attribute | Production data source name |
| shadow-data-source-name | Attribute | Shadow data source name |

<shadow:default-shadow-algorithm-name />

| Name | Type | Description |
|---|---|---|
| name | Attribute | Default shadow algorithm name |

<shadow:shadow-table />

| Name | Type | Description |
|---|---|---|
| name | Attribute | Shadow table name |
| data-sources | Attribute | Shadow table location shadow data source names (multiple values are separated by “,” ) |
| algorithm (?) | Tag | Shadow table location shadow algorithm configuration |

<shadow:algorithm />

| Name | Type | Description |
|---|---|---|
| shadow-algorithm-ref | Attribute | Shadow table location shadow algorithm name |

<shadow:shadow-algorithm />

| Name | Type | Description |
|---|---|---|
| id | Attribute | Shadow algorithm name |
| type | Attribute | Shadow algorithm type |
| props (?) | Attribute | Shadow algorithm property configuration |

## SQL Parser

### Configuration Item Explanation

Namespace: http://shardingsphere.apache.org/schema/shardingsphere/sql-parser/sql-parser-5.1.0.xsd

<sql-parser:rule />

| Name | Type | Description |
|---|---|---|
| id | Attribute | Spring Bean Id |
| sql-comment-parse-enable | Attribute | Whether to parse SQL comments |
| parse-tree-cache-ref | Attribute | Parse tree local cache name |
| sql-statement-cache-ref | Attribute | SQL statement local cache name |

<sql-parser:cache-option />

| Name | T ype | Description |
|---|---|---|
| id | Attribute | Local cache configuration item name |
| initial-capacity | Attribute | Initial capacity of local cache |
| maximum-size | Attribute | Maximum capacity of local cache |
| concurrency-level | Attribute | Local cache concurrency level, the maximum number of concurrent updates allowed by threads |

## Mixed Rules

### Configuration Item Explanation

```
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:shardingsphere="http://shardingsphere.apache.org/schema/shardingsphere/datasource"
    xmlns:readwrite-splitting="http://shardingsphere.apache.org/schema/shardingsphere/readwrite-splitting"
    xmlns:encrypt="http://shardingsphere.apache.org/schema/shardingsphere/encrypt"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
                http://www.springframework.org/schema/beans/spring-beans.xsd
                http://shardingsphere.apache.org/schema/shardingsphere/datasource
                http://shardingsphere.apache.org/schema/shardingsphere/datasource/datasource.xsd
                http://shardingsphere.apache.org/schema/shardingsphere/readwrite-splitting
                http://shardingsphere.apache.org/schema/shardingsphere/readwrite-splitting/readwrite-splitting.xsd
                http://shardingsphere.apache.org/schema/shardingsphere/encrypt
                http://shardingsphere.apache.org/schema/shardingsphere/encrypt/encrypt.xsd
                ">
  <bean id="write_ds0" class="  com.zaxxer.hikari.HikariDataSource" init-method="init" destroy-method="close">
    <property name="driverClassName" value="com.mysql.jdbc.Driver" />
    <property name="jdbcUrl" value="jdbc:mysql://localhost:3306/write_ds?useSSL=false&amp;useUnicode=true&amp;
characterEncoding=UTF-8" />
    <property name="username" value="root" />
    <property name="password" value="" />
  </bean>

  <bean id="read_ds0_0" class="  com.zaxxer.hikari.HikariDataSource" init-method="init" destroy-method="close">
    <!-- ...Omit specific configuration. -->
  </bean>

  <bean id="read_ds0_1" class="  com.zaxxer.hikari.HikariDataSource" init-method="init" destroy-method="close">
    <!-- ...Omit specific configuration. -->
  </bean>

  <bean id="write_ds1" class="  com.zaxxer.hikari.HikariDataSource" init-method="init" destroy-method="close">
    <!-- ...Omit specific configuration. -->
  </bean>

  <bean id="read_ds1_0" class="  com.zaxxer.hikari.HikariDataSource" init-method="init" destroy-method="close">
    <!-- ...Omit specific configuration. -->
  </bean>

  <bean id="read_ds1_1" class="  com.zaxxer.hikari.HikariDataSource" init-method="init" destroy-method="close">
    <!-- ...Omit specific configuration. -->
  </bean>

  <!-- load balance algorithm configuration for readwrite-splitting -->
  <readwrite-splitting:load-balance-algorithm id="randomStrategy" type="RANDOM" />

  <!-- readwrite-splitting rule configuration -->
  <readwrite-splitting:rule id="readWriteSplittingRule">
    <readwrite-splitting:data-source-rule id="ds_0" type="Static" load-balance-algorithm-ref="randomStrategy">
      <props>
        <prop key="write-data-source-name">write_ds0</prop>
        <prop key="read-data-source-names">read_ds0_0, read_ds0_1</prop>
      </props>
    </readwrite-splitting:data-source-rule>
    <readwrite-splitting:data-source-rule id="ds_1" type="Static" load-balance-algorithm-ref="randomStrategy">
      <props>
        <prop key="write-data-source-name">write_ds1</prop>
        <prop key="read-data-source-names">read_ds1_0, read_ds1_1</prop>
      </props>
    </readwrite-splitting:data-source-rule>
  </readwrite-splitting:rule>

  <!-- sharding strategy configuration -->
  <sharding:standard-strategy id="databaseStrategy" sharding-column="user_id" algorithm-ref=
"inlineDatabaseStrategyAlgorithm" />
  <sharding:standard-strategy id="orderTableStrategy" sharding-column="order_id" algorithm-ref=
"inlineOrderTableStrategyAlgorithm" />
  <sharding:standard-strategy id="orderItemTableStrategy" sharding-column="order_item_id" algorithm-ref=
"inlineOrderItemTableStrategyAlgorithm" />

  <sharding:sharding-algorithm id="inlineDatabaseStrategyAlgorithm" type="INLINE">
    <props>
      <!-- the expression enumeration is the logical data source name of the readwrite-splitting configuration -->
      <prop key="algorithm-expression">ds_${user_id % 2}</prop>
    </props>
  </sharding:sharding-algorithm>
  <sharding:sharding-algorithm id="inlineOrderTableStrategyAlgorithm" type="INLINE">
    <props>
      <prop key="algorithm-expression">t_order_${order_id % 2}</prop>
    </props>
  </sharding:sharding-algorithm>
  <sharding:sharding-algorithm id="inlineOrderItemTableStrategyAlgorithm" type="INLINE">
    <props>
      <prop key="algorithm-expression">t_order_item_${order_item_id % 2}</prop>
    </props>
  </sharding:sharding-algorithm>

  <!-- sharding rule configuration -->
  <sharding:rule id="shardingRule">
    <sharding:table-rules>
      <!-- the expression 'ds_${0..1}' enumeration is the logical data source name of the readwrite-splitting configuration  -->
      <sharding:table-rule logic-table="t_order" actual-data-nodes="ds_${0..1}.t_order_${0..1}" database-strategy-ref=
"databaseStrategy" table-strategy-ref="orderTableStrategy" key-generate-strategy-ref="orderKeyGenerator"/>
```

```
      <sharding:table-rule logic-table="t_order_item" actual-data-nodes="ds_${0..1}.t_order_item_${0..1}" database-strategy-ref="databaseStrategy" table-strategy-ref="orderItemTableStrategy" key-generate-strategy-ref="itemKeyGenerator"/>
    </sharding:table-rules>
    <sharding:binding-table-rules>
      <sharding:binding-table-rule logic-tables="t_order, t_order_item"/>
    </sharding:binding-table-rules>
    <sharding:broadcast-table-rules>
      <sharding:broadcast-table-rule table="t_address"/>
    </sharding:broadcast-table-rules>
  </sharding:rule>

  <!-- data encrypt configuration -->
  <encrypt:encrypt-algorithm id="name_encryptor" type="AES">
    <props>
      <prop key="aes-key-value">123456</prop>
    </props>
  </encrypt:encrypt-algorithm>
  <encrypt:encrypt-algorithm id="pwd_encryptor" type="assistedTest" />

  <encrypt:rule id="encryptRule">
    <encrypt:table name="t_user">
      <encrypt:column logic-column="username" cipher-column="username" plain-column="username_plain" encrypt-algorithm-ref="name_encryptor" />
      <encrypt:column logic-column="pwd" cipher-column="pwd" assisted-query-column="assisted_query_pwd" encrypt-algorithm-ref="pwd_encryptor" />
    </encrypt:table>
  </encrypt:rule>

  <!-- datasource configuration -->
  <!-- the element data-source-names's value is all of the datasource name -->
  <shardingsphere:data-source id="readQueryDataSource" data-source-names="write_ds0, read_ds0_0, read_ds0_1, write_ds1, read_ds1_0, read_ds1_1"
    rule-refs="readWriteSplittingRule, shardingRule, encryptRule" >
    <props>
      <prop key="sql-show">true</prop>
    </props>
  </shardingsphere:data-source>
</beans>
```

### 7.1.7 Properties Configuration

DBPlusEngine provides the way of property configuration to configure system level configuration.

### Configuration Item Explanation

| Name | Data Type | Description | Default Value |
|---|---|---|---|
| sql-show (?) | boolean | Whether show SQL or not in log. Print SQL details can help developers debug easier. The log details include: logic SQL, actual SQL and SQL parse result. Enable this property will log into log topic ShardingSphere-SQL, log level is INFO | false |
| sql-simple (?) | boolean | Whether show SQL details in simple style | false |
| kernel-executor-size (?) | int | The max thread size of worker group to execute SQL. One ShardingSphereDataSource will use a independent thread pool, it does not share thread pool even different data source in same JVM | infinite |
| max-connections-size-per-query (?) | int | Max opened connection size for each query | 1 |
| check-table-metadata-enabled (?) | boolean | Whether validate table meta data consistency when application startup or updated | false |
| check-duplicate-table-enabled (?) | boolean | Whether validate duplicate table when application startup or updated | false |
| sql-federation-enabled (?) | boolean | Whether enable SQL federation | false |

## 7.1.8 Builtin Algorithm

### Introduction

DBPlusEngine allows developers to implement algorithms via SPI; At the same time, DBPlusEngine also provides a couple of builtin algorithms for simplify developers.

## Usage

The builtin algorithms are configured by type and props. Type is defined by the algorithm in SPI, and props is used to deliver the customized parameters of the algorithm.

No matter which configuration type is used, the configured algorithm is named and passed to the corresponding rule configuration. This chapter distinguishes and lists all the builtin algorithms of DBPlusEngine according to its functions for developers' reference.

## Metadata Repository

### Background

DBPlusEngine provides different metadata persistence methods for different running modes. Users can choose an appropriate way to store metadata while configuring the running mode.

### Parameters

#### ZooKeeper Repository

Type: ZooKeeper

Mode: Cluster

Attributes:

| Name | Type | Description | Default Value |
|---|---|---|---|
| retryIn tervalMilliseconds | int | Milliseconds of retry interval | 500 |
| maxRetries | int | Max retries of client connection | 3 |
| timeToLiveSeconds | int | Seconds of ephemeral data live | 60 |
| operationT imeoutMilliseconds | int | Milliseconds of operation timeout | 500 |
| digest | String | Login password | |

#### Etcd Repository

Type: Etcd

Mode: Cluster

Attributes:

| Name | Type | Description | Default Value |
|---|---|---|---|
| timeToLiveSeconds | long | Seconds of ephemeral data live | 30 |
| connectionTimeout | long | Seconds of connection timeout | 30 |

### SSD Repository

Type: SphereEx:MATE

Mode: Cluster

Attributes:

none

### Procedure

1. Configure running mode in server.yaml.
2. Configure metadata persistence warehouse type.

### Sample

- Cluster mode.

```
mode:
 type: Cluster
 repository:
  type: SphereEx:MATE
  props:
   namespace: governance
   server-lists: localhost:21506
```

## Sharding Algorithm

### Auto Sharding Algorithm

### Modulo Sharding Algorithm

Type: MOD

Attributes:

| Name | DataType | Description |
|---|---|---|
| sharding-count | int | Sharding count |

### Hash Modulo Sharding Algorithm

Type: HASH_MOD

Attributes:

| Name | DataType | Description |
|---|---|---|
| sharding-count | int | Sharding count |

### Volume Based Range Sharding Algorithm

Type: VOLUME_RANGE

Attributes:

| Name | DataType | Description |
|---|---|---|
| range-lower | long | Range lower bound, throw exception if lower than bound |
| range-upper | long | Range upper bound, throw exception if upper than bound |
| sharding-volume | long | Sharding volume |

### Boundary Based Range Sharding Algorithm

Type: BOUNDARY_RANGE

Attributes:

| Name | DataType | Description |
|---|---|---|
| sharding-ranges | String | Range of sharding border, multiple boundaries separated by commas |

### Auto Interval Sharding Algorithm

Type: AUTO_INTERVAL

Attributes:

| Name | DataType | Description |
|---|---|---|
| datetime-lower | String | Shard datetime begin boundary, pattern: yyyy-MM-dd HH:mm:ss |
| datetime-upper | String | Shard datetime end boundary, pattern: yyyy-MM-dd HH:mm:ss |
| sharding-seconds | long | Max seconds for the data in one shard, allows sharding key timestamp format seconds with time precision, but time precision after seconds is automatically erased |

### Consistent Hash Sharding Algorithm

Type: SphereEx:CONSISTENT_HASH

Attributes:

| Name | DataType | Description |
|---|---|---|
| sharding-count | int | Required item, used to specify the number of sharding. |
| sharding-node-weight | String | It is not a required item. The weight of sharding data nodes is separated by commas, for example: "2,2,1,1". The larger the weight, the more virtual nodes used for consistency hash calculation, and the default weight is 1. |
| virtual-node-count | int | Required item. The number of virtual nodes corresponding to the sharding data node. The more virtual nodes, the more balanced the sharding. But too many virtual nodes will also affect the performance. |
| consistent-hash-seed | int | It is not required. The seed parameter of consistency hash is calculated. The default value is 0. |

### Standard Sharding Algorithm

Apache ShardingSphere built-in standard sharding algorithm are:

### Inline Sharding Algorithm

With Groovy expressions, InlineShardingStrategy provides single-key support for the sharding operation of = and IN in SQL. Simple sharding algorithms can be used through a simple configuration to avoid laborious Java code developments. For example, t_user_$->{u_id % 8} means table t_user is divided into 8 tables according to u_id, with table names from t_user_0 to t_user_7. Please refer to Inline Expression for more details.

Type: INLINE

Attributes:

| Name | DataType | Description | Default Value |
|---|---|---|---|
| algorithm-expression | String | Inline expression sharding algorithm | - |
| allow-range-query-with-inline-sharding (?) | boolean | Whether range query is allowed. Note: range query will ignore sharding strategy and conduct full routing | false |

### Interval Sharding Algorithm

Type: INTERVAL

Attributes:

| Name | DataType | Description | DefaultValue |
|---|---|---|---|
| datetime-pattern | String | Timestamp pattern of sharding value, must can be transformed to Java LocalDateTime. For example: yyyy-MM-dd HH:mm:ss | - |
| datetime-lower | String | Datetime sharding lower boundary, pattern is defined datetime-pattern | - |
| datetime-upper (?) | String | Datetime sharding upper boundary, pattern is defined datetime-pattern | Now |
| sharding-suffix-pattern | String | Suffix pattern of sharding data sources or tables, must can be transformed to Java LocalDateTime, must be consistent with datetime-interval-unit. For example: yyyyMM | - |
| datetime-interval-amount (?) | int | Interval of sharding value | 1 |
| datetime-interval-unit (?) | String | Unit of sharding value interval, must can be transformed to Java ChronoUnit’s Enum value. For example: MONTHS | DAYS |

### Complex Sharding Algorithm

### Complex Inline Sharding Algorithm

Please refer to Inline Expression for more details.

Type: COMPLEX_INLINE

| Name | DataType | Description | Default Value |
|---|---|---|---|
| sharding-columns (?) | String | sharing column names | - |
| algorithm-expression | String | Inline expression sharding algorithm | - |
| allow-range-query-with-inline-sharding (?) | boolean | Whether range query is allowed. Note: range query will ignore sharding strategy and conduct full routing | false |

### Hint Sharding Algorithm

#### Hint Inline Sharding Algorithm

Please refer to Inline Expression for more details.

Type: COMPLEX_INLINE

| Name | DataType | Description | Default Value |
|---|---|---|---|
| alg orithm-expression | String | Inline expression sharding algorithm | ${value} |

### Class Based Sharding Algorithm

Realize custom extension by configuring the sharding strategy type and algorithm class name.

Type： CLASS_BASED

Attributes：

| Name | DataType | Description |
|---|---|---|
| strategy | String | Sharding strategy type, support STANDARD, COMPLEX or HINT (case insensitive) |
| algorithmClassName | String | Fully qualified name of sharding algorithm |

### Key Generate Algorithm

#### Snowflake

Type: SNOWFLAKE

Attributes:

| Name | DataType | Description | Default Value |
|---|---|---|---|
| max-tolerate-time-difference-milliseconds (?) | long | The max tolerate time for different server’ s time difference in milliseconds | 10 milliseconds |
| max-vibration-offset (?) | int | The max upper limit value of vibrate number, range [0, 4096). Notice: To use the generated value of this algorithm as sharding value, it is recommended to configure this property. The algorithm generates key mod 2^n (2^n is usually the sharding amount of tables or databases) in different milliseconds and the result is always 0 or 1. To prevent the above sharding problem, it is recommended to configure this property, its value is (2^n)-1 | 1 |

### UUID

Type: UUID

Attributes: None

## Load Balance Algorithm

### Round Robin Algorithm

Type: ROUND_ROBIN

Attributes: None

### Random Algorithm

Type: RANDOM

Attributes: None

### Delay Algorithm

Type: SPHERE_EX_DELAY_REPLICA

Attributes: None

### Weight Algorithm

Type: WEIGHT

Attributes:

All read data in use must be configured with weights

| Name | Dat aType | Description |
|---|---|---|
| - <read-data_source-name> (+ ) | d ouble | The attribute name uses the read database name, and the parameter fills in the weight:Double.MAX_VALUE. |

## Encryption Algorithm

### MD5 Encrypt Algorithm

Type: MD5

Attributes: None

### AES Encrypt Algorithm

Type: AES

Attributes:

| Name | DataType | Description |
|---|---|---|
| aes-key-value | String | AES KEY |

### RC4 Encrypt Algorithm

Type: RC4

Attributes:

| Name | DataType | Description |
|---|---|---|
| rc4-key-value | String | RC4 KEY |

### SM3 Encrypt Algorithm

Type: SM3

Attributes:

| Name | DataType | Description |
|---|---|---|
| sm3-salt | String | SM3 SALT (should be blank or 8 bytes long) |

### SM4 Encrypt Algorithm

Type: SM4

Attributes:

| Name | DataType | Description |
|---|---|---|
| sm4-key | String | SM4 KEY (should be 16 bytes) |
| sm4-mode | String | SM4 MODE (should be CBC or ECB) |
| sm4-iv | String | SM4 IV (should be specified on CBC, 16 bytes long) |
| sm4-padding | String | SM4 PADDING (should be PKCS5Padding or PKCS7Padding, NoPadding excepted) |

### Shadow Algorithm

### Column Shadow Algorithm

### Column Value Match Shadow Algorithm

Type：VALUE_MATCH

Attributes:

| Name | DataType | Description |
|---|---|---|
| column | String | Shadow column |
| operation | String | SQL operation type（INSERT, UPDATE, DELETE, SELECT） |
| value | String | Shadow column matching value |

### Column Regex Match Shadow Algorithm

Type: REGEX_MATCH

Attributes:

| Name | DataType | Description |
|---|---|---|
| column | String | Shadow column |
| operation | String | SQL operation type (insert, update, delete, select) |
| regex | String | Shadow column matching regular expression |

### Hint Shadow Algorithm

### Simple Hint Shadow Algorithm

Type: SIMPLE_HINT

Attributes:

Configure at least a set of arbitrary key-value pairs. For example: foo:bar

| Name | DataType | Description |
|---|---|---|
| foo | String | bar |

### Sharding Audit Algorithm

### Background

The sharding audit is to audit the SQL statements in the sharding database. Sharding audit not only intercept illegal SQL statements, but also gather the SQL statistics.

### Parameters

### DML_SHARDING_CONDITIONS algorithm

Type: DML_SHARDING_CONDITIONS

### Procedure

1. when configuring data sharding rules, create sharding audit configurations.

### Sample

- DML_SHARDING_CONDITIONS

```
auditors:
 sharding_key_required_auditor:
  type: DML_SHARDING_CONDITIONS
```

## 7.1.9 Special API

This chapter will introduce the special API of DB Plus Engine-Driver.

### Sharding

This chapter will introduce the Sharding API of DBPlusEngine-Driver.

#### Mandatory Routing

#### Introduction

DBPlusEngine uses ThreadLocal to manage sharding values for mandatory routing. A sharding value can be added to HintManager through programming, and the value is only valid in the current thread. Additionally, DBPlusEngine can also perform mandatory routing by adding annotations to SQL.

Hint can be used in the following scenarios:

- Fields used for sharding exist in external business logic rather than in SQL, database, and table structures.
- Some data operations are forcibly carried out in the primary database.

#### Procedure

1. Call HintManager.getInstance() to obtain HintManager instance.
2. Call HintManager.addDatabaseShardingValue, HintManager.addTableShardingValue method to set the sharding key value.
3. Execute SQL statements to complete routing and execution.
4. Call HintManager.close to clear the content of ThreadLocal.

#### Sample

#### Sharding with Hint

#### Rules Configuration

Hint sharding algorithm requires users to implement org.apache.shardingsphere.sharding.api.sharding.hint.HintShardingAlgorithm interface. DBPlusEngine will obtain sharding values from HintManager for routing.

Configuration sample:

```yaml
rules:
- !SHARDING
 tables:
  t_order:
   actualDataNodes: demo_ds_${0..1}.t_order_${0..1}
   databaseStrategy:
    hint:
     algorithmClassName: xxx.xxx.xxx.HintXXXAlgorithm
   tableStrategy:
    hint:
     algorithmClassName: xxx.xxx.xxx.HintXXXAlgorithm
 defaultTableStrategy:
  none:
 defaultKeyGenerateStrategy:
    type: SNOWFLAKE
    column: order_id

props:
  sql-show: true
```

### Obtain HintManager

```
HintManager hintManager = HintManager.getInstance();
```

### Add Sharding Key Value

- Use hintManager.addDatabaseShardingValue to add data source sharding key value.
- Use hintManager.addTableShardingValue to add table sharding key value.

  When database sharding is required and table sharding is not, while mandatory routing is carried out to a sub-database, you can use hintManager.setDatabaseShardingValue to set sharding values.

### Clear Sharding Key Value

The sharding key value is stored in ThreadLocal, so you need to call hintManager.close() at the end of the operation to clear the content of ThreadLocal.

__hintManager has implemented the AutoCloseable interface, and you are advised to use try with resource to close it automatically.

### Complete Code Example

```
// Sharding database and table using HintManager
String sql = "SELECT * FROM t_order";
try (HintManager hintManager = HintManager.getInstance();
     Connection conn = dataSource.getConnection();
     PreparedStatement preparedStatement = conn.prepareStatement(sql)) {
    hintManager.addDatabaseShardingValue("t_order", 1);
    hintManager.addTableShardingValue("t_order", 2);
    try (ResultSet rs = preparedStatement.executeQuery()) {
        while (rs.next()) {
            // ...
        }
    }
}

// Sharding database without sharding table and routing to only one database using HintManager
String sql = "SELECT * FROM t_order";
try (HintManager hintManager = HintManager.getInstance();
     Connection conn = dataSource.getConnection();
     PreparedStatement preparedStatement = conn.prepareStatement(sql)) {
    hintManager.setDatabaseShardingValue(3);
    try (ResultSet rs = preparedStatement.executeQuery()) {
        while (rs.next()) {
            // ...
        }
    }
}
```

### Use SQL Annotations

#### Terms of Use

Before using the SQL Hint, users should enable configurations of parsing annotations in advance and set sqlCommentParseEnabled to true.

The annotation format only supports /* */ and content has to start with ShardingSphere hint:. Optional properties include:

- {table}.SHARDING_DATABASE_VALUE: used to add data source sharding key value corresponding to {table}. Multiple properties are separated by a comma.
- {table}.SHARDING_TABLE_VALUE: used to add table sharding key value corresponding to {table}. Multiple properties are separated by a comma.

  When database sharding is required and table sharding is not, while mandatory routing is carried out to a sub-database, you can use SHARDING_DATABASE_VALUE to add shards without specifying {table}.

#### Complete Sample

```
/* SHARDINGSPHERE_HINT: t_order.SHARDING_DATABASE_VALUE=1, t_order.SHARDING_TABLE_VALUE=1 */
SELECT * FROM t_order;
```

### Transaction

Using distributed transaction through DBPlusEngine is no different from local transaction. In addition to transparent use of distributed transaction, DBPlusEngine can switch distributed transaction types every time the database accesses.

Supported transaction types include local, XA and BASE. It can be set before creating a database connection, and default value can be set when DBPlusEngine startup.

#### Use Java API

#### Import Maven Dependency

```
<dependency>
  <groupId>org.apache.shardingsphere</groupId>
  <artifactId>shardingsphere-jdbc-core</artifactId>
  <version>${shardingsphere.version}</version>
</dependency>

<!-- import if using XA transaction -->
<dependency>
  <groupId>org.apache.shardingsphere</groupId>
  <artifactId>shardingsphere-transaction-xa-core</artifactId>
  <version>${shardingsphere.version}</version>
</dependency>

<!-- import if using BASE transaction -->
<dependency>
  <groupId>org.apache.shardingsphere</groupId>
  <artifactId>shardingsphere-transaction-base-seata-at</artifactId>
  <version>${shardingsphere.version}</version>
</dependency>
```

### Use Distributed Transaction

```
TransactionTypeHolder.set(TransactionType.XA); // Support TransactionType.LOCAL, TransactionType.XA, TransactionType.BASE
try (Connection conn = dataSource.getConnection()) { // Use ShardingSphereDataSource
    conn.setAutoCommit(false);
    PreparedStatement ps = conn.prepareStatement("INSERT INTO t_order (user_id, status) VALUES (?, ?)");
    ps.setObject(1, 1000);
    ps.setObject(2, "init");
    ps.executeUpdate();
    conn.commit();
}
```

### Use Spring Boot Starter

### Import Maven Dependency

```
<dependency>
    <groupId>org.apache.shardingsphere</groupId>
    <artifactId>shardingsphere-jdbc-core-spring-boot-starter</artifactId>
    <version>${shardingsphere.version}</version>
</dependency>

<!-- import if using XA transaction -->
<dependency>
    <groupId>org.apache.shardingsphere</groupId>
    <artifactId>shardingsphere-transaction-xa-core</artifactId>
    <version>${shardingsphere.version}</version>
</dependency>

<!-- import if using BASE transaction -->
<dependency>
    <groupId>org.apache.shardingsphere</groupId>
    <artifactId>shardingsphere-transaction-base-seata-at</artifactId>
    <version>${shardingsphere.version}</version>
</dependency>
```

### Configure Transaction Manager

```
@Configuration
@EnableTransactionManagement
public class TransactionConfiguration {

    @Bean
    public PlatformTransactionManager txManager(final DataSource dataSource) {
        return new DataSourceTransactionManager(dataSource);
    }

    @Bean
    public JdbcTemplate jdbcTemplate(final DataSource dataSource) {
        return new JdbcTemplate(dataSource);
    }
}
```

### Use Distributed Transaction

```
@Transactional
@ShardingSphereTransactionType(TransactionType.XA)  // Support TransactionType.LOCAL, TransactionType.XA, TransactionType.BASE
public void insert() {
   jdbcTemplate.execute("INSERT INTO t_order (user_id, status) VALUES (?, ?)", (PreparedStatementCallback<Object>) ps -> {
      ps.setObject(1, i);
      ps.setObject(2, "init");
      ps.executeUpdate();
   });
}
```

## Use Spring Namespace

### Import Maven Dependency

```
<dependency>
   <groupId>org.apache.shardingsphere</groupId>
   <artifactId>shardingsphere-jdbc-core-spring-namespace</artifactId>
   <version>${shardingsphere.version}</version>
</dependency>

<!-- import if using XA transaction -->
<dependency>
   <groupId>org.apache.shardingsphere</groupId>
   <artifactId>shardingsphere-transaction-xa-core</artifactId>
   <version>${shardingsphere.version}</version>
</dependency>

<!-- import if using BASE transaction -->
<dependency>
   <groupId>org.apache.shardingsphere</groupId>
   <artifactId>shardingsphere-transaction-base-seata-at</artifactId>
   <version>${shardingsphere.version}</version>
</dependency>
```

### Configure Transaction Manager

```
<!-- ShardingDataSource configuration -->
<!-- ...  -->

<bean id="transactionManager" class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
   <property name="dataSource" ref="shardingDataSource" />
</bean>
<bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">
   <property name="dataSource" ref="shardingDataSource" />
</bean>
<tx:annotation-driven />

<!-- Enable auto scan @ShardingSphereTransactionType annotation to inject the transaction type before connection created -->
<sharding:tx-type-annotation-driven />
```

### Use Distributed Transaction

```
@Transactional
@ShardingSphereTransactionType(TransactionType.XA)  // Support TransactionType.LOCAL, TransactionType.XA,
TransactionType.BASE
public void insert() {
   jdbcTemplate.execute("INSERT INTO t_order (user_id, status) VALUES (?, ?)", (PreparedStatementCallback<Object>) ps -> {
      ps.setObject(1, i);
      ps.setObject(2, "init");
      ps.executeUpdate();
   });
}
```

### Atomikos Transaction

The default XA transaction manager of DBPlusEngine is Atomikos.

### Data Recovery

xa_tx.log generated in the project logs folder is necessary for the recovery when XA crashes. Please keep it.

### Update Configuration

Developer can add jta.properties in classpath of the application to customize Atomikos configuration. For detailed configuration rules.

Please refer to Atomikos official documentation for more details.

### Narayana Transaction

### Import Maven Dependency

```
<properties>
   <narayana.version>5.9.1.Final</narayana.version>
   <jboss-transaction-spi.version>7.6.0.Final</jboss-transaction-spi.version>
   <jboss-logging.version>3.2.1.Final</jboss-logging.version>
</properties>

<dependency>
   <groupId>org.apache.shardingsphere</groupId>
   <artifactId>shardingsphere-jdbc-core</artifactId>
   <version>${shardingsphere.version}</version>
</dependency>

<!-- Import if using XA transaction -->
<dependency>
   <groupId>org.apache.shardingsphere</groupId>
   <artifactId>shardingsphere-transaction-xa-core</artifactId>
   <version>${shardingsphere.version}</version>
</dependency>

<dependency>
   <groupId>org.apache.shardingsphere</groupId>
   <artifactId>shardingsphere-transaction-xa-narayana</artifactId>
   <version>${shardingsphere.version}</version>
```

```xml
</dependency>
<dependency>
  <groupId>org.jboss.narayana.jta</groupId>
  <artifactId>jta</artifactId>
  <version>${narayana.version}</version>
</dependency>
<dependency>
  <groupId>org.jboss.narayana.jts</groupId>
  <artifactId>narayana-jts-integration</artifactId>
  <version>${narayana.version}</version>
</dependency>
<dependency>
  <groupId>org.jboss</groupId>
  <artifactId>jboss-transaction-spi</artifactId>
  <version>${jboss-transaction-spi.version}</version>
</dependency>
<dependency>
  <groupId>org.jboss.logging</groupId>
  <artifactId>jboss-logging</artifactId>
  <version>${jboss-logging.version}</version>
</dependency>
```

### Customize Configuration Items

Add jbossts-properties.xml in classpath of the application to customize Narayana configuration.

Please refer to Narayana official documentation for more details.

### Configure XA Transaction Manager Type

Yaml:

```yaml
- !TRANSACTION
 defaultType: XA
 providerType: Narayana
```

SpringBoot:

```yaml
spring:
 shardingsphere:
  props:
   xa-transaction-manager-type: Narayana
```

Spring Namespace:

```xml
<shardingsphere:data-source id="xxx" data-source-names="xxx" rule-refs="xxx">
  <props>
    <prop key="xa-transaction-manager-type">Narayana</prop>
  </props>
</shardingsphere:data-source>
```

### Bitronix Transaction

#### Import Maven Dependency

```xml
<properties>
  <btm.version>2.1.3</btm.version>
</properties>

<dependency>
  <groupId>org.apache.shardingsphere</groupId>
  <artifactId>shardingsphere-jdbc-core</artifactId>
  <version>${shardingsphere.version}</version>
</dependency>

<dependency>
  <groupId>org.apache.shardingsphere</groupId>
  <artifactId>shardingsphere-transaction-xa-core</artifactId>
  <version>${shardingsphere.version}</version>
</dependency>

<dependency>
  <groupId>org.apache.shardingsphere</groupId>
  <artifactId>shardingsphere-transaction-xa-bitronix</artifactId>
  <version>${shardingsphere.version}</version>
</dependency>

<dependency>
  <groupId>org.codehaus.btm</groupId>
  <artifactId>btm</artifactId>
  <version>${btm.version}</version>
</dependency>
```

#### Customize Configuration Items

Please refer to Bitronix official documentation for more details.

#### Configure XA Transaction Manager Type

Yaml:

```yaml
- !TRANSACTION
 defaultType: XA
 providerType: Bitronix
```

SpringBoot:

```yaml
spring:
 shardingsphere:
  props:
   xa-transaction-manager-type: Bitronix
```

Spring Namespace:

```xml
<shardingsphere:data-source id="xxx" data-source-names="xxx" rule-refs="xxx">
  <props>
    <prop key="xa-transaction-manager-type">Bitronix</prop>
  </props>
</shardingsphere:data-source>
```

### Seata Transaction

#### Startup Seata Server

Download seata server according to seata-work-shop.

#### Create Undo Log Table

Create undo_log table in each physical database (sample for MySQL).

```sql
CREATE TABLE IF NOT EXISTS `undo_log`
(
  `id`            BIGINT(20)   NOT NULL AUTO_INCREMENT COMMENT 'increment id',
  `branch_id`     BIGINT(20)   NOT NULL COMMENT 'branch transaction id',
  `xid`           VARCHAR(100) NOT NULL COMMENT 'global transaction id',
  `context`       VARCHAR(128) NOT NULL COMMENT 'undo_log context,such as serialization',
  `rollback_info` LONGBLOB     NOT NULL COMMENT 'rollback info',
  `log_status`    INT(11)      NOT NULL COMMENT '0:normal status,1:defense status',
  `log_created`   DATETIME     NOT NULL COMMENT 'create datetime',
  `log_modified`  DATETIME     NOT NULL COMMENT 'modify datetime',
  PRIMARY KEY (`id`),
  UNIQUE KEY `ux_undo_log` (`xid`, `branch_id`)
) ENGINE = InnoDB
  AUTO_INCREMENT = 1
  DEFAULT CHARSET = utf8 COMMENT ='AT transaction mode undo table';
```

#### Update Configuration

Configure seata.conf file in classpath.

```
client {
    application.id = example  ## application unique ID
    transaction.service.group = my_test_tx_group   ## transaction group
}
```

Modify file.conf and registry.conf if needed.

## 7.1.10 Unsupported Items

#### DataSource Interface

- Do not support timeout related operations

#### Connection Interface

- Do not support operations of stored procedure, function and cursor
- Do not support native SQL
- Do not support savepoint related operations
- Do not support Schema/Catalog operation
- Do not support self-defined type mapping

### Statement and PreparedStatement Interface

- Do not support statements that return multiple result sets (stored procedures, multiple pieces of non-SELECT data)
- Do not support the operation of international characters

### ResultSet Interface

- Do not support getting result set pointer position
- Do not support changing result pointer position through none-next method
- Do not support revising the content of result set
- Do not support acquiring international characters
- Do not support getting Array

### JDBC 4.1

- Do not support new functions of JDBC 4.1 interface

For all the unsupported methods, please read org.apache.shardingsphere.driver.jdbc.unsupported package.

# 7.2 DBPlusEngine-Proxy

Configuration is the only module in DBPlusEngine-Proxy that interacts with application developers, through which developer can quickly and clearly understand the functions provided by DBPlusEngine-Proxy.

This chapter is a configuration manual for DBPlusEngine-Proxy, which can also be referred to as a dictionary if necessary.

DBPlusEngine-Proxy provided YAML configuration, and used DistSQL to communicate. By configuration, application developers can flexibly use data sharding, readwrite-splitting, data encryption, shadow database or the combination of them.

Rule configuration keeps consist with YAML configuration of DBPlusEngine-Driver. DistSQL and YAML can be replaced each other.

Please refer to Example for more details.

## 7.2.1 Use of License in DBPlusEngine-Proxy

### Background

| License Function | Description |
|---|---|
| limit duration | Limit the duration of DBPlusEngine. |
| limit resources | Limit the number of storage nodes created by the DBPlusEngine. |
| limit instances | Limit the number of instances created by [DBPlusEngine-Proxy]+[DBPlusEngine-Driver] in a single cluster. |
| limit versions | Limit the version of the DBPlusEngine instance in a stand-alone or cluster. |

### Notice

The default storage directory for the License is: config.

### Procedure

- Execute the following command to view the license.

```
SHOW LICENSE INFO;
```

- Use the registration code to register the license.

```
REGISTER LICENSE '[license-key]';
```

- Use the license file to register the license.

```
REGISTER LICENSE '[Direct path/license-file]';
```

- Use the registration code file to preview the license.

```
SHOW LICENSE INFO '[license-key]';
```

- Use the license file to preview the license.

```
SHOW LICENSE INFO '[Direct path/license-file]';
```

### Post-processing

If the following error occurs, please obtain the latest license and update the license.

- No license prompt

```
FATAL:  License not registered
```

- Incomplete license prompt

```
FATAL:  Incomplete license
```

- License expiration prompt

```
FATAL:  License is expired
```

- Illegal license prompt

```
ERROR:  The xxx exceeds the limit of the license
```

## 7.2.2 Startup

This chapter will introduce the deployment and startup of DBPlusEngine-Proxy.

### Use Binary Tar

#### Startup Steps

1. Download the latest version of ShardingSphere-Proxy.
2. After the decompression, revise conf/server.yaml and documents begin with config- prefix, conf/config-xxx.yaml for example, to configure sharding rules and readwrite-splitting rules. Please refer to Configuration Manual for the configuration method.
3. Please run bin/start.sh for Linux operating system; run bin/start.bat for Windows operating system to start ShardingSphere-Proxy. To configure start port and document location, please refer to Quick Start.

### Using database protocol

#### Using PostgreSQL

1. Use any PostgreSQL terminal to connect, such as psql -U root -h 127.0.0.1 -p 3307.

#### Using MySQL

1. Copy MySQL' s JDBC driver to folder ext-lib/.
2. Use any MySQL terminal to connect, such as mysql -u root -h 127.0.0.1 -P 3307.

#### Using openGauss

1. Copy openGauss' s JDBC driver whose package prefixed with org.opengauss to folder ext-lib/.
2. Use any openGauss terminal to connect, such as gsql -U root -h 127.0.0.1 -p 3307.

### Using metadata persist repository

#### Using ZooKeeper

Integrated ZooKeeper Curator client by default.

#### Using Etcd

1. Copy Etcd' s client driver to folder ext-lib/.

### Using Distributed Transaction

Same with ShardingSphere-JDBC. please refer to Distributed Transaction for more details.

### Using user-defined algorithm

When developer need to use user-defined algorithm, should use the way below to configure algorithm, use sharding algorithm as example.

1. Implement ShardingAlgorithm interface.
2. Create META-INF/services directory in the resources directory.
3. Create a new file org.apache.shardingsphere.sharding.spi.ShardingAlgorithm in the META-INF/services directory.
4. Absolute path of the implementation class are write to the file org.apache.shardingsphere.sharding.spi.ShardingAlgorithm
5. Package Java file to jar.
6. Copy jar to ShardingSphere-Proxy' s ext-lib/ folder.
7. Configure user-defined Java class into YAML file. Please refer to Configuration Manual for more details.

### Notices

1. DBPlusEngine-Proxy uses 3307 port in default. Users can start the script parameter as the start port number, like bin/start.sh 3308.
2. DBPlusEngine-Proxy uses conf/server.yaml to configure the registry center, authentication information and public properties.
3. DBPlusEngine-Proxy supports multi-logic data sources, with each yaml configuration document named by config- prefix as a logic data source.
4. DBPlusEngine-Proxy uses 0.0.0.0 as default listening address. Users can specify the listening address through the startup script, like bin/start.sh --help.

## Use Docker

### Background

This chapter introduces how to start DBPlusEngine-Proxy via Docker.

### Notice

Using Docker to start DBPlusEngine-Proxy does not require additional package supoort.

### Steps

1. Acquire Docker Image

- Method 1 (Recommended): pull from DockerHub

```
docker pull apache/shardingsphere-proxy
```

- Method 2: acquire latest master branch image master: https://github.com/apache/shardingsphere/pkgs/container/shardingsphere-proxy
- Method 3: build your own image

```
git clone https://github.com/apache/shardingsphere
mvn clean install
cd shardingsphere-distribution/shardingsphere-proxy-distribution
mvn clean package -Prelease,docker
```

If the following problems emerge, please make sure Docker daemon process is running.

```
I/O exception (java.io.IOException) caught when processing request to {}->unix://localhost:80: Connection refused ?
```

2. Configure conf/server.yaml and conf/config-*.yaml

Configuration file template can be attained from the Docker container and can be copied to any directory on the host:

```
docker run -d --name tmp --entrypoint=bash apache/shardingsphere-proxy
docker cp tmp:/opt/shardingsphere-proxy/conf /host/path/to/conf
docker rm tmp
```

Since the network conditions inside the container may differ from those of the host, if errors such as "cannot connect to the database" occur, please make sure that the IP of the database specified in the conf/config-*.yaml configuration file can be accessed from inside the Docker container.

For details, please refer to DBPlusEngine-Proxy quick start manual - Use Binary Tar.

4. Start the DBPlusEngine-Proxy container

Mount the conf and ext-lib directories from the host to the container. Start the container:

```
docker run -d \
   -v /host/path/to/conf:/opt/shardingsphere-proxy/conf \
   -v /host/path/to/ext-lib:/opt/shardingsphere-proxy/ext-lib \
   -e PORT=3308 -p13308:3308 apache/shardingsphere-proxy:latest
```

ext-lib is not necessary during the process. Users can mount it at will. DBPlusEngine-Proxy default port 3307 can be designated according to environment variable -e PORT Customized JVM related parameters can be set according to environment variable JVM_OPTS.

Note:

Support setting environment variable CGROUP_ MEM_ OPTS: used to set related memory parameters in the container environment. The default values in the script are:

```
-XX:InitialRAMPercentage=80.0 -XX:MaxRAMPercentage=80.0 -XX:MinRAMPercentage=80.0
```

5. Use Client to connect to DBPlusEngine-Proxy

Please refer to DBPlusEngine-Proxy quick start manual - Use Binary Tar.

### Configuration Example

For full configuration, please refer to the examples given in DBPlusEngine library: https://github.com/apache/shardingsphere/tree/master/examples/shardingsphere-proxy-example.

## Using Operator

### What is DBPlusEngine-Operator?

Kubernetes’ operator mode allows you to expand the ability of the cluster by associating controllers for one or more custom resources without modifying kubernetes’ own code. Operator is the kubernetes API client and acts as the custom resources controller.

The operator mode aims to capture the key objectives of the DevOps teams (who are managing one or a group of services).

DBPlusEngine-Operator helps users quickly deploy a set of DBPlusEngine-Proxy cluster in kubernetes environment, and is responsible for deploying and maintaining relevant resources around the cluster and monitoring the cluster status.

## Terms

### CRD

CRD (customresourcedefinition) user-defined resource definition means that DBPlusEngine-Operator will deploy a complete set of DBPlusEngine-Proxy clusters in kubernetes cluster by using CR (customresource) defined by CRD.

## Advantages

### Simplified configuration

You only need to write a simple yaml to deploy a complete set of DBPlusEngine-Proxy clusters in the cluster.

### Easy to expand

By modifying CR yaml, a series of features such as horizontal scaling can be used.

### Simple operation and maintenance

Using DBPlusEngine-Operator will not interfere with the status of DBPlusEngine-Proxy in the cluster. DBPlusEngine-Operator will automatically detect the status of the cluster and correct it.

## Architecture

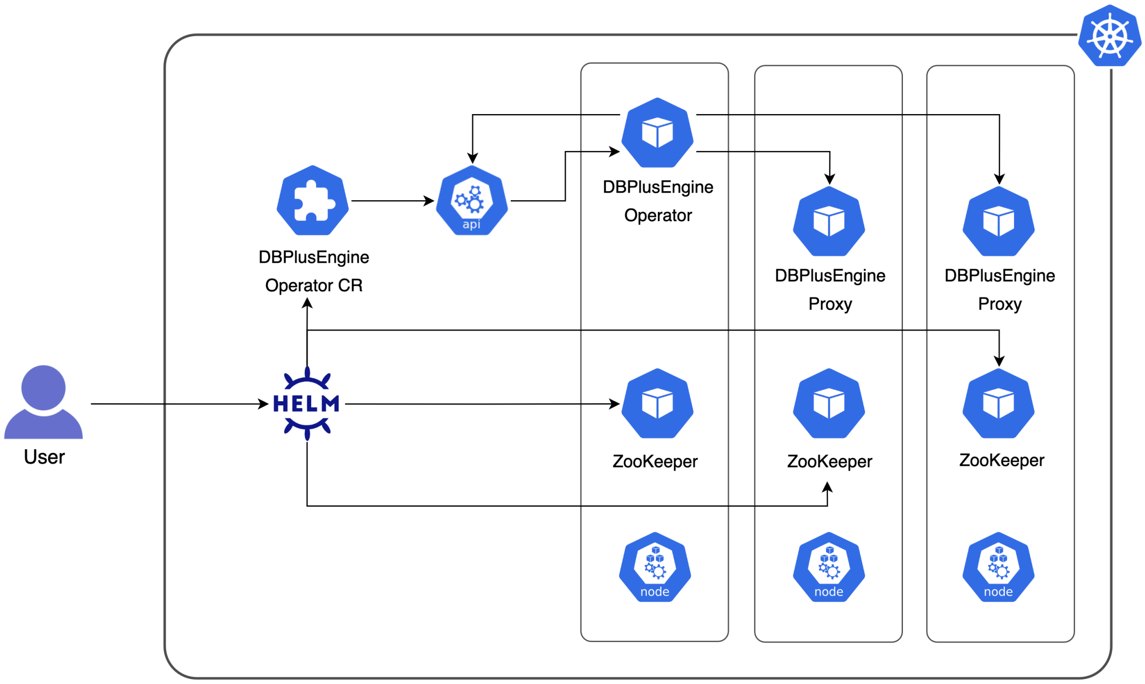


Fig. 1: Architecture

## Install DBPlusEngine-Operator

Configure DBPlusEngine-Operator Parameters (#DBPlusEngine-Operator Parameters), configuration file located in dbplusengine-operator/values.yaml.

Run

```
kubectl create ns  dbplusengine-operator
helm install dbplusengine-operator dbplusengine-operator -n dbplusengine-operator
```

## Install DBPlusEngine-Operator-Cluster cluster

Configure the DBPlusEngine-Operator-Cluster Parameters configuration file located in dbplusengine-proxy/values.yaml.

Move the sphere-ex.license to dbplusengine-proxy/license, and keep the name sphere-ex.license.

```
kubectl create ns  dbplusengine
helm install  dbplusengine-operator-cluster dbplusengine-operator-cluster -n dbplusengine
```

## DBPlusEngine-Operator Parameters

### DBPlusEngine-Operator parameters

| Name | Description | Value |
|---|---|---|
| replicaCount | operator replica count | 2 |
| image.repository | operator image name | sphereex/dbp lusengine-operator |
| image.pullPolicy | mirror pull policy | IfNotPresent |
| image.tag | image tag | 0.0.1 |
| imagePullSecrets | image pulls key of private repository | [] |
| resources | resources required by the operator | {} |
| webhook.port | operator webhook boot port | 9443 |
| heal th.healthProbePort | operator health check port | 8081 |

### DBPlusEngine-Operator-Cluster Parameters

## DBPlusEngine-Operator-Cluster parameters

| Name | Description | Value |
|---|---|---|
| re plicaCount | DBPlusEngine-Operator-Cluster cluster starts the number of replicas, Note: after you enable automaticScaling, this parameter will no longer take effect | "1" |
| a utomaticScaling.enable | Whether the DBPlusEngine-Operator-Cluster cluster has auto-scaling enabled | false |
| automatic Scaling.scal eUpWindows | DBPlusEngine-Operator-Cluster automatically scales the stable window | 30 |
| ` automaticScaling.scaleDownWindows` | DBPlusEngine-Operator-Cluster automatically shrinks the stabilized window | 30 |
| a utomaticScaling.target | DBPlusEngine-Operator-Cluster auto-scaling threshold, the value is a percentage. Note: at this stage, only cpu is supported as a metric for scaling | 70 |
| automa ticScaling.m axInstance | DBPlusEngine-Operator-Cluster maximum number of scaled-out replicas | 4 |
| automa ticScaling.m inInstance | DBPlusEngine-Operator-Cluster has a minimum number of boot replicas, and the shrinkage will not be less than this number of replicas | 1 |
| imag e.registry | DBPlusEngine-Operator-Cluster image host | `` docker.io`` |
| image. repository | DBPlusEngine-Operator-Cluster image repository name | sphereex/dbplusengine-proxy |
| ` image.tag` | DBPlusEngine-Operator-Cluster image tag | 5.1.2 |
| ` resources` | DBPlusEngine-Operator-Cluster starts the requirement resource, and after opening automaticScaling, the resource of the request multiplied by the percentage of target is used to trigger the scaling action | {} |
| se rvice.type | DBPlusEngine-Operator-Cluster external exposure mode | `` ClusterIP`` |
| se rvice.port | DBPlusEngine-Operator-Cluster exposes the port to the outside world | 3307 |
| ` startPort` | DBPlusEngine-Operator-Cluster boot port | 3307 |
| imageP ullSecrets | DBPlusEngine-Operator-Cluster private image repository key | [] |
| mySQLDriv er.version | The DBPlusEngine-Operator-Cluster mysql driver version will not be downloaded if it is empty | "" |
| GN.mode | DBPlusEngine-Operator-Cluster governance center mode, supporting sidecar/zookeeper | `` zookeeper`` |
| GN.Sidec arRegistry | DBPlusEngine-Operator-Cluster sidecar mode image host | <i mage wareho use host> |
| GN.Sidecar Repository | DBPlusEngine-Operator-Cluster sidecar mode image warehouse name | sphereex/dbplusengine-sidecar |
| GN. SidecarTag | DBPlusEngine-Operator-Cluster sidecar mode image tag | 0.2.0 |
| GN.sidecar ServerAddr | DBPlusEngine-Operator-Cluster sidecar address of mode image server | Serve r Address |
| ` withAgent` | DBPlusEngine-Operator-Cluster whether start agent parameter | false |

## Compute Node DBPlusEngine-Operator-Cluster Server Authority Configuration Items

| Name | Description | Value |
|---|---|---|
| ` serverConfig.author ity.privilege.type` | The authority provider type for storage node data authorization, the default value is ALL_PERMITTED | ALL_PE RMITTED |
| se rverConfig. authority .users[0]. password | The password used to login to the compute node | root |
| serverConfig.autho rity.users[0].user | The username used to login to the compute node, the authorized host. Format: @ hostname as % or an empty string indicates no restriction on the authorized host | ` root@%` |

## Compute Node DBPlusEngine-Operator-Cluster Server Mode Configuration Items

| Name | Des cription | Value |
|---|---|---|
| serverConfig.mode.type | The running mode type. At this stage, only cluster mode is s upported | Cluster |
| serverConfig.mode. repository. props.namespace | Registry center n amespace | governance_ds |
| serverConfig.mode.rep ository. props.server-lists | Registry center co nnection address | {{ printf "%s-zook eeper.%s:2181" . Release.Na me .Release.Namespace }} |
| serverConfig.mode.r epository. props.maxRetries | The maximum number of client con nections retries | 3 |
| serverCon fig.mode.repository. props.op erationTimeoutMilliseconds | The number of mill iseconds that the client o peration timed out | 5000 |
| server Config.mode.repository. props .retryIntervalMilliseconds | The number of mill iseconds between retries | 500 |
| serverConfig.mode.reposito ry. props.timeToLiveSeconds | The number of seconds that t emporary data inv alidated | 60 |
| serverC onfig.mode.repository. type | Persist re pository type. Only Z ooKeeper is s upported at this stage | ZooKeeper |

## Governance Node ZooKeeper Configuration Item

| Configuration Item | Description | Value |
|---|---|---|
| zookeeper.enabled | Used to switch whether use ZooKeeper chart | ` true` |
| zoo keeper.replicaCount | Number of ZooKeeper nodes | 1 |
| zookeeper. persistence. enabled | Identifies whether ZooKeeper uses PersistentVolumeClaim to apply for PersistentVolume | `` false`` |
| zookeeper.persi stence. storageClass | StorageClass for PersistentVolume | "" |
| zookeeper.pers istence. accessModes | Access mode of PersistentVolume | ["Rea dWriteO nce"] |
| zookeep er.persistence.size | PersistentVolume size | 8Gi |

## Sample

### dbplusengine-operator/values.yaml

```yaml
## @section DBPlusEngine-Operator-Cluster operator parameters
## @param replicaCount operator replica count
##
replicaCount: 2
image:
  ## @param image.repository operator image name
  ##
  repository: "sphere-ex/dbplusengine-operator"
  ## @param image.pullPolicy image pull strategy
  ##
  pullPolicy: IfNotPresent
  # Overrides the image tag whose default is the chart appVersion.
  ## @param image.tag image tag
  ##
  tag: "0.1.0"
## @param imagePullSecrets Private warehouse image pull key
## e.g:
## imagePullSecrets:
##   - name: mysecret
##
imagePullSecrets: []
## @param resources operator required resources
## e.g:
## resources:
##   limits:
##     cpu: 2
##   limits:
##     cpu: 2
##
resources: {}
## @param webhook.port operator webhook boot port
##
webhook:
  port: 9443
## @param health.healthProbePort operator health check port
##
health:
  healthProbePort: 8081
```

### dbplusengine-proxy/values.yaml

```yaml
## @section DBPlusEngine-Operator-Cluster cluster parameters
## @param replicaCount DBPlusEngine-Operator-Cluster The number of cluster startup copies. Note: this parameter will no longer take effect after automaticScaling is enabled.
##
replicaCount: "1"
## @param automaticScaling.enable DBPlusEngine-Operator-Cluster Whether the cluster starts automatic capacity expansion and contraction
## @param automaticScaling.scaleUpWindows DBPlusEngine-Operator-Cluster Auto expansion stable window
## @param automaticScaling.scaleDownWindows DBPlusEngine-Operator-Cluster Auto shrink stable window
## @param automaticScaling.target DBPlusEngine-Operator-Cluster The threshold value of automatic capacity expansion and contraction is percentage. Note: at this stage, only CPU is supported for capacity expansion and contraction.
## @param automaticScaling.maxInstance DBPlusEngine-Operator-Cluster Maximum number of capacity expansion copies
## @param automaticScaling.minInstance DBPlusEngine-Operator-Cluster The minimum number of startup copies, and the shrink size will not be less than this number of copies.
##
automaticScaling:
  enable: false
  scaleUpWindows: 30
  scaleDownWindows: 30
  target: 20
  maxInstance: 4
  minInstance: 1
## @param image.registry DBPlusEngine-Operator-Cluster image host
## @param image.repository DBPlusEngine-Operator-Cluster Image warehouse name
## @param image.tag DBPlusEngine-Operator-Cluster image tag
##
image:
  registry: <image warehouse host>
  repository: sphere-ex/dbplusengine-proxy
  tag: 1.1.0
withAgent: false
## @param resources DBPlusEngine-Operator-Cluster Start the demand resource. After automaticscaling is enabled, multiply the resource of request by the percentage of target as the actual utilization rate to trigger the expansion and contraction action.
## e.g:
## resources:
##   limits:
##     cpu: 2
##   requests:
##     cpu: 2
##
resources: {}
## @param service.type DBPlusEngine-Operator-Cluster external exposure mode
## @param service.port DBPlusEngine-Operator-Cluster external exposure port
##
service:
  type: ClusterIP
  port: 3307
## @param startPort DBPlusEngine-Operator-Cluster startup port
##
startPort: 3307
## @param imagePullSecrets DBPlusEngine-Operator-Cluster private image warehouse key
## e.g:
## imagePullSecrets:
##   - name: mysecret
##
imagePullSecrets: []
## @param mySQLDriver.version DBPlusEngine-Operator-Cluster mysql driver version. If it is empty, the driver will not be downloaded.
##
mySQLDriver:
  version: ""
## @section  DBPlusEngine-Operator-Cluster ServerConfiguration parameters
## NOTE: If you use the sub-charts to deploy Zookeeper, the server-lists field must be "{{ printf \"%s-zookeeper.%s:2181\" . Release.Name .Release.Namespace }}",
## otherwise please fill in the correct zookeeper address
## The server.yaml is auto-generated based on this parameter.
## If it is empty, the server.yaml is also empty.
## ref: https://shardingsphere.apache.org/document/current/en/user-manual/shardingsphere-jdbc/yaml-config/mode/
## ref: https://shardingsphere.apache.org/document/current/en/user-manual/common-config/builtin-algorithm/metadata-repository/
##
serverConfig:
  ## @section Compute-Node DBPlusEngine-Operator-Cluster ServerConfiguration authority parameters
  ## NOTE: It is used to set up initial user to login compute node, and authority data of storage node.
  ## @param serverConfig.authority.privilege.type authority provider for storage node, the default value is ALL_PERMITTED
  ## @param serverConfig.authority.users[0].password Password for compute node.
  ## @param serverConfig.authority.users[0].user Username,authorized host for compute node. Format: <username>@<hostname> hostname is % or empty string means do not care about authorized host
  ##
  authority:
```

```
  privilege:
   type: ALL_PERMITTED
  users:
   - password: root
    user: root@%
 ## @section Compute-Node DBPlusEngine-Operator-Cluster ServerConfiguration mode Configuration parameters
 ## @param serverConfig.mode.type Type of mode configuration. Now only support Cluster mode
 ## @param serverConfig.mode.repository.props.namespace Namespace of registry center
 ## @param serverConfig.mode.repository.props.server-lists Server lists of registry center
 ## @param serverConfig.mode.repository.props.maxRetries Max retries of client connection
 ## @param serverConfig.mode.repository.props.operationTimeoutMilliseconds Milliseconds of operation timeout
 ## @param serverConfig.mode.repository.props.retryIntervalMilliseconds Milliseconds of retry interval
 ## @param serverConfig.mode.repository.props.timeToLiveSeconds Seconds of ephemeral data live
 ## @param serverConfig.mode.repository.type Type of persist repository. Now only support ZooKeeper
 ## @param serverConfig.mode.overwrite Whether overwrite persistent configuration with local configuration
 ##
 ##
 ## mode:
 ##  repository:
 ##   props:
 ##    namespace: ssd1031test1
 ##    server-lists: "127.0.0.1:21506"
 ##   type: SphereEx:MATE
 ##  type: Cluster
 mode:
  overwrite: true
  repository:
   props:
    maxRetries: 3
    namespace: governance_ds
    operationTimeoutMilliseconds: 5000
    retryIntervalMilliseconds: 500
    server-lists: "{{ printf \"%s-zookeeper.%s:2181\" .Release.Name .Release.Namespace }}"
    timeToLiveSeconds: 600
   type: ZooKeeper
  type: Cluster
 props:
  proxy-frontend-database-protocol-type: PostgreSQL
## @section ZooKeeper chart parameters

## ZooKeeper chart configuration
## https://github.com/bitnami/charts/blob/master/bitnami/zookeeper/values.yaml
##
zookeeper:
 ## @param zookeeper.enabled Switch to enable or disable the ZooKeeper helm chart
 ##
 enabled: true
 ## @param zookeeper.replicaCount Number of ZooKeeper nodes
 ##
 replicaCount: 3
 ## ZooKeeper Persistence parameters
 ## ref: https://kubernetes.io/docs/user-guide/persistent-volumes/
 ## @param zookeeper.persistence.enabled Enable persistence on ZooKeeper using PVC(s)
 ## @param zookeeper.persistence.storageClass Persistent Volume storage class
 ## @param zookeeper.persistence.accessModes Persistent Volume access modes
 ## @param zookeeper.persistence.size Persistent Volume size
 ##
 persistence:
  enabled: false
  storageClass: ""
  accessModes:
   - ReadWriteOnce
  size: 8Gi
```

### Clean

```
helm uninstall dbplusengine-proxy -n dbplusengine

helm uninstall dbplusengine-operator -n dbplusengine-operator

kubectl delete crd clusters.dbplusengine.sphere-ex.com \
proxyconfigs.dbplusengine.sphere-ex.com \
plocks.dbplusengine.sphere-ex.com \
pmetadata.dbplusengine.sphere-ex.com \
pnodes.dbplusengine.sphere-ex.com \
ppipelines.dbplusengine.sphere-ex.com \
psys.dbplusengine.sphere-ex.com \
pworkids.dbplusengine.sphere-ex.com
```

### Related References

- Feature Description of Auto-Scaling on Cloud (HPA)
- Feature Configuration of Auto-Scaling on Cloud (HPA)

### Add dependencies

This chapter mainly introduces how to download optional dependencies of DBPlusEngine.

### Add Bitronix dependencies

### Add Bitronix dependencies

Adding Bitronix dependencies requires downloading the following jar files and adding them under ext-lib path.

### jar file downloads

- btm-2.1.3.jar
- shardingsphere-transaction-xa-bitronix.jar

Please download the corresponding shardingsphere-transaction-xa-bitronix.jar file according to the proxy version.

### Add Narayana dependencies

### Add Narayana dependencies

Adding Narayana dependencies requires downloading the following jar files and adding them under ext-lib path.

### jar file downloads

- arjuna-5.12.4.Final.jar
- common-5.12.4.Final.jar
- javax.activation-api-1.2.0.jar
- jaxb-api-2.3.0.jar
- jaxb-core-2.3.0.jar
- jaxb-impl-2.3.0.jar
- jboss-connector-api_1.7_spec-1.0.0.Final.jar
- jboss-logging-3.2.1.Final.jar
- jboss-transaction-api_1.2_spec-1.0.0.Alpha3.jar
- jboss-transaction-spi-7.6.0.Final.jar
- jta-5.12.4.Final.jar
- narayana-jts-integration-5.12.4.Final.jar
- shardingsphere-transaction-xa-narayana.jar

Please download the corresponding shardingsphere-transaction-xa-narayana.jar file according to the proxy version.

## 7.2.3 Yaml Configuration

The YAML configuration of DBPlusEngine-Driver is the subset of DBPlusEngine-Proxy. In server.yaml file, DBPlusEngine-Proxy can configure the authority feature and more properties for Proxy only.

Note: the YAML configuration file supports more than 3MB of configuration content.

This chapter will introduce the extra YAML configuration of DBPlusEngine-Proxy.

### Login Authentication

### Password Authentication

DBPlusEngine-Proxy uses password authentication by default. The configuration format is as follows:

```yaml
authority:
 users:
  - user: root@%
    password: root
  - user: sharding
    password: sharding
```

In this configuration, two users are specified for DBPlusEngine:

- root: @% means that the user can access DBPlusEngine through any host, password specifies the password as root.
- sharding: the user does not specify the host configuration, and the default value is @%, password specifies the password as sharding.

When the administrator needs to restrict specific users from logging in to the host, you can use the username@host to specify, such as:

```yaml
- user: user1@192.168.1.111
  password: user1_password
```

Indicates that user1 can access DBPlusEngine only through 192.168.1.111, the authentication password is user1_password.

### LDAP Authentication

Notes: - Before enabling LDAP authentication, users should first deploy an LDAP server, such as OpenLDAP. - When using the MySQL client, the cleartext-plugin needs to be displayed, such as: mysql -h 127.0.0.1 -P 3307 -u root -p –enable-cleartext-plugin

Configure LDAP in DBPlusEngine as follows:

#### Example 1

Each user needs to be authenticated with LDAP and use the same DN template.

```yaml
authority:
  users:
    - user: root@%
    - user: sharding
  authenticators:
    auth_ldap:
      type: LDAP
      props:
        ldap_server_url: ldap://localhost:389
        ldap_dn_template: cn={0},ou=users,dc=example,dc=org
  defaultAuthenticator: auth_ldap
```

This configuration specifies an authenticator auth_ldap, whose type is LDAP, and the necessary configuration is given in props:

- ldap_server_url: access address of LDAP server
- ldap_dn_template: user DN template

When using the above configuration, the corresponding user DN are root and sharding:

- root： cn=root,ou=users,dc=example,dc=org
- sharding： cn=sharding,ou=users,dc=example,dc=org

#### Example 2

Each user needs LDAP authentication, but uses a different DN template.

```yaml
authority:
  users:
    - user: root@%
      props:
        ldap_dn: cn=root,ou=admin,dc=example,dc=org
    - user: sharding
  authenticators:
    auth_ldap:
      type: LDAP
      props:
        ldap_server_url: ldap://localhost:389
        ldap_dn_template: cn={0},ou=users,dc=example,dc=org
  defaultAuthenticator: auth_ldap
```

The difference from ‘example 1’ is: User ‘root’ is not in the same ou as other users, so an explicit user DN is specified separately for ‘root’ .

When using the above configuration, the root and sharding of DN are: - root: cn=root,ou=admin,dc=example,dc=org - sharding: cn=sharding,ou=users,dc=example,dc=org

### Hybrid Authentication

Hybrid authentication means that some users use password authentication and others use LDAP authentication. This is a very flexible combination that can meet the needs of specific security scenarios.

The configuration format of hybrid authentication is as follows:

```
authority:
 users:
  - user: root@%
   auth: auth_ldap
  - user: sharding
   password: sharding
  - user: user1
   password: password_user1
 authenticators:
  auth_ldap:
   type: LDAP
   props:
    ldap_server_url: ldap://localhost:389
    ldap_dn_template: cn={0},ou=users,dc=example,dc=org
```

In the above configuration, defaultAuthenticator is not specified, and password authentication is used by default. At the same time, through display configuration auth: auth_ldap, which specifies the identity authenticator for the user 'root', and requires the user to log in through LDAP authentication.

When using the above configuration, the corresponding authentication methods for users 'root', 'sharding' and 'user1' are:

- root: LDAP
- sharding: password
- user1: password

Note: in the hybrid authentication scenario, the administrator can also enable LDAP authentication by default and use auth: password to set a small number of users to password authentication.

### Authority

It is used to set up initial user to login compute node, and authority data of storage node.

### Configuration Item Explanation

```
authority:
 users:
  - user: # Specify the username, and authorized host for logging in to the compute node. Format: <username>@<hostname>. When the hostname is % or an empty string, it indicates that the authorized host is not limited.
   password: # Password
 privilege:
  type: # Privilege provider type. The default value is ALL_PERMITTED.
```

### Example

#### ALL_PERMITTED

```yaml
authority:
 users:
  - user: root@localhost
   password: root
  - user: my_user@
   password: pwd
 privilege:
  type: ALL_PERMITTED
```

The above configuration indicates: - The user root can connect to Proxy only through localhost, and the password is root. - The user my_user can connect to Proxy through any host, and the password is pwd. - The privilege type is ALL_PERMITTED, which indicates that users are granted all authorities by default without authentication.

#### DATABASE_PERMITTED

```yaml
authority:
 users:
  - user: root@localhost
   password: root
  - user: my_user@
   password: pwd
 privilege:
  type: DATABASE_PERMITTED
  props:
   user-database-mappings: root@=sharding_db, root@=test_db, my_user@127.0.0.1=sharding_db
```

The above configuration means: - The user root can access sharding_db when connecting from any host - The user root can access test_db when connecting from any host - The user my_user can access sharding_db only when connected from 127.0.0.1

Refer to Authority Provider for more implementations.

### Traffic Dual Routing

#### Instruction for Use

The traffic dual routing function needs to use the hybrid deployment architecture, deploy DBPlusEngine-Driver and DBPlusEngine-Proxy at the same time, and uniformly manage the functional such as sharding, encryption and decryption or read-write splitting through the registry

Because the Traffic Dual Routing must be configured with DBPlusEngine-Proxy, traffic rules can only be added through YAML configuration or DistSQL at the access end of DBPlusEngine-Proxy. In addition, in order to cooperate with the traffic function, the DBPlusEngine-Proxy access terminal needs to configure the labels tag for the configuration of traffic forwarding.

### Configuration Item Description

```yaml
rules:
- !TRAFFIC
 trafficStrategies:
  sql_match_traffic:
   labels:
    - OLTP
   algorithmName: sql_match_algorithm
   loadBalancerName: random_load_balancer
  sql_regex_traffic:
   labels:
    - OLTP
   algorithmName: sql_regex_algorithm
   loadBalancerName: random_load_balancer
  sql_hint_traffic:
   labels:
    - OLAP
   algorithmName: sql_hint_algorithm
   loadBalancerName: round_robin_load_balancer
  transaction_traffic:
   # Optional configuration. The algorithms are JDBC and FIRST_SQL does not need to be configured.
   labels:
    - OLAP
    - OLTP
   algorithmName: transaction_algorithm
   # Optional configuration. The algorithms are JDBC and FIRST_SQL does not need to be configured.
   loadBalancerName: round_robin_load_balancer
 trafficAlgorithms:
  sql_match_algorithm:
   type: SQL_MATCH
   props:
    sql: SELECT * FROM t_order WHERE content IN (?, ?); UPDATE t_order SET creation_date = NOW() WHERE user_id = 1;
  sql_regex_algorithm:
   type: SQL_REGEX
   props:
    regex: (?i)^(UPDATE|SELECT).*WHERE user_id.*
  sql_hint_algorithm:
   type: SQL_HINT
  transaction_algorithm:
   # Support FIRST_SQL, JDBC and proxy
   # FIRST_ SQL will determine the forwarding result of the transaction unit according to the forwarding result of the first SQL
   # JDBC will execute the transaction unit on JDBC without forwarding
   # Proxy will forward the transaction unit to the proxy instance. In order to ensure the consistency of data, the transaction unit will execute on the same instance.
   type: PROXY
 loadBalancers:
  random_load_balancer:
   type: RANDOM
  round_robin_load_balancer:
   type: ROUND_ROBIN

labels:
 - OLTP
```

YamlTrafficRuleConfiguration configuration item description:

| Name | Data Type | Description |
|---|---|---|
| trafficStrategies | Map<String, YamlTrafficS trategyConfiguration> | Forwarding policy, required. By default, they are matched in the order of configured policies. The first matching policy is the target policy. If the user has configured the transaction forwarding policy, the transaction forwarding policy will be matched first. |
| trafficAlgorithms | Map<String, YamlShardingSphereAl gorithmConfiguration> | Forwarding matching algorithm, required. |
| loadBalancers | Map<String, YamlShardingSphereAl gorithmConfiguration> | Forwarding load balancing policy, required. |

YamlTrafficStrategyConfiguration configuration item description:

| Name | Data Type | Description |
|---|---|---|
| name | String | Forwarding policy name, required. |
| labels | Collection | Forward the Proxy instance tag when algorithmName is configured as JDBC or FIRST_SQL, no configuration is required, and other algorithms are required. |
| algorithmName | String | Forwarding matching algorithm, required. |
| loadBalancerName | String | Forwarding load balancing algorithm, when algorithmname is configured as JDBC or FIRST_SQL, no configuration is required, and other algorithms are required. |

### Built-in Forwarding Matching Algorithm

#### Hint Based Forwarding Matching Algorithm

#### Hint Forwarding Matching Algorithm Based on SQL

Type: SQL_HINT Configurable properties:

The SQL Hint function requires the user to turn on the configuration of parsing comments in advance, set sqlCommentParseEnabled to true, and the comment format only supports /* */ temporarily. The content needs to start with ShardingSphere hint:. The user can turn on and off the SQL hint forwarding matching algorithm by setting the useTraffic=true or useTraffic=false.

#### Forwarding Matching Algorithm Based on Segment (SegmentTrafficAlgorithm)

#### String Forwarding Matching Algorithm Based on SQL

String forwarding matching algorithm based on SQL

Type: SQL_MATCH

Configurable properties:

| Attribute Name | Data Type | Description |
|---|---|---|
| sql | String | It is used to configure SQL strings. Multiple SQL strings are separated by semicolons. The SQL string matching algorithm ignores case and blank characters. |

### Regular Forwarding Matching Algorithm Based on SQL String

Type: SQL_REGEX

Configurable properties:

| Name | Data Type | Description |
|---|---|---|
| regex | String | Used to configure regular expressions. |

### TransactionTrafficAlgorithm

### Forwarding Matching Algorithm Based on the First SQL

Type: FIRST_SQL

Configurable attribute: None

### Unified Forwarding JDBC Matching Algorithm

Type: JDBC

Configurable attribute: None

### Unified Forwarding Proxy Matching Algorithm

Type: PROXY

Configurable attribute: None

### Built-in Forwarding Load Balancing Algorithm

### TrafficLoadBalanceAlgorithm

### Load balancing Algorithm for Random Proxy Instances

Type: RANDOM

Configurable attribute: None

### Load Balancing Algorithm for Polling Proxy Instances

Type: ROUND_ROBIN

Configurable attribute: None

### Related Reference

Feature Description of Traffic Rule

## Properties

### Introduction

DBPlusEngine provides the way of property configuration to configure system level configuration.

## Configuration Item Explanation

| Name | DataType | Description | DefaultValue | DynamicUpdate |
|---|---|---|---|---|
| sql-show (?) | boolean | Whether show SQL or not in log. Print SQL details can help developers debug easier. The log details include: logic SQL, actual SQL and SQL parse result. Enable this property will log into log topic ShardingSphere-SQL, log level is INFO. | false | true |
| sql-simple (?) | boolean | Whether show SQL details in simple style. | false | true |
| kernel-executor-size (?) | int | The max thread size of worker group to execute SQL. One ShardingSphere-DataSource will use a independent thread pool, it does not share thread pool even different data source in same JVM. | infinite | false |
| max-connections-size-per-query (?) | int | Max opened connection size for each query. | 1 | true |
| check-table-metadata-enabled (?) | boolean | Whether validate table meta data consistency when application startup or updated. | false | false |
| proxy-frontend-flush-threshold (?) | int | Flush threshold for every records from databases for ShardingSphere-Proxy. | 128 | true |
| proxy-hint-enabled (?) | boolean | Whether enable hint for ShardingSphere-Proxy. Using Hint will switch proxy thread mode from IO multiplexing to per connection per thread, which will reduce system throughput. | false | true |
| proxy-backend-query-fetch-size (?) | int | Proxy backend query fetch size. A larger value may increase the memory usage of ShardingSphere Proxy. The default value is -1, which means set the minimum value | 1 | false |

Properties can be updated by DistSQL#RAL. Dynamic update can take effect immediately, static update can take effect after restarted.

### 7.2.4 DistSQL

This chapter will introduce the detailed syntax of DistSQL.

#### Syntax

This chapter describes the syntax of DistSQL in detail, and introduces use of DistSQL with practical examples.

#### Syntax Rule

In DistSQL statement, except for keywords, the input format of other elements shall conform to the following rules.

#### Identifier

1. The identifier represents an object in the SQL statement, including:

- database name
- table name
- column name
- index name
- resource name
- rule name
- algorithm name

2. The allowed characters in the identifier are: [A-Z, A-Z, 0-9, _] (letters, numbers, underscores) and should start with a letter.
3. When keywords or special characters appear in the identifier, use the backticks (`).

#### Literal

Types of literals include:

- string: enclosed in single quotes (' ) or double quotes ( “)
- int: it is generally a positive integer, such as 0-9;

Note: some DistSQL syntax allows negative values. In this case, a negative sign (-) can be added before the number, such as -1.

- boolean, containing only true & false. Case insensitive.

## RDL Syntax

RDL (Resource & Rule Definition Language) responsible for definition of resources/rules.

### Storage Unit Definition

#### Syntax

```
REGISTER STORAGE UNIT storageUnitDefinition [, storageUnitDefinition] ...

ALTER STORAGE UNIT storageUnitDefinition [, storageUnitDefinition] ...

UNREGISTER STORAGE UNIT storageUnitName [, storageUnitName] ... [ignore single tables]

storageUnitDefinition:
  simpleSource | urlSource

simpleSource:
  storageUnitName(HOST=hostname,PORT=port,DB=dbName,USER=user [,PASSWORD=password] [,PROPERTIES(property [,
property]) ...])

urlSource:
  storageUnitName(URL=url,USER=user [,PASSWORD=password] [,PROPERTIES(property [,property]) ...])

property:
  key=value
```

#### Parameters Explained

| Name | DataType | Description |
|---|---|---|
| storageUnitName | IDENTIFIER | Storage unit name |
| hostname | STRING | Host or IP |
| port | INT | Port |
| dbName | STRING | DB name |
| url | STRING | URL |
| user | STRING | username |
| password | STRING | password |

#### Notes

- Before adding storage unit, please confirm that a distributed database has been created, and execute the use command to successfully select a database;
- Confirm that the storage unit to be added or altered can be connected, otherwise the operation will not be successful;
- Duplicate storageUnitName is not allowed;
- PROPERTIES is used to customize connection pool parameters, key and value are both STRING types;
- ALTER STORAGE UNIT is not allowed to change the real data source associated with this storage unit;
- ALTER STORAGE UNIT will switch the connection pool. This operation may affect the ongoing business, please use it with caution;
- UNREGISTER STORAGE UNIT will only delete logical storage unit, not real data sources;
- Storage unit referenced by rules cannot be deleted;

- If the storage unit is only referenced by single table rule, and the user confirms that the restriction can be ignored, the optional parameter ignore single tables can be added to perform forced deletion.

### Example

```
REGISTER STORAGE UNIT ds_0 (
    HOST="127.0.0.1",
    PORT=3306,
    DB="db0",
    USER="root",
    PASSWORD="root"
),ds_1 (
    HOST="127.0.0.1",
    PORT=3306,
    DB="db1",
    USER="root"
),ds_2 (
    HOST="127.0.0.1",
    PORT=3306,
    DB="db2",
    USER="root",
    PROPERTIES("maximumPoolSize"="10")
),ds_3 (
    URL="jdbc:mysql://127.0.0.1:3306/db3?serverTimezone=UTC&useSSL=false",
    USER="root",
    PASSWORD="root",
    PROPERTIES("maximumPoolSize"="10","idleTimeout"="30000")
);

ALTER STORAGE UNIT ds_0 (
    HOST="127.0.0.1",
    PORT=3309,
    DB="db0",
    USER="root",
    PASSWORD="root"
),ds_1 (
    URL="jdbc:mysql://127.0.0.1:3309/db1?serverTimezone=UTC&useSSL=false",
    USER="root",
    PASSWORD="root",
    PROPERTIES("maximumPoolSize"="10","idleTimeout"="30000")
);

UNREGISTER STORAGE UNIT ds_0, ds_1;
UNREGISTER STORAGE UNIT ds_2, ds_3 ignore single tables;
```

## Rule Definition

This chapter describes the syntax of rule definition.

### Sharding

#### Syntax

#### Sharding Table Rule

```
CREATE SHARDING TABLE RULE shardingTableRuleDefinition [, shardingTableRuleDefinition] ...

ALTER SHARDING TABLE RULE shardingTableRuleDefinition [, shardingTableRuleDefinition] ...

DROP SHARDING TABLE RULE tableName [, tableName] ...

CREATE DEFAULT SHARDING shardingScope STRATEGY (shardingStrategy)

ALTER DEFAULT SHARDING shardingScope STRATEGY (shardingStrategy)

DROP DEFAULT SHARDING shardingScope STRATEGY;

DROP SHARDING ALGORITHM algorithmName [, algorithmName] ...

DROP SHARDING KEY GENERATOR [IF EXISTS] keyGeneratorName [, keyGeneratorName] ...

CREATE SHARDING AUDITOR auditorDefinition [, auditorDefinition] ...

ALTER SHARDING AUDITOR auditorDefinition [, auditorDefinition] ...

DROP SHARDING AUDITOR [IF EXISTS] auditorName [, auditorName] ...

shardingTableRuleDefinition:
  shardingAutoTableRule | shardingTableRule

shardingAutoTableRule:
  tableName(storageUnits, shardingColumn, algorithmDefinition [, keyGenerateDefinition] [, auditDeclaration])

shardingTableRule:
  tableName(dataNodes [, databaseStrategy] [, tableStrategy] [, keyGenerateDefinition] [, auditDeclaration])

storageUnits:
  STORAGE_UNITS(storageUnit [, storageUnit] ...)

dataNodes:
  DATANODES(dataNode [, dataNode] ...)

storageUnit:
  storageUnitName | inlineExpression

dataNode:
  dataNodeName | inlineExpression

shardingColumn:
  SHARDING_COLUMN=columnName

algorithmDefinition:
  TYPE(NAME=shardingAlgorithmType [, PROPERTIES([algorithmProperties])])

keyGenerateDefinition:
  KEY_GENERATE_STRATEGY(COLUMN=columnName, strategyDefinition)
```

```
auditDeclaration:
  auditDefinition | auditStrategy

auditDefinition:
  AUDIT_STRATEGY([(singleAuditDefinition),(singleAuditDefinition)], ALLOW_HINT_DISABLE=true)

singleAuditDefinition:
  NAME=auditor1, algorithmDefinition

auditStrategy:
  AUDIT_STRATEGY(AUDITORS=[auditor1,auditor2], ALLOW_HINT_DISABLE=true)

shardingScope:
  DATABASE | TABLE

databaseStrategy:
  DATABASE_STRATEGY(shardingStrategy)

tableStrategy:
  TABLE_STRATEGY(shardingStrategy)

shardingStrategy:
  TYPE=strategyType, shardingColumn, shardingAlgorithm

shardingAlgorithm:
  SHARDING_ALGORITHM(algorithmDefinition)

strategyDefinition:
  TYPE(NAME=keyGenerateStrategyType [, PROPERTIES([algorithmProperties])])

shardingAlgorithmDefinition:
  shardingAlgorithmName(algorithmDefinition)

algorithmProperties:
  algorithmProperty [, algorithmProperty] ...

algorithmProperty:
  key=value

keyGeneratorDefinition:
  keyGeneratorName (algorithmDefinition)

auditorDefinition:
  auditorName (auditorAlgorithmDefinition)

auditorAlgorithmDefinition:
  TYPE(NAME=auditorAlgorithmType [, PROPERTIES([algorithmProperties])])
```

- STORAGE_UNITS needs to use storage units managed by RDL
- shardingAlgorithmType specifies the type of automatic sharding algorithm, please refer to Auto Sharding Algorithm
- keyGenerateStrategyType specifies the distributed primary key generation strategy, please refer to Key Generate Algorithm
- auditorAlgorithmType specifies the sharding audit strategy, please refer to Sharding Audit Algorithm；
- Duplicate tableName will not be created
- shardingAlgorithm can be reused by different Sharding Table Rule, so when executing DROP SHARDING TABLE RULE, the corresponding shardingAlgorithm will not be removed
- To remove shardingAlgorithm, please execute DROP SHARDING ALGORITHM

- strategyType specifies the sharding strategy, please refer toSharding Strategy
- Sharding Table Rule supports both Auto Table and Table at the same time. The two types are different in syntax. For the corresponding configuration file, please refer to Sharding
- When using the autoCreativeAlgorithm way to specify shardingStrategy, a new sharding algorithm will be created automatically. The algorithm naming rule is tableName_strategyType_shardingAlgorithmType, such as t_order_database_inline
- executing CREATE SHARDING TABLE RULE , a new sharding algorithm will be created automatically. The algorithm naming rule is tableName_scope_shardingAlgorithmType , such as t_order_database_inline
- executing CREATE DEFAULT SHARDING STRATEGY , a new sharding algorithm is also created automatically, The algorithm naming rule is default_scope_shardingAlgorithmType , such as default_database_inline

### Sharding Table Reference Rule

```
CREATE SHARDING TABLE REFERENCE RULE tableReferenceRuleDefinition [, tableReferenceRuleDefinition] ...

ALTER SHARDING TABLE REFERENCE RULE tableReferenceRuleDefinition [, tableReferenceRuleDefinition] ...

DROP SHARDING TABLE REFERENCE RULE tableReferenceRuleDefinition [, tableReferenceRuleDefinition] ...

tableReferenceRuleDefinition:
    (tableName [, tableName] ... )
```

- ALTER will overwrite the sharding table references in the database with the new configuration

### Broadcast Table Rule

```
CREATE BROADCAST TABLE RULE tableName [, tableName] ...

DROP BROADCAST TABLE RULE tableName [, tableName] ...
```

### Example

### Sharding Table Rule

Key Generator

```
DROP SHARDING KEY GENERATOR snowflake_key_generator;
```

Auditor

```
CREATE SHARDING AUDITOR sharding_key_required_auditor (
TYPE(NAME="DML_SHARDING_CONDITIONS")
);

ALTER SHARDING AUDITOR sharding_key_required_auditor (
TYPE(NAME="DML_SHARDING_CONDITIONS")
);

DROP SHARDING AUDITOR IF EXISTS sharding_key_required_auditor;
```

Auto Table

```
CREATE SHARDING TABLE RULE t_order (
STORAGE_UNITS(ds_0,ds_1),
SHARDING_COLUMN=order_id,TYPE(NAME="hash_mod",PROPERTIES("sharding-count"="4")),
KEY_GENERATE_STRATEGY(COLUMN=another_id,TYPE(NAME="snowflake")),
AUDIT_STRATEGY(AUDITORS=[auditor1,auditor2],ALLOW_HINT_DISABLE=true)
);

ALTER SHARDING TABLE RULE t_order (
STORAGE_UNITS(ds_0,ds_1,ds_2,ds_3),
SHARDING_COLUMN=order_id,TYPE(NAME="hash_mod",PROPERTIES("sharding-count"="16")),
KEY_GENERATE_STRATEGY(COLUMN=another_id,TYPE(NAME="snowflake")),
AUDIT_STRATEGY(AUDITORS=[auditor1,auditor2],ALLOW_HINT_DISABLE=true)
);

DROP SHARDING TABLE RULE t_order;

DROP SHARDING ALGORITHM t_order_hash_mod;
```

Table

```
CREATE SHARDING TABLE RULE t_order_item (
DATANODES("ds_${0..1}.t_order_item_${0..1}"),
DATABASE_STRATEGY(TYPE="standard",SHARDING_COLUMN=user_id,SHARDING_ALGORITHM(TYPE(NAME="inline",
PROPERTIES("algorithm-expression"="ds_${user_id % 2}")))),
TABLE_STRATEGY(TYPE="standard",SHARDING_COLUMN=order_id,SHARDING_ALGORITHM(TYPE(NAME="inline",PROPERTIES(
"algorithm-expression"="t_order_item_${order_id % 2}")))),
KEY_GENERATE_STRATEGY(COLUMN=another_id,TYPE(NAME="snowflake")),
AUDIT_STRATEGY(AUDITORS=[auditor1,auditor2],ALLOW_HINT_DISABLE=true)
);

ALTER SHARDING TABLE RULE t_order_item (
DATANODES("ds_${0..3}.t_order_item${0..3}"),
DATABASE_STRATEGY(TYPE="standard",SHARDING_COLUMN=user_id,SHARDING_ALGORITHM(TYPE(NAME="inline",
PROPERTIES("algorithm-expression"="ds_${user_id % 4}")))),
TABLE_STRATEGY(TYPE="standard",SHARDING_COLUMN=order_id,SHARDING_ALGORITHM(TYPE(NAME="inline",PROPERTIES(
"algorithm-expression"="t_order_item_${order_id % 4}")))),
KEY_GENERATE_STRATEGY(COLUMN=another_id,TYPE(NAME="snowflake")),
AUDIT_STRATEGY(AUDITORS=[auditor1,auditor2],ALLOW_HINT_DISABLE=true)
);

DROP SHARDING TABLE RULE t_order_item;

DROP SHARDING ALGORITHM database_inline;

CREATE DEFAULT SHARDING DATABASE STRATEGY (
TYPE="standard",SHARDING_COLUMN=order_id,SHARDING_ALGORITHM(TYPE(NAME="inline",PROPERTIES("algorithm-
expression"="ds_${order_id % 2}")))
);

ALTER DEFAULT SHARDING DATABASE STRATEGY (
TYPE="standard",SHARDING_COLUMN=another_id,SHARDING_ALGORITHM(TYPE(NAME="inline",PROPERTIES("algorithm-
expression"="ds_${another_id % 2}")))
);

DROP DEFAULT SHARDING DATABASE STRATEGY;
```

### Sharding Table Reference Rule

```
CREATE SHARDING TABLE REFERENCE RULE (t_order,t_order_item),(t_1,t_2);

ALTER SHARDING TABLE REFERENCE RULE (t_order,t_order_item);

DROP SHARDING TABLE REFERENCE RULE;

DROP SHARDING TABLE REFERENCE RULE (t_order,t_order_item);
```

### Broadcast Table Rule

```
CREATE BROADCAST TABLE RULE t_a,t_b;

DROP BROADCAST TABLE RULE t_a;
```

### Single Table

#### Definition

```
SET DEFAULT SINGLE TABLE STORAGE UNIT = (storageUnitName | RANDOM)
```

- storageUnitName needs to use storage unit managed by RDL. The RANDOM stands for random storage.

#### Example

```
SET DEFAULT SINGLE TABLE STORAGE UNIT = ds_0

SET DEFAULT SINGLE TABLE STORAGE UNIT = RANDOM
```

### Read/write Splitting

#### Syntax

```
CREATE READWRITE_SPLITTING RULE readwriteSplittingRuleDefinition [, readwriteSplittingRuleDefinition] ...

ALTER READWRITE_SPLITTING RULE readwriteSplittingRuleDefinition [, readwriteSplittingRuleDefinition] ...

DROP READWRITE_SPLITTING RULE ruleName [, ruleName] ...

readwriteSplittingRuleDefinition:
  ruleName ([staticReadwriteSplittingRuleDefinition | dynamicReadwriteSplittingRuleDefinition]
        [, loadBalancerDefinition])

staticReadwriteSplittingRuleDefinition:
  WRITE_STORAGE_UNIT=storageUnitName, READ_STORAGE_UNITS(storageUnitName [, storageUnitName] ... )

dynamicReadwriteSplittingRuleDefinition:
  AUTO_AWARE_RESOURCE=autoAwareResourceName [, WRITE_DATA_SOURCE_QUERY_
ENABLED=writeDataSourceQueryEnabled]

loadBalancerDefinition:
  TYPE(NAME=loadBalancerType [, PROPERTIES([algorithmProperties] )] )
```

```
algorithmProperties:
  algorithmProperty [, algorithmProperty] ...

algorithmProperty:
  key=value

writeDataSourceQueryEnabled:
  TRUE | FALSE
```

### Parameters Explained

| name | Date Type | Description |
|---|---|---|
| ruleName | IDENTIFIER | Rule name |
| storageUnitName | IDENTIFIER | Registered data source name |
| autoAwareResourceName | IDENTIFIER | Database discovery logic data source name |
| writeDataSourceQueryEnabled | BOOLEAN | All read data source are offline, write data source whether the data source is responsible for read traffic |
| loadBalancerType | STRING | Load balancing algorithm type |

### Notes

- Support the creation of static readwrite-splitting rules and dynamic readwrite-splitting rules
- Dynamic readwrite-splitting rules rely on database discovery rules
- loadBalancerType specifies the load balancing algorithm type, please refer to Load Balance Algorithm
- Duplicate ruleName will not be created

### Example

```
// Static
CREATE READWRITE_SPLITTING RULE ms_group_0 (
WRITE_STORAGE_UNIT=write_ds,
READ_STORAGE_UNITS(read_ds_0,read_ds_1),
TYPE(NAME="random")
);

// Dynamic
CREATE READWRITE_SPLITTING RULE ms_group_1 (
AUTO_AWARE_RESOURCE=group_0,
WRITE_DATA_SOURCE_QUERY_ENABLED=false,
TYPE(NAME="random"));


ALTER READWRITE_SPLITTING RULE ms_group_1 (
WRITE_STORAGE_UNIT=write_ds,
READ_STORAGE_UNITS(read_ds_0,read_ds_1,read_ds_2),
TYPE(NAME="random"));


DROP READWRITE_SPLITTING RULE ms_group_1;
```

### DB Discovery

#### Syntax

```
CREATE DB_DISCOVERY RULE ruleDefinition [, ruleDefinition] ...

ALTER DB_DISCOVERY RULE ruleDefinition [, ruleDefinition] ...

DROP DB_DISCOVERY RULE ruleName [, ruleName] ...

DROP DB_DISCOVERY TYPE discoveryTypeName [, discoveryTypeName] ...

DROP DB_DISCOVERY HEARTBEAT discoveryHeartbeatName [, discoveryHeartbeatName] ...

ruleDefinition:
  ruleName (storageUnits, typeDefinition, heartbeatDefinition)

storageUnits:
  STORAGE_UNITS(storageUnitName [, storageUnitName] ...)

typeDefinition:
  TYPE(NAME=typeName [, PROPERTIES([properties] )] )

heartbeatDefinition
  HEARTBEAT (PROPERTIES (properties))

properties:
  property [, property] ...

property:
  key=value
```

#### Parameters Explained

| name | DateType | Description |
|---|---|---|
| discoveryTypeName | IDENTIFIER | Database discovery type name |
| ruleName | IDENTIFIER | Rule name |
| discoveryHeartbeatName | IDENTIFIER | Detect heartbeat name |
| typeName | STRING | Database discovery type, such as: MySQL.MGR |
| storageUnitName | IDENTIFIER | Storage unit name |

#### Notes

- discoveryType specifies the database discovery service type, ShardingSphere has built-in support for MySQL.MGR
- Duplicate ruleName will not be created
- The discoveryType and discoveryHeartbeat being used cannot be deleted
- Names with - need to use " " when changing
- When removing the discoveryRule, the discoveryType and discoveryHeartbeat used by the discoveryRule will not be removed

### Example

#### When creating a discoveryRule, create both discoveryType and discoveryHeartbeat

```
CREATE DB_DISCOVERY RULE db_discovery_group_0 (
STORAGE_UNITS(ds_0, ds_1, ds_2),
TYPE(NAME='MySQL.MGR',PROPERTIES('group-name'='92504d5b-6dec')),
HEARTBEAT(PROPERTIES('keep-alive-cron'='0/5 * * * * ?'))
);

ALTER DB_DISCOVERY RULE db_discovery_group_0 (
STORAGE_UNITS(ds_0, ds_1, ds_2),
TYPE(NAME='MySQL.MGR',PROPERTIES('group-name'='246e9612-aaf1')),
HEARTBEAT(PROPERTIES('keep-alive-cron'='0/5 * * * * ?'))
);

DROP DB_DISCOVERY RULE db_discovery_group_0;

DROP DB_DISCOVERY TYPE db_discovery_group_0_mgr;

DROP DB_DISCOVERY HEARTBEAT db_discovery_group_0_heartbeat;
```

## Encrypt

### Syntax

```
CREATE ENCRYPT RULE encryptRuleDefinition [, encryptRuleDefinition] ...

ALTER ENCRYPT RULE encryptRuleDefinition [, encryptRuleDefinition] ...

DROP ENCRYPT RULE tableName [, tableName] ...

encryptRuleDefinition:
  tableName(COLUMNS(columnDefinition [, columnDefinition] ...), QUERY_WITH_CIPHER_COLUMN=queryWithCipherColumn)

columnDefinition:
  (NAME=columnName [, PLAIN=plainColumnName] , CIPHER=cipherColumnName, encryptAlgorithm)

encryptAlgorithm:
  TYPE(NAME=encryptAlgorithmType [, PROPERTIES([algorithmProperties] )] )

algorithmProperties:
  algorithmProperty [, algorithmProperty] ...

algorithmProperty:
  key=value
```

## Parameters Explained

| name | DateType | Description |
|---|---|---|
| tableName | IDENTIFIER | Table name |
| columnName | IDENTIFIER | Logic column name |
| plainColumnName | IDENTIFIER | Plain column name |
| cipherColumnName | IDENTIFIER | Cipher column name |
| encryptAlgorithmType | STRING | Encryption algorithm type name |

## Notes

- PLAIN specifies the plain column, CIPHER specifies the cipher column
- encryptAlgorithmType specifies the encryption algorithm type, please refer to Encryption Algorithm
- Duplicate tableName will not be created
- queryWithCipherColumn support uppercase or lowercase true or false

## KeyManager Syntax

```
createEncryptKeyManager
  : CREATE ENCRYPT KEY MANAGER keyManagerName keyManagerDefinition
  ;

alterEncryptKeyManager
  : ALTER ENCRYPT KEY MANAGER keyManagerName keyManagerDefinition
  ;

dropEncryptKeyManager
  : DROP ENCRYPT KEY MANAGER ifExists? keyManagerName
  ;

keyManagerDefinition:
  (TYPE(NAME=keyManagerName, PROPERTIES([keyManagerProperties]))
);)

keyManagerProperties:
  keyManagerProperty [, keyManagerProperty] ...

keyManagerProperty:
  key=value
```

## Parameters Explained

| name | DateType | Description |
|---|---|---|
| keyManagerName | STRING | Name of key manager |

## Example

```
CREATE ENCRYPT RULE t_encrypt (
COLUMNS(
(NAME=user_id,PLAIN=user_plain,CIPHER=user_cipher,TYPE(NAME='AES',PROPERTIES('aes-key-value'='123456abc'))),
(NAME=order_id, CIPHER =order_cipher,TYPE(NAME='MD5'))
), QUERY_WITH_CIPHER_COLUMN=true),
t_encrypt_2 (
COLUMNS(
(NAME=user_id,PLAIN=user_plain,CIPHER=user_cipher,TYPE(NAME='AES',PROPERTIES('aes-key-value'='123456abc'))),
(NAME=order_id, CIPHER=order_cipher,TYPE(NAME='MD5'))
), QUERY_WITH_CIPHER_COLUMN=FALSE);

ALTER ENCRYPT RULE t_encrypt (
COLUMNS(
(NAME=user_id,PLAIN=user_plain,CIPHER=user_cipher,TYPE(NAME='AES',PROPERTIES('aes-key-value'='123456abc'))),
(NAME=order_id,CIPHER=order_cipher,TYPE(NAME='MD5'))
), QUERY_WITH_CIPHER_COLUMN=TRUE);

DROP ENCRYPT RULE t_encrypt,t_encrypt_2;
```

## Shadow

### Syntax

```
CREATE SHADOW RULE shadowRuleDefinition [, shadowRuleDefinition] ...

ALTER SHADOW RULE shadowRuleDefinition [, shadowRuleDefinition] ...

DROP SHADOW RULE ruleName [, ruleName] ...

DROP SHADOW ALGORITHM algorithmName [, algorithmName] ...

CREATE DEFAULT SHADOW ALGORITHM shadowAlgorithm

ALTER DEFAULT SHADOW ALGORITHM shadowAlgorithm

DROP DEFAULT SHADOW ALGORITHM [IF EXISTS]

SHOW DEFAULT SHADOW ALGORITHM

SHOW SHADOW ALGORITHMS

shadowRuleDefinition: ruleName(storageUnitMapping, shadowTableRule [, shadowTableRule] ...)

storageUnitMapping: SOURCE=storageUnitName, SHADOW=storageUnitName

shadowTableRule: tableName(shadowAlgorithm [, shadowAlgorithm] ...)

shadowAlgorithm: TYPE(NAME=shadowAlgorithmType, PROPERTIES([algorithmProperties] ...))

algorithmProperties: algorithmProperty [, algorithmProperty] ...

algorithmProperty: key=value
```

### Parameters Explained

| name | DateType | Description |
|---|---|---|
| ruleName | IDENTIFIER | Rule name |
| storageUnitName | IDENTIFIER | Storage unit name |
| tableName | IDENTIFIER | Shadow table name |
| algorithmName | IDENTIFIER | Shadow algorithm name |
| shadowAlgorithmType | STRING | Shadow algorithm type |

### Notes

- Duplicate ruleName cannot be created
- storageUnitMapping specifies the mapping relationship between the source database and the shadow library. You need to use the storage unit managed by RDL, please refer to storage unit
- shadowAlgorithm can act on multiple shadowTableRule at the same time
- shadowAlgorithmType currently supports VALUE_MATCH, REGEX_MATCH and SIMPLE_HINT
- shadowTableRule can be reused by different shadowRuleDefinition, so when executing DROP SHADOW RULE, the corresponding shadowTableRule will not be removed
- shadowAlgorithm can be reused by different shadowTableRule, so when executing ALTER SHADOW RULE, the corresponding shadowAlgorithm will not be removed
- algorithmName will be automatically generated according to ruleName, tableName, shadowAlgorithmType and algorithm collection index. The default name is default_shadow_algorithm.

### Example

```
CREATE SHADOW RULE shadow_rule(
SOURCE=demo_ds,
SHADOW=demo_ds_shadow,
t_order(TYPE(NAME="SIMPLE_HINT", PROPERTIES("shadow"="true", "foo"="bar")),TYPE(NAME="REGEX_MATCH", PROPERTIES(
"operation"="insert","column"="user_id", "regex"='[1]'))),
t_order_item(TYPE(NAME="VALUE_MATCH", PROPERTIES("operation"="insert","column"="user_id", "value"='1'))));

ALTER SHADOW RULE shadow_rule(
SOURCE=demo_ds,
SHADOW=demo_ds_shadow,
t_order(TYPE(NAME="SIMPLE_HINT", PROPERTIES("shadow"="true", "foo"="bar")),TYPE(NAME="REGEX_MATCH", PROPERTIES(
"operation"="insert","column"="user_id", "regex"='[1]'))),
t_order_item(TYPE(NAME="VALUE_MATCH", PROPERTIES("operation"="insert","column"="user_id", "value"='1'))));

DROP SHADOW RULE shadow_rule;

DROP SHADOW ALGORITHM simple_hint_algorithm;

CREATE DEFAULT SHADOW ALGORITHM NAME = simple_hint_algorithm;

ALTER DEFAULT SHADOW ALGORITHM TYPE(NAME="SIMPLE_HINT", PROPERTIES("shadow"="false", "foo"="bar");

SHOW DEFAULT SHADOW ALGORITHM;

DROP DEFAULT SHADOW ALGORITHM;
```

### RQL Syntax

RQL (Resource & Rule Query Language) responsible for resources/rules query.

### Storage Unit Query

### Syntax

```
SHOW STORAGE UNITS [ [FROM databaseName] | [WHERE USAGE_COUNT = usageCount] ]
```

### Return Value Description

| Column | Description |
|---|---|
| name | Data source name |
| type | Data source type |
| host | Data source host |
| port | Data source port |
| db | Database name |
| attribute | Data source attribute |

### Example

```
mysql> SHOW STORAGE UNITS;
+------+-------+-----------+------+------+--------------------------------+--------------------------+--------------------------+---------------+---------------+--
---------+------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------+
| name | type  | host      | port | db   | connection_timeout_milliseconds | idle_timeout_milliseconds | max_lifetime_milliseconds |
max_pool_size | min_pool_size | read_only | other_attributes

                                                                                                                                    |
+------+-------+-----------+------+------+--------------------------------+--------------------------+--------------------------+---------------+---------------+--
---------+------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------+
| ds_0 | MySQL | 127.0.0.1 | 3306 | db_0 | 30000                          | 60000                    | 1800000                  | 50            | 1             | false     | {
"dataSourceProperties":{"cacheServerConfiguration":"true","elideSetAutoCommits":"true","useServerPrepStmts":"true",
"cachePrepStmts":"true","rewriteBatchedStatements":"true","cacheResultSetMetadata":"false","useLocalSessionState":"true",
"maintainTimeStats":"false","prepStmtCacheSize":"8192","tinyInt1isBit":"false","prepStmtCacheSqlLimit":"2048",
"netTimeoutForStreamingResults":"0","zeroDateTimeBehavior":"round"},"healthCheckProperties":{},"initializationFailTimeout
":1,"validationTimeout":5000,"leakDetectionThreshold":0,"poolName":"HikariPool-1","registerMbeans":false,
"allowPoolSuspension":false,"autoCommit":true,"isolateInternalQueries":false} |
| ds_1 | MySQL | 127.0.0.1 | 3306 | db_1 | 30000                          | 60000                    | 1800000                  | 50            | 1             | false     | {
"dataSourceProperties":{"cacheServerConfiguration":"true","elideSetAutoCommits":"true","useServerPrepStmts":"true",
"cachePrepStmts":"true","rewriteBatchedStatements":"true","cacheResultSetMetadata":"false","useLocalSessionState":"true",
"maintainTimeStats":"false","prepStmtCacheSize":"8192","tinyInt1isBit":"false","prepStmtCacheSqlLimit":"2048",
"netTimeoutForStreamingResults":"0","zeroDateTimeBehavior":"round"},"healthCheckProperties":{},"initializationFailTimeout
":1,"validationTimeout":5000,"leakDetectionThreshold":0,"poolName":"HikariPool-2","registerMbeans":false,
"allowPoolSuspension":false,"autoCommit":true,"isolateInternalQueries":false} |
+------+-------+-----------+------+------+--------------------------------+--------------------------+--------------------------+---------------+---------------+--
---------+------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------+
mysql> SHOW STORAGE UNITS FROM sharding_db;
+------+-------+-----------+------+------+-------------------------------+--------------------------+--------------------------+---------------+---------------+--
---------+-----------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------+
| name | type  | host      | port | db   | connection_timeout_milliseconds | idle_timeout_milliseconds | max_lifetime_milliseconds |
max_pool_size | min_pool_size | read_only | other_attributes

                                                                                                                                  |
+------+-------+-----------+------+------+-------------------------------+--------------------------+--------------------------+---------------+---------------+--
---------+-----------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------+
| ds_0 | MySQL | 127.0.0.1 | 3306 | db_0 | 30000                         | 60000                    | 1800000                  | 50            | 1             | false     | {
"dataSourceProperties":{"cacheServerConfiguration":"true","elideSetAutoCommits":"true","useServerPrepStmts":"true",
"cachePrepStmts":"true","rewriteBatchedStatements":"true","cacheResultSetMetadata":"false","useLocalSessionState":"true",
"maintainTimeStats":"false","prepStmtCacheSize":"8192","tinyInt1isBit":"false","prepStmtCacheSqlLimit":"2048",
"netTimeoutForStreamingResults":"0","zeroDateTimeBehavior":"round"},"healthCheckProperties":{},"initializationFailTimeout
":1,"validationTimeout":5000,"leakDetectionThreshold":0,"poolName":"HikariPool-1","registerMbeans":false,
"allowPoolSuspension":false,"autoCommit":true,"isolateInternalQueries":false} |
| ds_1 | MySQL | 127.0.0.1 | 3306 | db_1 | 30000                         | 60000                    | 1800000                  | 50            | 1             | false     | {
"dataSourceProperties":{"cacheServerConfiguration":"true","elideSetAutoCommits":"true","useServerPrepStmts":"true",
"cachePrepStmts":"true","rewriteBatchedStatements":"true","cacheResultSetMetadata":"false","useLocalSessionState":"true",
"maintainTimeStats":"false","prepStmtCacheSize":"8192","tinyInt1isBit":"false","prepStmtCacheSqlLimit":"2048",
"netTimeoutForStreamingResults":"0","zeroDateTimeBehavior":"round"},"healthCheckProperties":{},"initializationFailTimeout
":1,"validationTimeout":5000,"leakDetectionThreshold":0,"poolName":"HikariPool-2","registerMbeans":false,
"allowPoolSuspension":false,"autoCommit":true,"isolateInternalQueries":false} |
+------+-------+-----------+------+------+-------------------------------+--------------------------+--------------------------+---------------+---------------+--
---------+-----------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------+
2 rows in set (0.84 sec)
```

### Rule Query

This chapter describes the syntax of rule query.

### Sharding

#### Syntax

#### Sharding Table Rule

```
SHOW SHARDING TABLE tableRule | RULES [FROM databaseName]

SHOW SHARDING ALGORITHMS [FROM databaseName]

SHOW UNUSED SHARDING ALGORITHMS [FROM databaseName]

SHOW SHARDING AUDITORS [FROM databaseName]

SHOW SHARDING TABLE RULES USED ALGORITHM shardingAlgorithmName [FROM databaseName]

SHOW SHARDING KEY GENERATORS [FROM databaseName]
SHOW UNUSED SHARDING KEY GENERATORS [FROM databaseName]

SHOW UNUSED SHARDING AUDITORS [FROM databaseName]

SHOW SHARDING TABLE RULES USED KEY GENERATOR keyGeneratorName [FROM databaseName]

SHOW SHARDING TABLE RULES USED AUDITOR auditorName [FROM databaseName]

SHOW DEFAULT SHARDING STRATEGY

SHOW SHARDING TABLE NODES

tableRule:
  RULE tableName
```

- Support query all data fragmentation rules and specified table query
- Support query all sharding algorithms
- Support query all sharding audit algorithms

### Sharding Table Reference Rule

```
SHOW SHARDING TABLE REFERENCE RULES [FROM databaseName]
```

### Broadcast Table Rule

```
SHOW BROADCAST TABLE RULES [FROM databaseName]
```

### Sharding Table Rule

| Column | Description |
|---|---|
| table | Logical table name |
| actual_data_nodes | Actual data node |
| actual_data_sources | Actual data source (Displayed when creating rules by RDL) |
| database_strategy_type | Database sharding strategy type |
| da tabase_sharding_column | Database sharding column |
| database_s harding_algorithm_type | Database sharding algorithm type |
| database_sh arding_algorithm_props | Database sharding algorithm properties |
| table_strategy_type | Table sharding strategy type |
| table_sharding_column | Table sharding column |
| table_s harding_algorithm_type | Table sharding algorithm type |
| table_sh arding_algorithm_props | Table sharding algorithm properties |
| key_generate_column | Sharding key generator column |
| key_generator_type | Sharding key generator type |
| key_generator_props | Sharding key generator properties |
| auditor_types | Sharding auditor types |
| allow_hint_disable | Enable or disable sharding audit hint |

### Sharding Algorithms

| Column | Description |
|---|---|
| name | Sharding algorithm name |
| type | Sharding algorithm type |
| props | Sharding algorithm properties |

### Unused Sharding Algorithms

| Column | Description |
|---|---|
| name | Sharding algorithm name |
| type | Sharding algorithm type |
| props | Sharding algorithm properties |

### Sharding auditors

| Column | Description |
|---|---|
| name | Sharding audit algorithm name |
| type | Sharding audit algorithm type |
| props | Sharding audit algorithm properties |

### Unused Sharding Auditors

| Column | Description |
|---|---|
| name | Sharding audit algorithm name |
| type | Sharding audit algorithm type |
| props | Sharding audit algorithm properties |

### Sharding key generators

| Column | Description |
|---|---|
| name | Sharding key generator name |
| type | Sharding key generator type |
| props | Sharding key generator properties |

### Unused Sharding Key Generators

| Column | Description |
|---|---|
| name | Sharding key generator name |
| type | Sharding key generator type |
| props | Sharding key generator properties |

### Default Sharding Strategy

| Column | Description |
|---|---|
| name | Strategy name |
| type | Sharding strategy type |
| sharding_column | Sharding column |
| sharding_algorithm_name | Sharding algorithm name |
| sharding_algorithm_type | Sharding algorithm type |
| sharding_algorithm_props | Sharding algorithm properties |

### Sharding Table Nodes

| Column | Description |
|---|---|
| name | Sharding rule name |
| nodes | Sharding nodes |

### Sharding Table Reference Rule

| Column | Description |
|---|---|
| sharding_table_reference | Sharding table reference |

### Broadcast Table Rule

| Column | Description |
|---|---|
| broadcast_table | Broadcast table |

### Sharding Table Rule

SHOW SHARDING TABLE RULES

```
mysql> SHOW SHARDING TABLE RULES;
+--------------+--------------------------------+------------------+--------------------+----------------------+----------------------------+------------------
---------------------+------------------+-------------------+-------------------------+---------------------------------------------+------------------+---
---------------+------------------+
| table       | actual_data_nodes            | actual_data_sources | database_strategy_type | database_sharding_column | database_
sharding_algorithm_type | database_sharding_algorithm_props       | table_strategy_type | table_sharding_column | table_
sharding_algorithm_type | table_sharding_algorithm_props                | key_generate_column | key_generator_type | key_
generator_props |auditor_types | allow_hint_disable |
+--------------+--------------------------------+------------------+--------------------+----------------------+----------------------------+------------------
---------------------+------------------+-------------------+-------------------------+---------------------------------------------+------------------+---
---------------+------------------+
| t_order     | ds_${0..1}.t_order_${0..1}     |              | INLINE          | user_id           | INLINE                 | algorithm-expression:ds_$
{user_id % 2} | INLINE         | order_id        | INLINE                 | algorithm-expression:t_order_${order_id % 2}     | order_id        |
SNOWFLAKE      |              |DML_SHARDING_CONDITIONS    |true     |
| t_order_item | ds_${0..1}.t_order_item_${0..1} |              | INLINE          | user_id           | INLINE                 | algorithm-
expression:ds_${user_id % 2} | INLINE         | order_id        | INLINE                 | algorithm-expression:t_order_item_${order_id %
2} | order_item_id    | SNOWFLAKE      |              |                 |       |
| t2        |                          | ds_0,ds_1      |              |               |                   |                         | mod           | id              |
mod                | sharding-count:10                          |             |            |             |                    |       |
+--------------+---------------------------------+-------------------+----------------------+------------------------+------------------------------+-------------------
----------------------+-------------------+--------------------+---------------------------+---------------------------------------------------+-------------------+---
---------------+-------------------+
3 rows in set (0.02 sec)
```

SHOW SHARDING TABLE RULE tableName

```
mysql> SHOW SHARDING TABLE RULE t_order;
+---------+---------------------------+-------------------+----------------------+-----------------------+------------------------------+----------------------------
------------+-------------------+--------------------+---------------------------+--------------------------------------------+-------------------+------------------
+-------------------+
| table   | actual_data_nodes        | actual_data_sources | database_strategy_type | database_sharding_column | database_
sharding_algorithm_type | database_sharding_algorithm_props        | table_strategy_type | table_sharding_column | table_
sharding_algorithm_type | table_sharding_algorithm_props               | key_generate_column | key_generator_type | key_
generator_props | auditor_types | allow_hint_disable |
+---------+---------------------------+-------------------+----------------------+-----------------------+------------------------------+----------------------------
------------+-------------------+--------------------+---------------------------+--------------------------------------------+-------------------+------------------
+-------------------+
| t_order | ds_${0..1}.t_order_${0..1} |                 | INLINE            | user_id              | INLINE                  | algorithm-expression:ds_$
{user_id % 2} | INLINE          | order_id         | INLINE                  | algorithm-expression:t_order_${order_id % 2} | order_id         |
SNOWFLAKE       |                 | DML_SHARDING_CONDITIONS   |true     |
+---------+---------------------------+-------------------+----------------------+-----------------------+------------------------------+----------------------------
------------+-------------------+--------------------+---------------------------+--------------------------------------------+-------------------+------------------
+-------------------+
1 row in set (0.01 sec)
```

SHOW SHARDING ALGORITHMS

```
mysql> SHOW SHARDING ALGORITHMS;
+-------------------------+--------+-----------------------------------------------------+
| name                  | type   | props                                          |
+-------------------------+--------+-----------------------------------------------------+
| t_order_inline        | INLINE | algorithm-expression=t_order_${order_id % 2}       |
| t_order_item_inline    | INLINE | algorithm-expression=t_order_item_${order_id % 2}  |
+-------------------------+--------+-----------------------------------------------------+
2 row in set (0.01 sec)
```

SHOW UNUSED SHARDING ALGORITHMS

```
mysql> SHOW UNUSED SHARDING ALGORITHMS;
+---------------+--------+-----------------------------------------------------+
| name        | type   | props                                          |
+---------------+--------+-----------------------------------------------------+
| t1_inline     | INLINE | algorithm-expression=t_order_${order_id % 2}       |
+---------------+--------+-----------------------------------------------------+
1 row in set (0.01 sec)
```

SHOW SHARDING AUDITORS

```
mysql> SHOW SHARDING AUDITORS;
+------------+-------------------------+-------+
| name      | type                  | props |
+------------+-------------------------+-------+
| dml_audit  | DML_SHARDING_CONDITIONS |       |
+------------+-------------------------+-------+
2 row in set (0.01 sec)
```

SHOW SHARDING TABLE RULES USED ALGORITHM shardingAlgorithmName

```
mysql> SHOW SHARDING TABLE RULES USED ALGORITHM t_order_inline;
+-------+---------+
| type  | name    |
+-------+---------+
| table | t_order |
+-------+---------+
1 row in set (0.01 sec)
```

SHOW SHARDING KEY GENERATORS

```
mysql> SHOW SHARDING KEY GENERATORS;
+-------------------------+-----------+-----------------+
| name                    | type      | props           |
+-------------------------+-----------+-----------------+
| t_order_snowflake       | snowflake |                 |
| t_order_item_snowflake  | snowflake |                 |
| uuid_key_generator      | uuid      |                 |
+-------------------------+-----------+-----------------+
3 row in set (0.01 sec)
```

SHOW UNUSED SHARDING KEY GENERATORS

```
mysql> SHOW UNUSED SHARDING KEY GENERATORS;
+-------------------------+-----------+-----------------+
| name                    | type      | props           |
+-------------------------+-----------+-----------------+
| uuid_key_generator      | uuid      |                 |
+-------------------------+-----------+-----------------+
1 row in set (0.01 sec)
```

SHOW UNUSED SHARDING KEY AUDITORS

```
mysql> SHOW UNUSED SHARDING KEY AUDITORS;
+------------+-------------------------+-------+
| name       | type                    | props |
+------------+-------------------------+-------+
| dml_audit  | DML_SHARDING_CONDITIONS |       |
+------------+-------------------------+-------+
1 row in set (0.01 sec)
```

SHOW SHARDING TABLE RULES USED KEY GENERATOR keyGeneratorName

```
mysql> SHOW SHARDING TABLE RULES USED KEY GENERATOR keyGeneratorName;
+-------+---------+
| type  | name    |
+-------+---------+
| table | t_order |
+-------+---------+
1 row in set (0.01 sec)
```

SHOW SHARDING TABLE RULES USED AUDITOR auditorName

```
mysql> SHOW SHARDING TABLE RULES USED AUDITOR sharding_key_required;
+-------+---------+
| type  | name    |
+-------+---------+
| table | t_order |
+-------+---------+
1 row in set (0.01 sec)
```

SHOW DEFAULT SHARDING STRATEGY

```
mysql> SHOW DEFAULT SHARDING STRATEGY ;

+----------+---------+-------------------+------------------------+------------------------+-----------------------------------------+
| name     | type    | sharding_column   | sharding_algorithm_name | sharding_algorithm_type | sharding_algorithm_props                |
+----------+---------+-------------------+------------------------+------------------------+-----------------------------------------+
| TABLE    | NONE    |                   |                        |                        |                                         |
| DATABASE | STANDARD| order_id          | database_inline        | INLINE                 | {algorithm-expression=ds_${user_id % 2}} |
+----------+---------+--------------------+-------------------------+-------------------------+------------------------------------------+
2 rows in set (0.07 sec)
```

SHOW SHARDING TABLE NODES

```
mysql> SHOW SHARDING TABLE NODES;
+---------+----------------------------------------------------------------+
| name    | nodes                                                          |
+---------+----------------------------------------------------------------+
| t_order | ds_0.t_order_0, ds_1.t_order_1, ds_0.t_order_2, ds_1.t_order_3 |
+---------+----------------------------------------------------------------+
1 row in set (0.02 sec)
```

### Sharding Table Reference Rule

```
mysql> SHOW SHARDING TABLE REFERENCE RULES;
+--------------------------+
| sharding_table_reference |
+--------------------------+
| t_order,t_order_item     |
| t1,t2                    |
+--------------------------+
2 rows in set (0.00 sec)
```

### Broadcast Table Rule

```
mysql> SHOW BROADCAST TABLE RULES;
+-----------------------+
| broadcast_table       |
+-----------------------+
| t_1                   |
| t_2                   |
+-----------------------+
2 rows in set (0.00 sec)
```

### Single Table

### Syntax

```
SHOW DEFAULT SINGLE TABLE STORAGE UNIT [FROM databaseName]

SHOW SINGLE (TABLES | table) [FROM databaseName]

COUNT SINGLE_TABLE RULE [FROM databaseName]

table:
  TABLE tableName
```

### Return Value Description

#### Single Table Storage Unit

| Column | Description |
|---|---|
| storage_unit_name | Storage unit name |

#### Single Table

| Column | Description |
|---|---|
| table_name | Single table name |
| resource_name | The resource name where the single table is located |

#### Single Table Rule Count

| Column | Description |
|---|---|
| rule_name | Single table rule name |
| database | The database name where the single table is located |
| count | The count of single table rules |

### Example

SHOW DEFAULT SINGLE TABLE STORAGE UNIT

```
sql> SHOW DEFAULT SINGLE TABLE STORAGE UNIT;
+-------------------+
| storage_unit_name |
+-------------------+
| ds_0              |
+-------------------+
1 row in set (0.01 sec)
```

SHOW SINGLE TABLE tableName

```
sql> SHOW SINGLE TABLE t_single_0;
+---------------+---------------+
| table_name    | resource_name |
+---------------+---------------+
| t_single_0    | ds_0          |
+---------------+---------------+
1 row in set (0.01 sec)
```

SHOW SINGLE TABLES

```
mysql> SHOW SINGLE TABLES;
+--------------+---------------+
| table_name   | resource_name |
+--------------+---------------+
| t_single_0   | ds_0          |
| t_single_1   | ds_1          |
+--------------+---------------+
2 rows in set (0.02 sec)
```

COUNT SINGLE_TABLE RULE

```
mysql> COUNT SINGLE_TABLE RULE;
+--------------+----------+-------+
| rule_name    | database | count |
+--------------+----------+-------+
| t_single_0   | ds       | 2     |
+--------------+----------+-------+
1 row in set (0.02 sec)
```

### Read/write Splitting

### Syntax

```
SHOW READWRITE_SPLITTING RULES [FROM databaseName]
```

### Return Value Description

| Column | Description |
|---|---|
| name | Rule name |
| auto_aware_data_source_name | Auto-Aware discovery data source name (Display configuration dynamic readwrite splitting rules) |
| write_data_source_query_enabled | All read data source are offline, write data source whether the data source is responsible for read traffic |
| write_data_source_name | Write data source name |
| read_data_source_names | Read data source name list |
| load_balancer_type | Load balance algorithm type |
| load_balancer_props | Load balance algorithm parameter |

### Example

Static Readwrite Splitting Rules

```
mysql> SHOW READWRITE_SPLITTING RULES;
+------------+-----------------------------+------------------------+------------------------+--------------------+---------------------+
| name       | auto_aware_data_source_name | write_data_source_name | read_data_source_names | load_balancer_type | load_
balancer_props |
+------------+-----------------------------+------------------------+------------------------+--------------------+---------------------+
| ms_group_0 |                             | ds_primary             | ds_slave_0, ds_slave_1 | random             |                     |
+------------+-----------------------------+------------------------+------------------------+--------------------+---------------------+
1 row in set (0.00 sec)
```

Dynamic Readwrite Splitting Rules

```
mysql> SHOW READWRITE_SPLITTING RULES FROM readwrite_splitting_db;
+-------------+-----------------------------+------------------------+------------------------+--------------------+---------------------+
| name        | auto_aware_data_source_name | write_data_source_query_enabled | write_data_source_name | read_data_source_
names | load_balancer_type | load_balancer_props |
+-------------+-----------------------------+------------------------+------------------------+--------------------+---------------------+
| readwrite_ds | ms_group_0                 |                        |                        |                    | random             | read_weight:2:1     |
+-------+-----------------------------+------------------------+------------------------+--------------------+---------------------+
1 row in set (0.01 sec)
```

Static Readwrite Splitting Rules And Dynamic Readwrite Splitting Rules

```
mysql> SHOW READWRITE_SPLITTING RULES FROM readwrite_splitting_db;
+-------------+---------------------------+-----------------------+-----------------------+-------------------+--------------------+
| name        | auto_aware_data_source_name | write_data_source_query_enabled | write_data_source_name | read_data_source_
names | load_balancer_type | load_balancer_props |
+-------------+---------------------------+-----------------------+-----------------------+-------------------+--------------------+
| readwrite_ds | ms_group_0                |                       | write_ds              | read_ds_0, read_ds_1 | random            | read_weight=2:1
 |
+-------+---------------------------+-----------------------+-----------------------+-------------------+--------------------+
1 row in set (0.00 sec)
```

## DB Discovery

### Syntax

```
SHOW DB_DISCOVERY RULES [FROM databaseName]

SHOW DB_DISCOVERY TYPES [FROM databaseName]

SHOW DB_DISCOVERY HEARTBEATS [FROM databaseName]
```

### Return Value Description

### DB Discovery Rule

| Column | Description |
|---|---|
| group_name | Rule name |
| data_source_names | Data source name list |
| primary_data_source_name | Primary data source name |
| discovery_type | Database discovery service type |
| discovery_heartbeat | Database discovery service heartbeat |

### DB Discovery Type

| Column | Description |
|---|---|
| name | Type name |
| type | Type category |
| props | Type properties |

### DB Discovery Heartbeat

| Column | Description |
|---|---|
| name | Heartbeat name |
| props | Heartbeat properties |

## Example

DB Discovery Rule

```
mysql> SHOW DB_DISCOVERY RULES;
+----------------------+-------------------+-------------------------+---------------------------------------------------------------------------------+---------------
-------------------------------------------------------------+
| group_name         | data_source_names | primary_data_source_name | discovery_type                                                     |
discovery_heartbeat                                       |
+----------------------+-------------------+-------------------------+---------------------------------------------------------------------------------+---------------
-------------------------------------------------------------+
| db_discovery_group_0 | ds_0,ds_1,ds_2   |        ds_0            | {name=db_discovery_group_0_mgr, type=MySQL.MGR, props=
{group-name=92504d5b-6dec}} | {name=db_discovery_group_0_heartbeat, props={keep-alive-cron=0/5 * * * * ?}} |
+----------------------+-------------------+-------------------------+---------------------------------------------------------------------------------+---------------
-------------------------------------------------------------+
1 row in set (0.20 sec)
```

DB Discovery Type

```
mysql> SHOW DB_DISCOVERY TYPES;
+---------------------------+------------+------------------------------+
| name                   | type       | props                     |
+---------------------------+------------+------------------------------+
| db_discovery_group_0_mgr  | MySQL.MGR  | {group-name=92504d5b-6dec}   |
+---------------------------+------------+------------------------------+
1 row in set (0.01 sec)
```

DB Discovery Heartbeat

```
mysql> SHOW DB_DISCOVERY HEARTBEATS;
+--------------------------------+--------------------------------+
| name                        | props                     |
+--------------------------------+--------------------------------+
| db_discovery_group_0_heartbeat | {keep-alive-cron=0/5 * * * * ?} |
+--------------------------------+--------------------------------+
1 row in set (0.01 sec)
```

## Encrypt

## Syntax

```
SHOW ENCRYPT RULES [FROM databaseName]

SHOW ENCRYPT TABLE RULE tableName [FROM databaseName]
```

- Support to query all data encryption rules and specify logical table name query

## Return Value Description

| Column | Description |
|---|---|
| table | Logical table name |
| logic_column | Logical column name |
| logic_data_type | Logical column data type |
| cipher_column | Ciphertext column name |
| cipher_data_type | Ciphertext column data type |
| plain_column | Plaintext column name |
| plain_data_type | Plaintext column data type |
| assisted_query_column | Assisted query column name |
| assisted_query_data_type | Assisted query column data type |
| encryptor_type | Encryption algorithm type |
| encryptor_props | Encryption algorithm parameter |
| query_with_cipher_column | Whether to use encrypted column for query |

## Example

Show Encrypt Rules

```
mysql> SHOW ENCRYPT RULES FROM encrypt_db;
+-------------+-------------+-----------------+---------------+------------------+--------------+-----------------+-----------------------+---------------------------
+---------------+-----------------------+-------------------------+
| table       | logic_column | logic_data_type | cipher_column | cipher_data_type | plain_column | plain_data_type | assisted_
query_column | assisted_query_data_type | encryptor_type | encryptor_props        | query_with_cipher_column |
+-------------+-------------+-----------------+---------------+------------------+--------------+-----------------+-----------------------+---------------------------
+---------------+-----------------------+-------------------------+
| t_encrypt   | user_id     |             | user_cipher   |             | user_plain   |            |               |                   | AES         | aes-key-
value=123456abc | true                       |
| t_encrypt   | order_id    |             | order_cipher  |             |          |            |               |                   | MD5         |                  | true
          |
| t_encrypt_2 | user_id     |             | user_cipher   |             | user_plain   |            |               |                   | AES         | aes-key-
value=123456abc | false                      |
| t_encrypt_2 | order_id    |             | order_cipher  |             |          |            |               |                   | MD5         |                  |
false                |
+-------------+-------------+-----------------+---------------+------------------+--------------+-----------------+-----------------------+---------------------------
+---------------+-----------------------+-------------------------+
4 rows in set (0.78 sec)
```

Show Encrypt Table Rule Table Name

```
mysql> SHOW ENCRYPT TABLE RULE t_encrypt;
+-------------+-------------+-----------------+---------------+------------------+--------------+-----------------+-----------------------+---------------------------
+---------------+-----------------------+-------------------------+
| table       | logic_column | logic_data_type | cipher_column | cipher_data_type | plain_column | plain_data_type | assisted_
query_column | assisted_query_data_type | encryptor_type | encryptor_props        | query_with_cipher_column |
+-------------+-------------+-----------------+---------------+------------------+--------------+-----------------+-----------------------+---------------------------
+---------------+-----------------------+-------------------------+
| t_encrypt   | user_id     |             | user_cipher   |             | user_plain   |            |               |                   | AES         | aes-key-
value=123456abc | true                       |
| t_encrypt   | order_id    |             | order_cipher  |             |          |            |               |                   | MD5         |                  | true
          |
+-------------+-------------+-----------------+---------------+------------------+--------------+-----------------+-----------------------+---------------------------
+---------------+-----------------------+-------------------------+
2 rows in set (0.01 sec)

mysql> SHOW ENCRYPT TABLE RULE t_encrypt FROM encrypt_db;
+-------------+-------------+-----------------+---------------+------------------+--------------+-----------------+-----------------------+---------------------------
+---------------+-----------------------+-------------------------+
| table       | logic_column | logic_data_type | cipher_column | cipher_data_type | plain_column | plain_data_type | assisted_
query_column | assisted_query_data_type | encryptor_type | encryptor_props        | query_with_cipher_column |
+------------+--------------+----------------+-------------+-----------------+-------------+----------------+---------------------+------------------------
+--------------+------------------------+------------------------+
| t_encrypt  | user_id    |           | user_cipher  |           | user_plain  |         |           |               | AES        | aes-key-
value=123456abc | true            |
| t_encrypt  | order_id   |           | order_cipher |           |        |         |           |               | MD5        |              | true
           |
+------------+--------------+----------------+-------------+-----------------+-------------+----------------+---------------------+------------------------
+--------------+------------------------+------------------------+
2 rows in set (0.01 sec))
```

### Shadow

### Syntax

```
SHOW SHADOW shadowRule | RULES [FROM databaseName]

SHOW SHADOW TABLE RULES [FROM databaseName]

SHOW SHADOW ALGORITHMS [FROM databaseName]

shadowRule:
  RULE ruleName
```

- Support querying all shadow rules and specified table query
- Support querying all shadow table rules
- Support querying all shadow algorithms

### Return Value Description

### Shadow Rule

| Column | Description |
|---|---|
| rule_name | Rule name |
| source_name | Source database |
| shadow_name | Shadow database |
| shadow_table | Shadow table |

### Shadow Table Rule

| Column | Description |
|---|---|
| shadow_table | Shadow table |
| shadow_algorithm_name | Shadow algorithm name |

### Shadow Algorithms

| Column | Description |
|---|---|
| shadow_algorithm_name | Shadow algorithm name |
| type | Shadow algorithm type |
| props | Shadow algorithm properties |
| is_default | Default |

### Shadow Rule status

| Column | Description |
|---|---|
| status | Enable |

### Example

SHOW SHADOW RULES

```
mysql> SHOW SHADOW RULES;
+--------------------+-------------+-------------+--------------+
| rule_name          | source_name | shadow_name | shadow_table |
+--------------------+-------------+-------------+--------------+
| shadow_rule_1      | ds_1        | ds_shadow_1 | t_order      |
| shadow_rule_2      | ds_2        | ds_shadow_2 | t_order_item |
+--------------------+-------------+-------------+--------------+
2 rows in set (0.02 sec)
```

SHOW SHADOW RULE ruleName

```
mysql> SHOW SHADOW RULE shadow_rule_1;
+------------------+-------------+-------------+--------------+
| rule_name        | source_name | shadow_name | shadow_table |
+------------------+-------------+-------------+--------------+
| shadow_rule_1    | ds_1        | ds_shadow_1 | t_order      |
+------------------+-------------+-------------+--------------+
1 rows in set (0.01 sec)
```

SHOW SHADOW TABLE RULES

```
mysql> SHOW SHADOW TABLE RULES;
+--------------+-------------------------------------------------------------------------------+
| shadow_table | shadow_algorithm_name                                                         |
+--------------+-------------------------------------------------------------------------------+
| t_order_1    | user_id_match_algorithm,simple_hint_algorithm_1                                      |
+--------------+-------------------------------------------------------------------------------+
1 rows in set (0.01 sec)
```

SHOW SHADOW ALGORITHMS

```
mysql> SHOW SHADOW ALGORITHMS;
+-------------------------+-------------------+-------------------------------------------+----------------+
| shadow_algorithm_name   | type              | props                                     | is_default     |
+-------------------------+-------------------+-------------------------------------------+----------------+
| user_id_match_algorithm | REGEX_MATCH       | operation=insert,column=user_id,regex=[1] | false          |
| simple_hint_algorithm_1 | SIMPLE_HINT       | shadow=true,foo=bar                       | false          |
+-------------------------+-------------------+-------------------------------------------+----------------+
2 rows in set (0.01 sec)
```

## RAL Syntax

RAL (Resource & Rule Administration Language) is responsible for the added-on features of hint, transaction type switch, resharding, sharding execute planning and so on.

### Hint

| Statement | Function | Example |
|---|---|---|
| set read write_splitting hint source = [auto / write] | For current connection, set read/write splitting routing strategy (automatic or forced to write data source) | set readwr ite_splitting hint source = write |
| set sharding hint database_value = yy | For current connection, set sharding value for database sharding only, yy: sharding value | set sharding hint d atabase_value = 100 |
| add sharding hint database_value xx= yy | For current connection, add sharding value for table, xx: logic table, yy: database sharding value | add sharding hint d atabase_value t_order = 100 |
| add sharding hint table_value xx = yy | For current connection, add sharding value for table, xx: logic table, yy: table sharding value | add sharding hint table_value t_order = 100 |
| clear hint | For current connection, clear all hint settings | clear hint |
| clear [sharding hint / read write_splitting hint] | For current connection, clear hint settings of sharding or readwrite splitting | clear readwr ite_splitting hint |
| show [sharding / readw rite_splitting] hint status | For current connection, query hint settings of sharding or readwrite splitting | show readwr ite_splitting hint status |

### Migration

| Statement | Function | Example |
|---|---|---|
| SHOW MIGRATION RULE | Show migration rule | SHOW MIGRATION RULE |
| ALTER MIGRATION RULE | Alter migration rule | ALTER MIGRATION RULE (READ(RATE_LIMITER (TYPE(NAME= 'QPS ' ,PROPERTIES( 'qps' = '5000' ))))) |
| MIGRATE TABLE ds.schema.table INTO table | Migrate table from source to target | MIGRATE TABLE ds_0.public.t_order INTO t_order |
| SHOW MIGRATION LIST | Query running list | SHOW MIGRATION LIST |
| SHOW MIGRATION STATUS jobId | Query migration status | SHOW MIGRATION STATUS 1234 |
| STOP MIGRATION jobId | Stop migration | STOP MIGRATION 1234 |
| START MIGRATION jobId | Start the stopped migration | START MIGRATION 1234 |
| CHECK MIGRATION jobId | Data consistency check | CHECK MIGRATION 1234 |
| SHOW MIGRATION CHECK ALGORITHMS | Show available consistency check algorithms | SHOW MIGRATION CHECK ALGORITHMS |
| CHECK MIGRATION jobId BY TYPE(NAME=a lgorithmTypeName) | Data consistency check with defined algorithm | CHECK MIGRATION 1234 BY TYPE(NAME= "DATA_MATCH" ) |
| SHOW MIGRATION CHECK STATUS jobId | Query data consistency check status | SHOW MIGRATION CHECK STATUS 1234 |
| STOP MIGRATION CHECK jobId | Stop data consistency check | STOP MIGRATION CHECK 1234 |
| START MIGRATION CHECK jobId | Start data consistency check | START MIGRATION CHECK 1234 |
| ROLLBACK MIGRATION jobId | Rollback migration | ROLLBACK MIGRATION 1234 |
| COMMIT MIGRATION jobId | Commit migration | COMMIT MIGRATION 1234 |

## Resharding

| Statement | Function | Example |
|---|---|---|
| SHOW RESHARDING RULE | Show resharding rule | SHOW RESHARDING RULE |
| ALTER RESHARDING RULE | Alter resharding rule | ALTER RESHARDING RULE (READ(RATE_LIMITER (TYPE(NAME= ‘QPS’ ,PROPERTIES( ‘qps’ = ‘5000’ ))))) |
| SHOW RESHARDING LIST | Query running list | SHOW RESHARDING LIST |
| SHOW RESHARDING STATUS jobId | Query resharding status | SHOW RESHARDING STATUS 1234 |
| STOP RESHARDING jobId | Stop resharding | STOP RESHARDING 1234 |
| START RESHARDING jobId | Start resharding | START RESHARDING 1234 |
| CHECK RESHARDING jobId | Data consistency check with algorithm | CHECK RESHARDING 1234 |
| SHOW RESHARDING CHECK ALGORITHMS | Show available consistency check algorithms | SHOW RESHARDING CHECK ALGORITHMS |
| CHECK RESHARDING jobId BY TYPE(NAME=a lgorithmTypeName) | Data consistency check with defined algorithm | CHECK RESHARDING 1234 BY TYPE(NAME=DEFAULT) |
| SHOW RESHARDING CHECK STATUS jobId | Query data consistency check status | SHOW RESHARDING CHECK STATUS 1234 |
| STOP RESHARDING CHECK jobId | Stop data consistency check | STOP RESHARDING CHECK 1234 |
| START RESHARDING CHECK jobId | Start data consistency check | START RESHARDING CHECK 1234 |
| STOP RESHARDING SOURCE WRITING jobId | The source DBPlusEngine data source is discontinued | STOP RESHARDING SOURCE WRITING 1234 |
| FORCE? APPLY RESHARDING jobId | Switch to target DBPlusEngine metadata | APPLY RESHARDING 1234 |
| RESTORE RESHARDING SOURCE WRITING jobId | The source DBPlusEngine data source resumes writing | RESTORE RESHARDING SOURCE WRITING 1234 |
| FORCE? COMMIT RESHARDING jobId | Commit resharding | COMMIT RESHARDING 1234 |
| ROLLBACK RESHARDING jobId | Rollback resharding | ROLLBACK RESHARDING 1234 |

## Encryption

| St atement | Function | Example |
|---|---|---|
| SHOW ENCRYPTING RULE | Show encryption rule | SHOW ENCRYPTING RULE |
| ALTER ENCRYPTING RULE | Alter encryption rule | ALTER ENCRYPTING RULE (READ(RATE_LIMITER (TYPE(N AME= ‘QPS’ ,PROPERTIE S( ‘qps’ = ‘5000’ ))))) |
| SHOW ENCRYPTING LIST; | Query running list | SHOW ENCRYPTING LIST; |
| SHOW ENCRYPTING STATUS { jobId}; | Query task status, jobId: task ID | SHOW ENCRYPTING STATUS 1234; |
| ENCRYPT TABLE {tableName}; | Start running task, tableName: table name | ENCRYPT TABLE t_user; |
| COMMIT ENCRYPTING { jobId}; | Submit the encryption task, jobId: task id. The current version of commit only cleans up jobs, and subsequent versions will support commit and rollback | COMMIT ENCRYPTING 1234; |

## Decryption

| Stat ement | Function | Example |
|---|---|---|
| SHOW DECRYPTING RULE | Show decryption rule | SHOW DECRYPTING RULE |
| ALTER DECRYPTING RULE | Alter decryption rule | ALTER DECRYPTING RULE (READ(RATE_LIMITER (TYPE (NAME= ‘QPS’ ,PROPERTI ES( ‘qps’ = ‘5000’ ))))) |
| SHOW DECRYPTING LIST; | Query running list | SHOW DECRYPTING LIST; |
| SHOW DECRY PTING S TATUS {jobId}; | Query task status, jobId: task ID | SHOW DECRYPTING STATUS 1234; |
| DE CRYPT TABLE {t ableName}; | Start running task, tableName: table name | DECRYPT TABLE t_user; |
| C OMMIT DECRY PTING{jobId}; | Submit the decryption task, jobId: task id. The current version of commit only cleans up jobs, and subsequent versions will support commit and rollback | COMMIT ENCRYPTING 1234; |

### Circuit Breaker

| Statement | Function | Example |
|---|---|---|
| ALTER READWRITE_SPLITTING RULE [ groupName ] (ENABLE / DISABLE) storageUnitName [FROM databaseName] | Enable or disable read data source | ALTER READWRITE_SPLITTING RULE group_1 ENABLE read_ds_1 |
| [ENABLE / DISABLE] COMPUTE NODE instanceId | Enable or disable proxy instance | DISABLE COMPUTE NODE instance_1 |
| SHOW COMPUTE NODES | Query proxy instance information | SHOW COMPUTE NODES |
| SHOW STATUS FROM READWRITE_SPLITTING (RULES / RULE groupName) [FROM databaseName] | Query data sources status of readwrite splitting groups | SHOW STATUS FROM READWRITE_SPLITTING RULES |

## Global Rule

| Statement | Function | Example |
|---|---|---|
| SHOW AUTHORITY RULE | Query authority rule configuration | SHOW AUTHORITY RULE |
| SHOW TRANSACTION RULE | Query transaction rule configuration | SHOW TRANSACTION RULE |
| SHOW SQL_PARSER RULE | Query SQL parser rule configuration | SHOW SQL_PARSER RULE |
| SHOW TRAFFIC[ RULES / RULE ruleName] | Query the specified double routing rules or all double routing rules | SHOW TRAFFIC RULES; |
| CREATE TRAFFIC RULE sql_match_traffic ( LABLES(xxx),TRAFFIC_ALGORITHM(TYPE(NAME=xxx,PROPERTIES( “key” = “value” ))),LOAD_BALANCER(TYPE(NAME=xxx, PROPERTIES( “key” = “value” )))) | Create a dual routing rule,TRAFFIC_ALGORITHM support SQL_MATCH and SQL_HINT two type s;LOAD_BALANCER support RANDOM and ROUND_ROBIN two types. | CREATE TRAFFIC RULE sql_match_traffic (LABLES( OLTP),TRAFFIC_ALGORITHM(TYPE(NAME=SQL_MATCH,PROPERTIES( “sql” = “SELECT * FROM t_order WHERE order_id = ?; UPDATE t_order SET order_id = ?;” ))),LOAD_BALANCER (TYPE(NAME=RANDOM))) |
| ALTER TRANSACTION RULE(DEFAULT=xx,TYPE(NAME=xxx, PROPER TIES( “key1” = “value1” , “key2” = “value2” …))) | Alter transaction rule configur ation，DEFAULT: default transaction type，support LOCAL、XA、BASE; NAME: name of transaction manager, support Atomikos, Narayana and Bitronix | ALTER TRANSACTION RULE(DEFAULT=XA,TYPE(NAME=Narayana, PROPERTIES( “datab aseName” = “jbossts” , “host” = “127.0.0.1” ))) |
| ALTER SQL_PARSER RULE SQL_COMM ENT_PARSE_ENABLE=xx, PARSE_TREE_CACHE( INITIAL_CAPACITY=xx, MAXIMUM_SIZE=xx, CO NCURRENCY_LEVEL=xx), S QL_STATEMENT_CACHE(I NITIAL_CAPACITY=xxx, MAXIMUM_SIZE=xxx, CO NCURRENCY_LEVEL=xxx) | Alter SQL parser rule configuration, SQL_COMMENT_PARSE_ENABLE: whether to parse the SQL comment, PARSE_TREE_CACHE: local cache configuration of syntax tree, S QL_STATEMENT_CACHE: local cache of SQL statement | ALTER SQL_PARSER RULE SQL_COMMENT_PARSE_ENABLE=false, PARSE_TREE_CACHE( INITIAL_CAPACITY=10, MAXIMUM_SIZE=11, C ONCURRENCY_LEVEL=1), SQL_STATEMENT_CACHE( INITIAL_CAPACITY=11, MAXIMUM_SIZE=11, CO NCURRENCY_LEVEL=100) |
| ALTER TRAFFIC RULE sql_match_traffic ( LABLES(xxx),TRAFFIC_ALGORITHM(TYPE(NAME=xxx,PROPERTIES( “key” = “value” ))),LOAD_BALANCER(TYPE(NAME=xxx,PROPERTI ES( “key” = “value” )))) | Modify dual routing rules,TRAFFIC_ALGORITHM support SQL_MATCH and SQL_HINT two type s;LOAD_BALANCER support RANDOM and ROUND_ROBIN two types. | ALTER TRAFFIC RULE sql_match_traffic ( LABLES( OLTP),TRAFFIC_ALGORITHM(TYPE(NAME=SQL_MATCH,PROPERTIES( “sql” = “SELECT * FROM t_order WHERE order_id = ?; UPDATE t_order SET order_id = ?;” ))),LOAD_BALANCER (TYPE(NAME=RANDOM))) |
| DROP TRAFFIC RULE ruleName [, ruleName] | Delete double routing rule | DROP TRAFFIC RULE sql_match_traffic |

## Variable Management

In DistSQL, attribute configuration names are separated by underscores.

| Statement | Function | Example |
|---|---|---|
| SET DIST VARIABLE variable_name = xx | variable_name as Attribute Co nfiguration of proxy | SET VARIABLE sql_show = true |
| SHOW DIST VARIABLES | Query all attribute configurations of the proxy | SHOW ALL VARIABLES |
| SHOW DIST VARIABLE WHERE name = variable_name | Query the specified proxy attribute | SHOW VARIABLE WHERE name = sql_show |

- Special variables

The special variables are not in Attribute Configuration of proxy, However, it can be dynamically managed through DistSQL:

| Statement | Function | Example |
|---|---|---|
| SHOW DIST VARIABLE WHERE name = transaction_type | Query Transaction Type | SHOW DIST VARIABLE WHERE name = transaction_type |
| SET DIST VARIABLE transaction_type = xx | Modify the transaction type of the current connection, support LOCAL, XA, BASE | SET DIST VARIABLE transaction_type = XA |
| SHOW DIST VARIABLE WHERE name = agen t_plugins_enabled | Querying agent plugin enabling status | SHOW DIST VARIABLE WHERE name = agen t_plugins_enabled |
| SET DIST VARIABLE agen t_plugins_enabled = xx | Set the enabling status of the agent plugin. The value is Boolean | SET DIST VARIABLE agen t_plugins_enabled = true |
| SHOW DIST VARIABLE WHERE name = c ached_connections | Query the number of connections currently cached | SHOW DIST VARIABLE WHERE name = c ached_connections |
| SHOW DIST VARIABLE WHERE name = general_query_log | Querying the Full Log Enable Status | SHOW DIST VARIABLE WHERE name = general_query_log |
| SET DIST VARIABLE general_query_log = xx | Set the Full Audit Log Enable Status | SET DIST VARIABLE general_query_log = true |
| SHOW DIST VARIABLE WHERE name = slow_query_log | Querying the Slow Log Enable Status | SHOW DIST VARIABLE WHERE name = slow_query_log |
| SET DIST VARIABLE slow_query_log = xx | Set the Slow Log Enable Status | SET DIST VARIABLE slow_query_log = true |
| SHOW DIST VARIABLE WHERE name = long_query_time | Query Slow Query Threshold | SHOW DIST VARIABLE WHERE name = long_query_time |
| SET DIST VARIABLE long_query_time = xx | Set slow query threshold, unit is milliseconds | SET DIST VARIABLE long_query_time = 5000 |

### Other

| Statement | Function | Example |
|---|---|---|
| SHOW COMPUTE NODE INFO | Query the instance information of the proxy | SHOW COMPUTE NODE INFO |
| SHOW COMPUTE NODE MODE | Query the mode configuration of the proxy | SHOW COMPUTE NODE MODE |
| LABEL COMPUTE NODE instanceId WITH labelName [ , labelName ] | Add tags to instances. | LABEL COMPUTE NODE 1 0.7.7.136@3309 with olap,group_by |
| UNLABEL COMPUTE NODE instanceId | Remove all tags from instances. | UNLABEL COMPUTE NODE 1 0.7.7.136@3309 |
| UNLABEL COMPUTE NODE instanceId WITH labelName | Removes the specified tags from the instance | UNLABEL COMPUTE NODE 1 0.7.7.136@3309 WITH group_by |
| RELABEL COMPUTE NODE instanceId with labelName [ , labelName ] | Readd tags to the instances.(overwrite the original tags) | RELABEL COMPUTE NODE 1 0.7.7.136@3309 with label_9, label_0 |
| REFRESH TABLE METADATA | Refresh the metadata of all tables | REFRESH TABLE METADATA |
| REFRESH TABLE METADATA tableName | Refresh the metadata of the specified table | REFRESH TABLE METADATA t_order |
| REFRESH TABLE METADATA tableName FROM STORAGE UNIT storageUnitName | Refresh the tables’ metadata in the specified data source | REFRESH TABLE METADATA t_order FROM STORAGE UNIT ds_1 |
| REFRESH TABLE METADATA FROM STORAGE UNIT storageUnitName SCHEMA schemaName | Refresh the tables’ metadata in a schema of a specified data source. If there are no tables in the schema, the schema will be deleted. | REFRESH TABLE METADATA FROM STORAGE UNIT ds_1 SCHEMA db_schema |
| SHOW TABLE METADATA tableName [, tableName] … | Query table metadata | SHOW TABLE METADATA t_order |
| EXPORT DATABASE CONFIGURATION [FROM databaseName] [TO FILE “filePath” ] | Export data sources and rule configurations to YAML format | EXPORT DATABASE CONFIGURATION FROM readwrit e_splitting_db |
| IMPORT DATABASE CONFIGURATION FILE= “file_path” | Import data sources and rule configurations from YAML, only supports import into an empty database | IMPORT DATABASE CONFIGURATION FILE = “/xxx/config-sharding.yaml” |
| SHOW RULES USED STORAGE UNIT storageUnitName [FROM databaseName] | Query the rules for using the specified data source in database | SHOW RULES USED STORAGE UNIT ds_0 FROM databaseName |

### Notice

DBPlusEngine-Proxy does not support hint by default, to support it, set proxy-hint-enabled to true in conf/server.yaml.

### RUL Syntax

RUL (Resource Utility Language) responsible for SQL parsing, SQL formatting, preview execution plan and more utility functions.

### SQL Utility

| Statement | Function | Example |
|---|---|---|
| PARSE SQL | Parse SQL and output abstract syntax tree | PARSE SELECT * FROM t_order |
| FORMAT SQL | Parse SQL and output formated SQL statement | FORMAT SELECT * FROM t_order |
| PREVIEW SQL | Preview SQL execution plan | PREVIEW SELECT * FROM t_order |

### Usage

This chapter will introduce how to use DistSQL to manage resources and rules in a distributed database.

### Introduction

We will here use MySQL as example, but the same can be applicable to other databases.

1. Start the MySQL service.
2. Create to be registered MySQL databases.
3. Create role and user in MySQL with creation permission for DBPlusEngine-Proxy.
4. Start Zookeeper service.
5. Add mode and authentication configurations to server.yaml.
6. Start DBPlusEngine-Proxy.
7. Use SDK or terminal connect to DBPlusEngine-Proxy.

### Create Logic Database

1. Create logic database

```sql
CREATE DATABASE foo_db;
```

2. Use newly created logic database

```sql
USE foo_db;
```

### Resource Operation

For more details please see concentrate rule examples.

### Rule Operation

For more details please see concentrate rule examples.

### Notice

1. Currently, DROP DATABASE will only remove the logical distributed database, not the user’ s actual database.
2. DROP TABLE will delete all logical fragmented tables and actual tables in the database.
3. CREATE DATABASE will only create a logical distributed database, so users need to create actual databases in advance.

## Sharding

### Resource Operation

- Configure data source information

```
REGISTER STORAGE UNIT ds_0 (
  HOST="127.0.0.1",
  PORT=3306,
  DB="ds_1",
  USER="root",
  PASSWORD="root"
),ds_1 (
  HOST="127.0.0.1",
  PORT=3306,
  DB="ds_2",
  USER="root",
  PASSWORD="root"
);
```

### Rule Operation

- Create sharding rule

```
CREATE SHARDING TABLE RULE t_order(
STORAGE_UNITS(ds_0,ds_1),
SHARDING_COLUMN=order_id,
TYPE(NAME="hash_mod",PROPERTIES("sharding-count"="4")),
KEY_GENERATE_STRATEGY(COLUMN=order_id,TYPE(NAME="snowflake"))
);
```

- Create sharding table

```
CREATE TABLE `t_order` (
 `order_id` int NOT NULL,
 `user_id` int NOT NULL,
 `status` varchar(45) DEFAULT NULL,
 PRIMARY KEY (`order_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
```

- Drop sharding table

```
DROP TABLE t_order;
```

- Drop sharding rule

```
DROP SHARDING TABLE RULE t_order;
```

- Unregister storage unit

```
UNREGISTER STORAGE UNIT ds_0, ds_1;
```

- Drop distributed database

```
DROP DATABASE foo_db;
```

## readwrite_splitting

### Resource Operation

```
REGISTER STORAGE UNIT write_ds (
   HOST="127.0.0.1",
   PORT=3306,
   DB="ds_0",
   USER="root",
   PASSWORD="root"
),read_ds (
   HOST="127.0.0.1",
   PORT=3307,
   DB="ds_0",
   USER="root",
   PASSWORD="root"
);
```

### Rule Operation

- Create readwrite_splitting rule

```
CREATE READWRITE_SPLITTING RULE group_0 (
WRITE_STORAGE_UNIT=write_ds,
READ_STORAGE_UNITS(read_ds),
TYPE(NAME="random")
);
```

- Alter readwrite_splitting rule

```
ALTER READWRITE_SPLITTING RULE group_0 (
WRITE_STORAGE_UNIT=write_ds,
READ_STORAGE_UNITS(read_ds),
TYPE(NAME="random",PROPERTIES("read_weight:"2:0"))
);
```

- Drop readwrite_splitting rule

```
DROP READWRITE_SPLITTING RULE group_0;
```

- Unregister storage unit

```
UNREGISTER STORAGE UNIT write_ds,read_ds;
```

- Drop distributed database

```
DROP DATABASE readwrite_splitting_db;
```

### Encrypt

### Resource Operation

```
REGISTER STORAGE UNIT ds_0 (
   HOST="127.0.0.1",
   PORT=3306,
   DB="ds_0",
   USER="root",
   PASSWORD="root"
);
```

### Rule Operation

- Create encrypt rule

```
CREATE ENCRYPT RULE t_encrypt (
   COLUMNS(
      (NAME=user_id,PLAIN=user_plain,CIPHER=user_cipher,TYPE(NAME='AES',PROPERTIES('aes-key-value'='123456abc'))),
      (NAME=order_id,PLAIN=order_plain,CIPHER =order_cipher,TYPE(NAME='RC4',PROPERTIES('rc4-key-value'='123456abc')))
));
```

- Create encrypt table

```
CREATE TABLE `t_encrypt` (
   `id` int(11) NOT NULL,
   `user_id` varchar(45) DEFAULT NULL,
   `order_id` varchar(45) DEFAULT NULL,
   PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

- Alter encrypt rule

```
ALTER ENCRYPT RULE t_encrypt (
   COLUMNS(
      (NAME=user_id,PLAIN=user_plain,CIPHER=user_cipher,TYPE(NAME='AES',PROPERTIES('aes-key-value'='123456abc')))
));
```

- Drop encrypt rule

```
DROP ENCRYPT RULE t_encrypt;
```

- Unregister storage unit

```
UNREGISTER STORAGE UNIT ds_0;
```

- Drop distributed database

```
DROP DATABASE encrypt_db;
```

## Shadow

### Resource Operation

```
REGISTER STORAGE UNIT ds_0 (
HOST="127.0.0.1",
PORT=3306,
DB="ds_0",
USER="root",
PASSWORD="root"
),ds_1 (
HOST="127.0.0.1",
PORT=3306,
DB="ds_1",
USER="root",
PASSWORD="root"
),ds_2 (
HOST="127.0.0.1",
PORT=3306,
DB="ds_2",
USER="root",
PASSWORD="root"
);
```

### Rule Operation

- Create shadow rule

```
CREATE SHADOW RULE group_0(
SOURCE=ds_0,
SHADOW=ds_1,
t_order(TYPE(NAME="SIMPLE_HINT", PROPERTIES("foo"="bar")),TYPE(NAME="REGEX_MATCH", PROPERTIES("operation"=
"insert","column"="user_id", "regex"='[1]'))),
t_order_item(TYPE(NAME="SIMPLE_HINT", PROPERTIES("foo"="bar"))));
```

- Alter shadow rule

```
ALTER SHADOW RULE group_0(
SOURCE=ds_0,
SHADOW=ds_2,
t_order_item(TYPE(NAME="SIMPLE_HINT", PROPERTIES("foo"="bar"))));
```

- Drop shadow rule

```
DROP SHADOW RULE group_0;
```

- Unregister storage unit

```
UNREGISTER STORAGE UNIT ds_0,ds_1,ds_2;
```

- Drop distributed database

```
DROP DATABASE foo_db;
```

## DB Discovery

### Resource Operation

```
REGISTER STORAGE UNIT ds_0 (
   HOST="127.0.0.1",
   PORT=3306,
   DB="ds_0",
   USER="root",
   PASSWORD="root"
),ds_1 (
   HOST="127.0.0.1",
   PORT=3306,
   DB="ds_1",
   USER="root",
   PASSWORD="root"
),ds_2 (
   HOST="127.0.0.1",
   PORT=3306,
   DB="ds_2",
   USER="root",
   PASSWORD="root"
);
```

### Rule Operation

- Create DB discovery rule

```
CREATE DB_DISCOVERY RULE db_discovery_group_0 (
STORAGE_UNITS(ds_0, ds_1),
TYPE(NAME='MySQL.MGR',PROPERTIES('group-name'='92504d5b-6dec')),
HEARTBEAT(PROPERTIES('keep-alive-cron'='0/5 * * * * ?'))
);
```

- Alter DB discovery rule

```
ALTER DB_DISCOVERY RULE db_discovery_group_0 (
STORAGE_UNITS(ds_0, ds_1, ds_2),
TYPE(NAME='MySQL.MGR',PROPERTIES('group-name'='92504d5b-6dec')),
HEARTBEAT(PROPERTIES('keep-alive-cron'='0/5 * * * * ?'))
);
```

- Drop db_discovery rule

```
DROP DB_DISCOVERY RULE db_discovery_group_0;
```

- Drop db_discovery type

```
DROP DB_DISCOVERY TYPE db_discovery_group_0_mgr;
```

- Drop db_discovery heartbeat

```
DROP DB_DISCOVERY HEARTBEAT db_discovery_group_0_heartbeat;
```

- Unregister storage unit

```
UNREGISTER STORAGE UNIT ds_0,ds_1,ds_2;
```

- Drop distributed database

```
DROP DATABASE discovery_db;
```

### 7.2.5 Authority Control

#### Authority Classification

| Authorization Item | SELECT | INSERT | UPDATE | DELETE | CREATE | DROP | ALTER | INDEX | CREATE_USER | SUPER |
|---|---|---|---|---|---|---|---|---|---|---|
| Global Authority | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Object Authority/Database | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | / | / |
| Object Authority/Table | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | / | / |
| Object Authority/Column | ✓ | ✓ | ✓ | / | / | / | / | / | / | / |

#### Global Authority

Global authority refers to that the authorization obtained by the user does not distinguish the target object, and the user can perform corresponding operations on any logical database and logical table.

For example, , if the following instruction grants the global INSERT, SELECT, UPDATE and DELETE authorities to user ‘sharding’ @ ‘%’ , the user can perform DML operation on tables in any logical database.

```
-- The following two statements are equivalent.
GRANT DIST INSERT,SELECT,UPDATE,DELETE TO 'sharding'@'%';
GRANT DIST INSERT,SELECT,UPDATE,DELETE TO sharding;
```

Note: The global authority includes two special authorities: CREATE_USER and SUPER.

Get CREATE_USER authorization user can perform the following operations:

| Operation | Description |
|---|---|
| CREATE USER | Create User |
| ALTER USER | Modify User |
| DROP USER | Delete User |
| CREATE ROLE | Create Role |
| DROP ROLE | Delete Role |
| REVOKE ALL PRIVILEGES | Revoke all authorization of a user or role. |

SUPER represents the highest authority of the database system. By default, the initial user configured before ShardingSphere starts has super authorization.

#### Object Authority

Object Authority means that the granted authority is limited by the scope of action, and users cannot perform corresponding operations outside the scope of authorization.

The scope of object authority can be all logical databases, or you can specify single or multiple logical databases and tables and columns in the databases.

For example, the following instructions sets all authorities of t_order in logical database sharding_db to user ‘sharding’ @ ‘%’ . After that the user can operate on table sharding_db.t_order. However the user cannot operate other tables of sharding_db without additional authorization.

```
GRANT DIST ALL ON sharding_db.t_order TO sharding;
```

### DistSQL List

The complete authority management list of DistSQL is as follows:

| Description | Syntax | Examples |
|---|---|---|
| Create user | CREATE DIST USER ‘user’@‘host’ IDENTIFIED BY ‘password’ ; | CREATE DIST USER ‘sharding’ @ ‘%’ IDENTIFIED BY ‘123456’ ; |
| Modify user password | ALTER DIST USER ‘user’ @ ‘host’ IDENTIFIED BY ‘password’ ; | ALTER DIST USER ‘sharding’ @ ‘%’ IDENTIFIED BY ‘sharding’ ; |
| Delete user | DROP DIST USER ‘user’ @ ‘host’ ; | DROP DIST USER ‘sharding’ @ ‘%’ ; |
| Create role | CREATE DIST ROLE role; | CREATE DIST ROLE admin; |
| Delete role | DROP DIST ROLE role; | DROP DIST ROLE admin; |
| Assign user roles | GRANT DIST role TO ‘user’@‘host’ ; | GRANT DIST admin TO ‘sharding’ @ ‘%’ ; |
| Cancel user role | REVOKE DIST role FROM ‘user’ @ ‘host’ ; | REVOKE DIST admin FROM ‘sharding’ @ ‘%’ ; |
| Grant global authorities | GRANT DIST SELECT TP ‘user’ @ ‘host’ ; | GRANT DIST SELECT TO ‘sharding’ @ ‘%’ ; |
| Grant authorities (all objects) | GRANT DIST privileges ON . to ‘user’ @ ‘host’ ; | GRANT DIST SELECT, INSERT ON . TO ‘sharding’ @ ‘%’ ; |
| Grant authority (database level) | GRANT DIST privileges ON schema.* to ‘user’ @ ‘host’ ; | GRANT DIST SELECT, INSERT ON sharding_db.* TO ‘sharding’ @ ‘%’ ; |
| Grant authority (table level) | GRANT DIST privileges ON schema.table to ‘user’ @ ‘host’ ; | GRANT DIST SELECT, INSERT ON sharding_db.t_order TO ‘sharding’ @ ‘%’ ; |
| Grant authority (column level) | GRANT DIST privileges ON schema.table to ‘user’ @ ‘host’ ; | GRANT DIST SELECT (order_id), SELECT (user_id, status) ON sharding_db. t_order TO ‘sharding’ @ ‘%’ ; |
| Revoke all authorities | REVOKE DIST ALL FROM ‘user’ @ ‘host’ ; | REVOKE DIST ALL FROM ‘sharding’ @ ‘%’ ; |
| Revoke global authority | REVOKE DIST privileges FROM ‘user’ @ ‘host’ ; | REVOKE DIST SELECT FROM ‘sharding’ @ ‘%’ ; |
| Revoke authorities (all objects) | REVOKE DIST privileges ON . FROM ‘user’ @ ‘host’ ; | REVOKE DIST SELECT ON . FROM ‘sharding’ @ ‘%’ ; |
| Revoke authority (database level) | REVOKE DIST privileges ON schema.* FROM ‘user’ @ ‘host’ ; | REVOKE DIST INSERT ON sharding_db.* FROM ‘sharding’ @ ‘%’ ; |
| Revoke authority (table level) | REVOKE DIST privileges ON schema.table FROM ‘user’ @ ‘host’ ; | REVOKE DIST INSERT ON sharding_db.t_order FROM ‘sharding’ @ ‘%’ ; |
| Revoke authority (column level) | REVOKE DIST privileges ON schema.table FROM ‘user’ @ ‘host’ ; | REVOKE DIST SELECT (order_id), SELECT (user_id, status) ON sharding_db.t_order FROM ‘sharding’ @ ‘%’ ; |
| View user authority | SHOW DIST GRANTS FOR ‘user’ @ ‘host’ ; | SHOW DIST GRANTS FOR ‘sharding’ @ ‘%’ ; |
| Refresh authority | FLUSH DIST PRIVILEGES; | FLUSH DIST PRIVILEGES; |

Note: - The syntax of “authorize / revoke authorization” for users and roles is the same. Replace ‘user’ @ ‘host’ with role; - REVOKE DIST ALL will revoke all roles assigned to the user at the same time.

Configuration of Authority

It is used to configure the initial user logging in to the computing node and the data authorization of the storage node.

### Example

```
authority:
 users:
  - user: root@%
    password: root
 privilege:
  type: SphereEx:PERMITTED
```

## User Management

### Initial User Management

The initial user is specified by the administrator before the DBPlusEngine starts.

### Ordinary User Management

#### Create User

The CREATE DIST USER statement is used to create a new user with the specified password.

In DBPlusEngine, a user is a combination of a user name and the host to which the user name is connected.

Example: - Create a user named 'sharding', with a password of '123456', '%' indicates that login to the host is not restricted.

```
CREATE DIST USER 'sharding'@'%' IDENTIFIED BY '123456';
```

- Create a user who can only log in at 127.0.0.1.

```
CREATE DIST USER 'sharding'@'127.0.0.1' IDENTIFIED BY '123456';
```

#### Modify User

The ALTER DIST USER statement is used to modify existing users. Currently, it only supports password modification.

Example: - Change the password of the user of 'sharding' @ '%' to 'sharding'.

```
ALTER DIST USER 'sharding'@'%' IDENTIFIED BY 'sharding';
```

#### Delete User

The DROP DIST USER statement is used to delete a user.

Example: - Remove the user of 'sharding' @ '%' from the system.

```
DROP DIST USER 'sharding'@'%';
```

### View User List

The SHOW DIST USERS statement is used to view the user list.

Example: - View all users, only the authorized host and user name are queried, and the user password is not displayed.

```
SHOW DIST USERS;
```

## Role Management

### Create Role

The CREATE DIST ROLE statement is used to create the role.

Example: - Create a role named ‘dml_only’.

```
CREATE DIST ROLE dml_only;
```

### Delete Role

The DROP DIST ROLE statement is used to delete a role.

Example: - Remove user ‘dml_only’ from the system.

```
DROP DIST ROLE dml_only;
```

### View Role List

The SHOW DIST ROLES statement is used to view the role list.

Example: - View all role names that have been created.

```
SHOW DIST ROLES;
```

## Authority Management

### Authorize

The GRANT DIST privileges statement is used to authorize a user or role.

Example: - Grant global SELECT authority to user ‘sharding’@‘%’.

```
GRANT DIST SELECT TO 'sharding'@'%';
```

- Grant SELECT and INSERT authority of logical databse sharding_db to user ‘sharding’@‘%’.

```
GRANT DIST SELECT, INSERT ON sharding_db.* TO 'sharding'@'%';
```

- Grant SELECT and INSERT authority of logical table sharding_db.t_order to user ‘sharding’@‘%’.

```
GRANT DIST SELECT, INSERT ON sharding_db.t_order TO 'sharding'@'%';
```

- Grant the specified SELECT authority in logical table sharding_db.t_order to user ‘sharding’@‘%’.

```
GRANT DIST SELECT (order_id), SELECT (user_id, status) ON sharding_db.t_order TO 'sharding'@'%';
```

- Assign Global SELECT, INSERT, UPDATE and DELETE authority to role dml_only.

```
GRANT DIST INSERT,SELECT,UPDATE,DELETE TO dml_only;
```

- Grant role dml_only to user ‘sharding’ @ ‘%’ .

```
GRANT DIST dml_only TO 'sharding'@'%';
```

### Revocation of Authorization

The REVOKE DIST privileges statement is used to revoke authorization of a user or role.

Example: - Revoke Global Select authorization for user ‘sharding’ @ ‘%’ .

```
REVOKE DIST SELECT FROM 'sharding'@'%';
```

- Revoke SELECT and INSERT authority of user ‘sharding’ @ ‘%’ in logical database sharding_db.

```
REVOKE DIST SELECT, INSERT ON sharding_db.* FROM 'sharding'@'%';
```

- Revoke SELECT and INSERT authority of user ‘sharding’ @ ‘%’ in logical table sharding_db.

```
REVOKE DIST SELECT, INSERT ON sharding_db.t_order FROM 'sharding'@'%';
```

- Revoke SELECT authority of user ‘sharding’ @ ‘%’ in logical sharding_db.t_order.

```
REVOKE DIST SELECT (order_id), SELECT (user_id, status) ON sharding_db.t_order FROM 'sharding'@'%';
```

- Revoke SELECT, INSERT, UPDATE and DELETE authority of role dml_only.

```
REVOKE DIST INSERT,SELECT,UPDATE,DELETE FROM dml_only;
```

- Revoke the role dml_only of user ‘sharding’ @ ‘%’ .

```
REVOKE DIST dml_only FROM 'sharding'@'%';
```

### Related Reference

Authority Control

## 7.2.6 DistSQL Authority Control

### Authority Classification

DistSQL authority classification is similar to standard database authority, they also have global authority, database level object authority and table level object authority. At present, DistSQL does not have column level operations, so there are no column level authority in the authority classification.

DistSQL authority has more types of authorization items than standard database authority. DistSQL authorization item is composed of operation + operation object, and its combination method is as follows.

| | CREATE | ALTER | DROP | SHOW |
|---|---|---|---|---|
| RESOURCE | ✓ | ✓ | ✓ | ✓ |
| SHARDING | ✓ | ✓ | ✓ | ✓ |
| READWRITE_SPLITTING | ✓ | ✓ | ✓ | ✓ |
| ENCRYPT | ✓ | ✓ | ✓ | ✓ |
| DB_DISCOVERY | ✓ | ✓ | ✓ | ✓ |
| SHADOW | ✓ | ✓ | ✓ | ✓ |
| SINGLE_TABLE | ✓ | ✓ | ✓ | ✓ |

In addition to the authorization items shown in the above table, RDL, RQL and RAL syntax types are also provided as authorization items.

### Global Authority

Global authority refers to that the authorization obtained by the user does not distinguish the target object, and the user can perform corresponding operations on any logical database and logical table.

For example, the following instruction grants the global SHOW SHARDING authority to user ‘sharding’@‘%’, and the user can perform SHOW operation on sharding rules in any logical database.

```
GRANT DIST SHOW SHARDING TO 'sharding'@'%';
```

In addition, because there are many DistSQL authorization items, when granting multiple authorization items at a time, you can use the syntax type as the authorization item. For example, the following instruction grants SHOW of all operations object global to user ‘sharding’@‘%’, the user can perform the SHOW operation on any operation object in any logical database.

```
GRANT DIST RQL SHARDING TO 'sharding'@'%';
```

### Object Authority

Object Authority means that the granted authority is limited by the scope of action, and users cannot perform corresponding operations outside the scope of authorization.

The scope of object authority can be all logical databases, or you can specify single or multiple logical databases and tables in databases.

For example, the following instruction grants sharding rule t_order creation and modification in logical database sharding_db to user ‘sharding’@‘%’. After that, the user can create and modify rule sharding_db.t_order. However, the user cannot operate any other rules in sharding_db.

```
GRANT DIST CREATE SHARDING, ALTER SHARDING ON sharding_db.t_order TO 'sharding'@'%';
```

## DistSQL List

| Description | Syntax | Sample |
|---|---|---|
| Create user | CREATE DIST USER ‘user’ @ ‘host’ IDENTIFIED BY ‘password’ ; | CREATE DIST USER ‘sharding’ @ ‘%’ IDENTIFIED BY ‘123456’ ; |
| Modify user Password | ALTER DIST USER ‘user’ @ ‘host’ IDENTIFIED BY ‘password’ ; | ALTER DIST USER ‘sharding’ @ ‘%’ IDENTIFIED BY ‘sharding’ ; |
| Delete user | DROP DIST USER ‘user’ @ ‘host’ ; | DROP DIST USER ‘sharding’ @ ‘%’ ; |
| Create role | CREATE DIST ROLE role; | CREATE DIST ROLE admin; |
| Delete role | DROP DIST ROLE role; | DROP DIST ROLE admin; |
| Grant role to user | GRANT DIST role TO ‘user’ @ ‘host’ ; | GRANT DIST admin TO ‘sharding’ @ ‘%’ ; |
| Cancel user role | REVOKE DIST role FROM ‘user’ @ ‘host’ ; | REVOKE DIST admin FROM ‘sharding’ @ ‘%’ ; |
| Grant global authority | GRANT DIST privileges TO ‘user’ @ ‘host’ ; | GRANT DIST CREATE SHARDING TO ‘sharding’ @ ‘%’ ; |
| Grant authority (all objects) | GRANT DIST privileges ON . to ‘user’ @ ‘host’ ; | GRANT DIST CREATE SHARDING, SHOW SHARDING ON . TO ‘sharding’ @ ‘%’ ; |
| Grant authority (database level) | GRANT DIST privileges ON schema.* to ‘user’ @ ‘host’ ; | GRANT DIST CREATE SHARDING, SHOW SHARDING ON sharding_db.* TO ‘sharding’ @ ‘%’ ; |
| Grant authority (table level) | GRANT DIST privileges ON schema.table to ‘user’ @ ‘host’ ; | GRANT DIST CREATE SHARDING, SHOW SHARDING ON sharding_db.t_order TO ‘sharding’ @ ‘%’ ; |
| Cancel global authority | REVOKE DIST privileges FROM ‘user’ @ ‘host’ ; | REVOKE DIST SHOW SHARDING FROM ‘sharding’ @ ‘%’ ; |
| Cancel authority (all objects) | REVOKE DIST privileges ON . FROM ‘user’ @ ‘host’ ; | REVOKE DIST SHOW SHARDING ON . FROM ‘sharding’ @ ‘%’ ; |
| Cancel authority (database level) | REVOKE DIST privileges ON schema.* FROM ‘user’ @ ‘host’ ; | REVOKE DIST CREATE SHARDING ON sharding_db.* FROM ‘sharding’ @ ‘%’ ; |
| Cancel authority (table level) | REVOKE DIST privileges ON schema.table FROM ‘user’ @ ‘host’ ; | REVOKE DIST CREATE SHARDING ON sharding_db.t_order FROM ‘sharding’ @ ‘%’ ; |
| View user authority | SHOW DIST GRANTS FOR ‘user’ @ ‘host’ ; | SHOW DIST GRANTS FOR ‘sharding’ @ ‘%’ ; |
| Refresh authorization | FLUSH DIST PRIVILEGES; | FLUSH DIST PRIVILEGES; |

The list of DistSQL using syntax type as authorization item is as follows:

| Description | Syntax | Sample |
|---|---|---|
| Grant global authority | GRANT DIST privileges TO ‘user’ @ ‘host’ ; | GRANT DIST RAL, RDL TO ‘sharding’ @ ‘%’ ; |
| Cancel global authority | REVOKE DIST privileges FROM ‘user’ @ ‘host’ ; | REVOKE DIST RAL, RDL FROM ‘sharding’ @ ‘%’ ; |

Note: - The syntax of “authorize / revoke authorization” for users and roles is the same. Replace ‘user’ @ ‘host’ as role; - REVOKE DIST ALL will revoke all roles assigned to the user at the same time.

## 7.2.7 Slow Query Log

### Background

The slow query log feature is used to log SQL statements that takes longer to execute, which is easy for DBAs and developers to identify potentially problematic SQL statements and is an important reference for database and SQL management.

### Parameters

- slow_query_log: Whether to enable the slow query log function, the default value is true.
- long_query_time: The slow query time threshold at which SQL statements that take longer to execute are recorded in the slow query log. The configuration unit is milliseconds (ms) and the default value is 5000.

### Requirements

The slow query log feature is based on Agent technology and is only available for ShardingSphere-Proxy scenarios where Agent is enabled.

### Sample

Because the slow query log is agent-based, the following configuration is located in agent.yaml:

```yaml
plugins:
 Logging:
  props:
   # Whether to enable general query log.
   # general_query_log: true
   # Whether to enable slow query log.
   slow_query_log: true
   # Long query threshold, in milliseconds.
   long_query_time: 5000
```

Where slow_query_log and long_query_time are configured with default values, meaning: 1. Enable slow query logs. 2. When SQL execution takes more than 5000 milliseconds, record slow query logs.

### Slow Query Log Format

The format of the slow query log is as follows:

```
timestamp: {time} db: {db database user: {user} host: {host} query_time: {query time}
{sql}
```

- timestamp: The time at which the log records were generated;
- db: Database name.
- user: The name of the user used in the current connection;
- host: Client access address;
- query_time: SQL execution time, the unit is ms;
- sql: An SQL statement that is sent by the client when a slow query occurs.

For Example:

```
timestamp: 2022-07-01 00:00:00.000 db: sharding_db user: root host: 127.0.0.1 query_time: 159
CREATE TABLE `t_order` (
  `order_id` bigint(20) NOT NULL AUTO_INCREMENT,
  `user_id` int(11) NOT NULL,
  `status` varchar(50) COLLATE utf8mb4_bin DEFAULT NULL,
  PRIMARY KEY (`order_id`)
)
```

### Related References

Observability

## 7.2.8 Full Audit Logs

### Background

Full audit log means that once this function is enabled, the system will record all executed SQL statements, and include the database, user, access address, access time and other information corresponding to the statement. This is convenient for enterprises to conduct audit operations.

### Parameters

- general_query_log: The full audit log has only one parameter. When the value is true, the full log is enabled, and when the value is false, the function is disabled.

### Requirements

Similar to the "slow query log" function, the implementation of the full audit log is also based on the agent, so this function is only applicable to the scenario of ShardingSphere-Proxy and the agent is enabled.

### Sample

The full audit log is an Agent-based feature, so the following configuration is located in agent.yaml:

```
plugins:
  Logging:
    props:
      # Whether to enable general query log.
      general_query_log: false
      # Whether to enable slow query log.
      #slow_query_log: true
      # Long query threshold, in milliseconds.
      #long_query_time: 5000
```

Among them, general_query_log is configured as false, indicating that full logs are not enabled.

When you need to enable full logs, configure general_query_log to true and restart ShardingSphere-Proxy.

### Full Audit Log Format

The full audit log format is as follows:

```
db: {database} user: {user} host: {host} query_time: {query time}
{sql}
```

- db: Database name;
- user: Username used in the current connection;
- host: Client access address;
- query_time: SQL execution time, the unit is ms;
- sql: The SQL statement sent by the client.

For example:

```
[INFO ] 2022-07-01 00:00:00.000 [ShardingSphere-Command-0] GENERAL-QUERY - db: sharding_db user: root host: 127.0.0.1
query_time: 145
CREATE TABLE `t_order` (
 `order_id` bigint(20) NOT NULL AUTO_INCREMENT,
 `user_id` int(11) NOT NULL,
 `status` varchar(50) COLLATE utf8mb4_bin DEFAULT NULL,
 PRIMARY KEY (`order_id`)
)
```

### Related References

Observability

## 7.2.9 Scaling

### Introduction

ShardingSphere-Scaling is a common solution for resharding data in Apache ShardingSphere since version **4.1.0**, the current state is **Experimental** version.

### Build & Deployment

1. Install DBPlusEngine-Proxy.
2. Modify the configuration file conf/server.yaml. For for details please see Mode Configuration.

Currently mode must be ‘Cluster’ and the corresponding registry center needs to be activated in advance.

For example:

```yaml
mode:
 type: Cluster
 repository:
  type: ZooKeeper
  props:
   namespace: governance_ds
   server-lists: localhost:2181
   retryIntervalMilliseconds: 500
   timeToLiveSeconds: 60
   maxRetries: 3
   operationTimeoutMilliseconds: 500
 overwrite: false
```

3. Introduce the DBPlusEngine-Driver driver.

If the backend connects to the following database, download the appropriate DBPlusEngine-Driver driver jar package and put it in the ‘${shardingsphere-proxy}/lib’ directory.

| Database | Driver driver | Reference |
|---|---|---|
| MySQL | **`mysql-co nnector-java-5.1.47.jar < https://repo1.maven.org/m aven2/mysql/mysql-connect or-java/5.1.47/mysql-conn ector-java-5.1.47.jar>`__** | Connector/J Versions |
| openGauss | opengauss-jd bc-2.0.1-compatibility.ja r | |

4. Start DBPlusEngine-Proxy.

```
sh bin/start.sh
```

Check the Proxy logs or log in to the Proxy using the database client to confirm that the Proxy started successfully.

5. Configure RULE on demand.

5.1. Query configuration.

```
SHOW RESHARDING RULE;
```

The default configuration is as follows:

```
+-------------------------------------------------------------+--------------------------------------+------------------------------------------------------+
| read                                         | write                          | stream_channel                               |
+-------------------------------------------------------------+--------------------------------------+------------------------------------------------------+
| {"workerThread":40,"batchSize":1000,"shardingSize":10000000} | {"workerThread":40,"batchSize":1000} | {"type":"MEMORY",
"props":{"block-queue-size":10000}} |
+-------------------------------------------------------------+--------------------------------------+------------------------------------------------------+
```

5.2. Alter configuration (Optional).

Since the migration rule has default values, there is no need to create it, only the ALTER statement is provided.

A completely configured DistSQL is as follows.

```
ALTER RESHARDING RULE (
READ(
 WORKER_THREAD=40,
 BATCH_SIZE=1000,
 SHARDING_SIZE=10000000,
 RATE_LIMITER (TYPE(NAME='QPS',PROPERTIES('qps'='500')))
),
WRITE(
 WORKER_THREAD=40,
 BATCH_SIZE=1000,
 RATE_LIMITER (TYPE(NAME='TPS',PROPERTIES('tps'='2000')))
),
STREAM_CHANNEL (TYPE(NAME='MEMORY',PROPERTIES('block-queue-size'='10000')))
);
```

Configuration item description:

```
ALTER RESHARDING RULE (
READ( -- Data reading configuration. If it is not configured, part of the parameters will take effect by default.
 WORKER_THREAD=40, -- Obtain the thread pool size of all the data from the source side. If it is not configured, the default value is used.
 BATCH_SIZE=1000, -- The maximum number of records returned by a query operation. If it is not configured, the default value is used.
 SHARDING_SIZE=10000000, -- Sharding size of all the data. If it is not configured, the default value is used.
 RATE_LIMITER ( -- Traffic limit algorithm. If it is not configured, traffic is not limited.
 TYPE( -- Algorithm type. Option: QPS
 NAME='QPS',
 PROPERTIES( -- Algorithm property
 'qps'='500'
 )))
),
WRITE( -- Data writing configuration. If it is not configured, part of the parameters will take effect by default.
 WORKER_THREAD=40, -- The size of the thread pool on which data is written into the target side. If it is not configured, the default value is used.
 BATCH_SIZE=1000, -- The maximum number of records for a batch write operation. If it is not configured, the default value is used.
 RATE_LIMITER ( -- Traffic limit algorithm. If it is not configured, traffic is not limited.
 TYPE( -- Algorithm type. Option: TPS
 NAME='TPS',
 PROPERTIES( -- Algorithm property.
 'tps'='2000'
 )))
),
STREAM_CHANNEL ( -- Data channel. It connects producers and consumers, used for reading and writing procedures. If it is not configured, the MEMORY type is used by default.
TYPE( -- Algorithm type. Option: MEMORY
NAME='MEMORY',
PROPERTIES( -- Algorithm property
'block-queue-size'='10000' -- Property: blocking queue size.
)))
);
```

DistSQL sample: configure READ for traffic limit.

```
ALTER RESHARDING RULE (
READ(
 RATE_LIMITER (TYPE(NAME='QPS',PROPERTIES('qps'='500')))
)
);
```

Configure data reading for traffic limit. Other configurations use default values.

5.3. Restore configuration.

```
ALTER RESHARDING RULE (
READ(
 WORKER_THREAD=40,
 BATCH_SIZE=1000,
 SHARDING_SIZE=10000000,
 RATE_LIMITER (TYPE(NAME='QPS',PROPERTIES('qps'='500')))
),
WRITE(
 WORKER_THREAD=40,
 BATCH_SIZE=1000,
 RATE_LIMITER (TYPE(NAME='TPS',PROPERTIES('tps'='2000')))
),
STREAM_CHANNEL (TYPE(NAME='MEMORY',PROPERTIES('block-queue-size'='10000')))
);
```

## Manual

### MySQL User Manual

#### Environment

MySQL 5.1.15 ~ 8.0.x.

#### Privileges

1. Start binlog.

MySQL 5.7 my.cnf configuration example:

```
[mysqld]
server-id=1
log-bin=mysql-bin
binlog-format=row
binlog-row-image=full
max_connections=600
```

Run the following command to confirm that binlog is enabled:

```
show variables like '%log_bin%';
show variables like '%binlog%';
```

If you receive the following result, it means that the binlog is enabled:

```
+-----------------------------------------+--------------------------------------+
| Variable_name                | Value                  |
+-----------------------------------------+--------------------------------------+
| log_bin                 | ON                    |
| binlog_format                | ROW                     |
| binlog_row_image                | FULL                     |
+-----------------------------------------+--------------------------------------+
```

2. Grant the MySQL account Replication permission.

Execute the following command to see if the user has migration permissions:

```
SHOW GRANTS FOR 'user';
```

Example results:

```
+------------------------------------------------------------------------------+
|Grants for ${username}@${host}                                     |
+------------------------------------------------------------------------------+
|GRANT REPLICATION SLAVE, REPLICATION CLIENT ON *.* TO ${username}@${host}    |
|.......                                            |
+------------------------------------------------------------------------------+
```

## Complete Procedure Example

### Prerequisite

1. Create a database in MySQL.

Example:

```
DROP DATABASE IF EXISTS resharding_ds_0;
CREATE DATABASE resharding_ds_0 DEFAULT CHARSET utf8;

DROP DATABASE IF EXISTS resharding_ds_1;
CREATE DATABASE resharding_ds_1 DEFAULT CHARSET utf8;

DROP DATABASE IF EXISTS resharding_ds_2;
CREATE DATABASE resharding_ds_2 DEFAULT CHARSET utf8;
```

### Procedure

1. Create a new logical database in proxy, configure resources and rules, create tables and initialize some data.

```
CREATE DATABASE sharding_db;

USE sharding_db;

REGISTER STORAGE UNIT ds_0 (
  URL="jdbc:mysql://127.0.0.1:3306/resharding_ds_0?useServerPrepStmts=true&serverTimezone=UTC&useSSL=false&
characterEncoding=utf-8",
  USER="root",
  PASSWORD="root"
), ds_1 (
  URL="jdbc:mysql://127.0.0.1:3306/resharding_ds_1?useServerPrepStmts=true&serverTimezone=UTC&useSSL=false&
characterEncoding=utf-8",
  USER="root",
  PASSWORD="root"
);

CREATE SHARDING TABLE RULE t_order(
STORAGE_UNITS(ds_0,ds_1),
SHARDING_COLUMN=order_id,
TYPE(NAME="hash_mod",PROPERTIES("sharding-count"="4")),
KEY_GENERATE_STRATEGY(COLUMN=order_id,TYPE(NAME="snowflake"))
);

CREATE TABLE t_order (order_id INT NOT NULL, user_id INT NOT NULL, status VARCHAR(45) NULL, PRIMARY KEY (order_id));

INSERT INTO t_order (order_id, user_id, status) VALUES (1,2,'ok'),(2,4,'ok'),(3,6,'ok'),(4,1,'ok'),(5,3,'ok'),(6,5,'ok');
```

2. Add a new data source in the proxy.

Example:

```
REGISTER STORAGE UNIT ds_2 (
  URL="jdbc:mysql://127.0.0.1:3306/resharding_ds_2?useServerPrepStmts=true&serverTimezone=UTC&useSSL=false&
characterEncoding=utf-8",
  USER="root",
  PASSWORD="root"
);
```

3. Start scaling-in and scaling-out.

Added ds_2 data source, example:

```
RESHARD TABLE t_order BY(
  STORAGE_UNITS(ds_0, ds_1, ds_2),
  SHARDING_COLUMN=order_id,
  TYPE(NAME="hash_mod",PROPERTIES("sharding-count"="6")),
  KEY_GENERATE_STRATEGY(COLUMN=order_id,TYPE(NAME="snowflake"))
);
```

4. View the list of scaling-in and scaling-out operations.

```
SHOW RESHARDING LIST;
```

Example results:

```
+---------------------------------------+---------+----------------------+--------+---------------------+-----------+
| id                                    | tables  | sharding_total_count | active | create_time         | stop_time |
+---------------------------------------+---------+----------------------+--------+---------------------+-----------+
| j51017f973ac82cb1edea4f5238a258c25e89 | t_order | 2                    | true   | 2022-10-25 10:10:58 | NULL      |
+---------------------------------------+---------+----------------------+--------+---------------------+-----------+
```

5. View the details of scaling-in and scaling-out.

```
SHOW RESHARDING STATUS 'j51017f973ac82cb1edea4f5238a258c25e89';
```

Example results:

```
+------+-------------+--------------------------+--------+-------------------------+-------------------------------+-------------------+--------------------------+----
-----------+
| item | data_source | status                   | active | processed_records_count | inventory_finished_percentage | remaining_seconds |
incremental_idle_seconds | error_message |
+------+-------------+--------------------------+--------+-------------------------+-------------------------------+-------------------+--------------------------+----
-----------+
| 0    | ds_0        | EXECUTE_INCREMENTAL_TASK | true   | 3                       | 100                           | 0                 | 92                       |               |
| 1    | ds_1        | EXECUTE_INCREMENTAL_TASK | true   | 3                       | 100                           | 0                 | 92                       |               |
+------+-------------+--------------------------+--------+-------------------------+-------------------------------+-------------------+--------------------------+----
-----------+
```

6. Execute write stop (Optional).

This statement will intercept the addition, deletion, modification and partial DistSQL, which is optional.

```
STOP RESHARDING SOURCE WRITING 'j51017f973ac82cb1edea4f5238a258c25e89';
```

7. Perform data consistency verification.

```
CHECK RESHARDING 'j51017f973ac82cb1edea4f5238a258c25e89' BY TYPE (NAME='CRC32_MATCH');
```

Data consistency verification algorithm type comes from:

```
SHOW RESHARDING CHECK ALGORITHMS;
+-------------+--------------------------------------------------------------+----------------------------+
| type        | supported_database_types                                     | description                |
+-------------+--------------------------------------------------------------+----------------------------+
| CRC32_MATCH | MySQL                                                        | Match CRC32 of records.    |
| DATA_MATCH  | SQL92,MySQL,MariaDB,PostgreSQL,openGauss,Oracle,SQLServer,H2 | Match raw data of records. |
+-------------+--------------------------------------------------------------+----------------------------+
```

When data encryption is enabled, you need to use DATA_MATCH.

Query data consistency verification progress:

```
SHOW RESHARDING CHECK STATUS 'j51017f973ac82cb1edea4f5238a258c25e89';
```

Example results:

```
+---------+--------+---------------------+-------------------+-------------------------+-------------------------+------------------+---------------+
| tables  | result | finished_percentage | remaining_seconds | check_begin_time        | check_end_time          | duration_seconds |
error_message |
+---------+--------+---------------------+-------------------+-------------------------+-------------------------+------------------+---------------+
| t_order | true   | 100                 | 0                 | 2022-10-25 10:13:33.220 | 2022-10-25 10:13:35.338 | 2                |               |
+---------+--------+---------------------+-------------------+-------------------------+-------------------------+------------------+---------------+
```

8. Switch metadata.

```
APPLY RESHARDING 'j51017f973ac82cb1edea4f5238a258c25e89';
```

Preview the rule after it takes effect.

```
PREVIEW SELECT * FROM t_order;
```

Example results:

```
+------------------+---------------------------------------------------------------+
| data_source_name | actual_sql                                                    |
+------------------+---------------------------------------------------------------+
| ds_0             | select * from v1_t_order_0 UNION ALL select * from v1_t_order_3 |
| ds_1             | select * from v1_t_order_1 UNION ALL select * from v1_t_order_4 |
| ds_2             | select * from v1_t_order_2 UNION ALL select * from v1_t_order_5 |
+------------------+---------------------------------------------------------------+
```

9. Recover write stop (optional).

If the write was stopped before, recovery is required.

```
RESTORE RESHARDING SOURCE WRITING 'j51017f973ac82cb1edea4f5238a258c25e89';
```

10. Complete jobs of scaling-in and scaling-out.

Example:

```
COMMIT RESHARDING 'j51017f973ac82cb1edea4f5238a258c25e89';
```

## PostgreSQL User Manual

### Environment

PostgreSQL 9.4+

### Privileges

1. Start test_decoding.
2. Adjust the WAL configuration.

postgresql.conf configuration example:

```
wal_level = logical
max_wal_senders = 10
max_replication_slots = 10
max_connections = 600
```

For details please see Write Ahead Log and Replication.

3. Configuring PostgreSQL allows the proxy to have replication permissions.

pg_hba.conf configuration example:

```
host replication repl_acct 0.0.0.0/0 md5
```

For details please see The pg_hba.conf File.

### Complete Procedure Example

#### Prerequisite

1. Create a database in PostgreSQL.

Example:

```
DROP DATABASE IF EXISTS resharding_ds_0;
CREATE DATABASE resharding_ds_0;

DROP DATABASE IF EXISTS resharding_ds_1;
CREATE DATABASE resharding_ds_1;

DROP DATABASE IF EXISTS resharding_ds_2;
CREATE DATABASE resharding_ds_2;
```

#### Procedure

1. Create a new logical database in proxy, configure resources and rules, create tables and initialize some data.

```
CREATE DATABASE sharding_db;

USE sharding_db;

REGISTER STORAGE UNIT ds_0 (
  URL="jdbc:postgresql://127.0.0.1:5432/resharding_ds_0",
  USER="root",
  PASSWORD="root"
), ds_1 (
  URL="jdbc:postgresql://127.0.0.1:5432/resharding_ds_1",
  USER="root",
  PASSWORD="root"
);

CREATE SHARDING TABLE RULE t_order(
STORAGE_UNITS(ds_0,ds_1),
SHARDING_COLUMN=order_id,
TYPE(NAME="hash_mod",PROPERTIES("sharding-count"="4")),
KEY_GENERATE_STRATEGY(COLUMN=order_id,TYPE(NAME="snowflake"))
);

CREATE TABLE t_order (order_id INT NOT NULL, user_id INT NOT NULL, status VARCHAR(45) NULL, PRIMARY KEY (order_id));

INSERT INTO t_order (order_id, user_id, status) VALUES (1,2,'ok'),(2,4,'ok'),(3,6,'ok'),(4,1,'ok'),(5,3,'ok'),(6,5,'ok');
```

2. Add a new data source in the proxy.

Example:

```
REGISTER STORAGE UNIT ds_2 (
  URL="jdbc:postgresql://127.0.0.1:5432/resharding_ds_2",
  USER="root",
  PASSWORD="root"
);
```

3. Start scaling-in and scaling-out.

Added data source ds_2, example:

```
RESHARD TABLE t_order BY(
  STORAGE_UNITS(ds_0, ds_1, ds_2),
  SHARDING_COLUMN=order_id,
  TYPE(NAME="hash_mod",PROPERTIES("sharding-count"="6")),
  KEY_GENERATE_STRATEGY(COLUMN=order_id,TYPE(NAME="snowflake"))
);
```

4. View the list of scaling-in and scaling-out operations.

```
SHOW RESHARDING LIST;
```

Example results:

```
+---------------------------------------+---------+----------------------+--------+---------------------+-----------+
| id                                    | tables  | sharding_total_count | active | create_time         | stop_time |
+---------------------------------------+---------+----------------------+--------+---------------------+-----------+
| j51017f973ac82cb1edea4f5238a258c25e89 | t_order | 2                    | true   | 2022-10-25 10:10:58 | NULL      |
+---------------------------------------+---------+----------------------+--------+---------------------+-----------+
```

5. View the details of scaling-in and scaling-out.

```
SHOW RESHARDING STATUS 'j51017f973ac82cb1edea4f5238a258c25e89';
```

Example results:

```
+------+-------------+--------------------------+--------+-------------------------+-------------------------------+-------------------+--------------------------+---------------+
| item | data_source | status                   | active | processed_records_count | inventory_finished_percentage | remaining_seconds | incremental_idle_seconds | error_message |
+------+-------------+--------------------------+--------+-------------------------+-------------------------------+-------------------+--------------------------+---------------+
| 0    | ds_0        | EXECUTE_INCREMENTAL_TASK | true   | 3                       | 100                           | 0                 | 92                       |               |
| 1    | ds_1        | EXECUTE_INCREMENTAL_TASK | true   | 3                       | 100                           | 0                 | 92                       |               |
+------+-------------+--------------------------+--------+-------------------------+-------------------------------+-------------------+--------------------------+---------------+
```

6. Execute write stop (optional).

This statement will intercept the addition, deletion, modification and partial DistSQL, which is optional.

```
STOP RESHARDING SOURCE WRITING 'j51017f973ac82cb1edea4f5238a258c25e89';
```

7. Perform data consistency verification.

```
CHECK RESHARDING 'j51017f973ac82cb1edea4f5238a258c25e89';
```

Query data consistency verification progress:

```
SHOW RESHARDING CHECK STATUS 'j51017f973ac82cb1edea4f5238a258c25e89';
```

Example results:

```
+---------+--------+---------------------+-------------------+-------------------------+-------------------------+------------------+---------------+
| tables  | result | finished_percentage | remaining_seconds | check_begin_time        | check_end_time          | duration_seconds | error_message |
+---------+--------+---------------------+-------------------+-------------------------+-------------------------+------------------+---------------+
| t_order | true   | 100                 | 0                 | 2022-10-25 10:13:33.220 | 2022-10-25 10:13:35.338 | 2                |               |
+---------+--------+---------------------+-------------------+-------------------------+-------------------------+------------------+---------------+
```

8. Switch metadata.

```
APPLY RESHARDING 'j51017f973ac82cb1edea4f5238a258c25e89';
```

Preview the rule after it takes effect.

```
PREVIEW SELECT * FROM t_order;
```

Example results:

```
+------------------+----------------------------------------------------------------+
| data_source_name | actual_sql                                                     |
+------------------+----------------------------------------------------------------+
| ds_0             | select * from v1_t_order_0 UNION ALL select * from v1_t_order_3 |
| ds_1             | select * from v1_t_order_1 UNION ALL select * from v1_t_order_4 |
| ds_2             | select * from v1_t_order_2 UNION ALL select * from v1_t_order_5 |
+------------------+----------------------------------------------------------------+
```

9. Recover write stop (optional).

If the write was stopped before, recovery is required.

```
RESTORE RESHARDING SOURCE WRITING 'j51017f973ac82cb1edea4f5238a258c25e89';
```

10. Complete jobs of scaling-in and scaling-out.

Example:

```
COMMIT RESHARDING 'j51017f973ac82cb1edea4f5238a258c25e89';
```

### openGauss User Manual

#### Environment

openGauss 2.0.1 ~ 3.0.0

#### Privileges

1. Adjust the WAL configuration.

postgresql.conf configuration example:

```
wal_level = logical
max_wal_senders = 10
max_replication_slots = 10
wal_sender_timeout = 0
max_connections = 600
```

For details please see Write Ahead Log and Replication.

2. Configuring PostgreSQL allows the proxy to have replication permissions.

pg_hba.conf configuration example:

```
host replication repl_acct 0.0.0.0/0 md5
```

For details please see Configuring Client Access Authentication and Example: Logic Replication Code.

## Complete Procedure Example

### Prerequisite

1. Create a database in openGauss.

Example:

```
DROP DATABASE IF EXISTS resharding_ds_0;
CREATE DATABASE resharding_ds_0;

DROP DATABASE IF EXISTS resharding_ds_1;
CREATE DATABASE resharding_ds_1;

DROP DATABASE IF EXISTS resharding_ds_2;
CREATE DATABASE resharding_ds_2;
```

### Procedure

1. Create a new logical database in proxy, configure resources and rules, create tables and initialize some data.

```
CREATE DATABASE sharding_db;

USE sharding_db;

REGISTER STORAGE UNIT ds_0 (
   URL="jdbc:opengauss://127.0.0.1:5432/resharding_ds_0",
   USER="root",
   PASSWORD="root"
), ds_1 (
   URL="jdbc:opengauss://127.0.0.1:5432/resharding_ds_1",
   USER="root",
   PASSWORD="root"
);

CREATE SHARDING TABLE RULE t_order(
STORAGE_UNITS(ds_0,ds_1),
SHARDING_COLUMN=order_id,
TYPE(NAME="hash_mod",PROPERTIES("sharding-count"="4")),
KEY_GENERATE_STRATEGY(COLUMN=order_id,TYPE(NAME="snowflake"))
);

CREATE TABLE t_order (order_id INT NOT NULL, user_id INT NOT NULL, status VARCHAR(45) NULL, PRIMARY KEY (order_id));

INSERT INTO t_order (order_id, user_id, status) VALUES (1,2,'ok'),(2,4,'ok'),(3,6,'ok'),(4,1,'ok'),(5,3,'ok'),(6,5,'ok');
```

2. Add a new data source in the proxy

Example:

```
REGISTER STORAGE UNIT ds_2 (
   URL="jdbc:opengauss://127.0.0.1:5432/resharding_ds_2",
   USER="root",
   PASSWORD="root"
);
```

3. Start scaling-in and scaling-out.

Added ds_2 data source, example:

```
RESHARD TABLE t_order BY(
   STORAGE_UNITS(ds_0, ds_1, ds_2),
   SHARDING_COLUMN=order_id,
   TYPE(NAME="hash_mod",PROPERTIES("sharding-count"="6")),
   KEY_GENERATE_STRATEGY(COLUMN=order_id,TYPE(NAME="snowflake"))
);
```

4. View the list of scaling-in and scaling-out operations.

```
SHOW RESHARDING LIST;
```

Example results:

```
+---------------------------------------+---------+----------------------+--------+---------------------+-----------+
| id                                    | tables  | sharding_total_count | active | create_time         | stop_time |
+---------------------------------------+---------+----------------------+--------+---------------------+-----------+
| j51017f973ac82cb1edea4f5238a258c25e89 | t_order | 2                    | true   | 2022-10-25 10:10:58 | NULL      |
+---------------------------------------+---------+----------------------+--------+---------------------+-----------+
```

5. View the details of scaling-in and scaling-out.

```
SHOW RESHARDING STATUS 'j51017f973ac82cb1edea4f5238a258c25e89';
```

Example results:

```
+------+-------------+--------------------------+--------+-------------------------+-------------------------------+-------------------+--------------------------+----
-----------+
| item | data_source | status                   | active | processed_records_count | inventory_finished_percentage | remaining_seconds |
incremental_idle_seconds | error_message |
+------+-------------+--------------------------+--------+-------------------------+-------------------------------+-------------------+--------------------------+----
-----------+
| 0    | ds_0        | EXECUTE_INCREMENTAL_TASK | true   | 3                       | 100                           | 0                 | 92                       |               |
| 1    | ds_1        | EXECUTE_INCREMENTAL_TASK | true   | 3                       | 100                           | 0                 | 92                       |               |
+------+-------------+--------------------------+--------+-------------------------+-------------------------------+-------------------+--------------------------+----
-----------+
```

6. Execute write stop (optional)

This statement will intercept the addition, deletion, modification and partial DistSQL, which is optional.

```
STOP RESHARDING SOURCE WRITING 'j51017f973ac82cb1edea4f5238a258c25e89';
```

7. Perform data consistency verification.

```
CHECK RESHARDING 'j51017f973ac82cb1edea4f5238a258c25e89';
```

Query data consistency verification progress:

```
SHOW RESHARDING CHECK STATUS 'j51017f973ac82cb1edea4f5238a258c25e89';
```

Example results:

```
+---------+--------+---------------------+-------------------+-------------------------+-------------------------+------------------+---------------+
| tables  | result | finished_percentage | remaining_seconds | check_begin_time        | check_end_time          | duration_seconds |
error_message |
+---------+--------+---------------------+-------------------+-------------------------+-------------------------+------------------+---------------+
| t_order | true   | 100                 | 0                 | 2022-10-25 10:13:33.220 | 2022-10-25 10:13:35.338 | 2                |               |
+---------+--------+---------------------+-------------------+-------------------------+-------------------------+------------------+---------------+
```

8. Switch metadata.

```
APPLY RESHARDING 'j51017f973ac82cb1edea4f5238a258c25e89';
```

Preview the rule after it takes effect.

```
PREVIEW SELECT * FROM t_order;
```

Example results:

```
+------------------+-----------------------------------------------------------------+
| data_source_name | actual_sql                                                      |
+------------------+-----------------------------------------------------------------+
| ds_0             | select * from v1_t_order_0 UNION ALL select * from v1_t_order_3 |
| ds_1             | select * from v1_t_order_1 UNION ALL select * from v1_t_order_4 |
| ds_2             | select * from v1_t_order_2 UNION ALL select * from v1_t_order_5 |
+------------------+-----------------------------------------------------------------+
```

9. Recover write stop (optional).

If the write was stopped before, recovery is required.

```
RESTORE RESHARDING SOURCE WRITING 'j51017f973ac82cb1edea4f5238a258c25e89';
```

10. Complete scaling-in and scaling-out.

Example:

```
COMMIT RESHARDING 'j51017f973ac82cb1edea4f5238a258c25e89';
```

## 7.2.10 Auto Scaling on Cloud (HPA)

### Parameters

| Name | Description | Default Value |
|---|---|---|
| a utomaticScal ing. enable | SphereEx-DBPlusEngine-Proxy whether the cluster starts auto scaling | false |
| automatic Scaling.scal eUpWindows | SphereEx-DBPlusEngine-Proxy auto scaling-out stable window | 30 |
| ` automaticScaling.scaleD ownWindows` | SphereEx-DBPlusEngine-Proxy auto scaling-in stable window | 30 |
| a utomaticScal ing. target | SphereEx-DBPlusEngine-Proxy threshold value of auto scaling is percentage. Note: at this stage, only CPU is supported for scaling. | 70 |
| automa ticScaling.m axInstance | SphereEx-DBPlusEngine-Proxy maximum number of scaling-out copies | 4 |
| automa ticScaling.m inInstance | SphereEx-DBPlusEngine-Proxy minimum number of startup copies, and the scaling-in size will not be less than this number of copies | 1 |

### Notes

When you turn on the automaticScaling function of SphereEx-DBPlusEngine, HPA will take over the number of copies of SphereEx-DBPlusEngine. Your SphereEx-DBPlusEngine may scale-in, and the application will flash.

When the automaticScaling function of SphereEx-DBPlusEngine is enabled, the corresponding HPA will be deleted.

### Procedure

- After modifying values.yaml according to the following configuration, execute helm install to create a new SphereEx-DBPlusEngine cluster.
- Or use helm upgrade to update the existing SphereEx-DBPlusEngine cluster configuration.

### Sample

If you want to turn on the auto scaling function of SphereEx-DBPlusEngine in SphereEx-Operator, you need to open the following configuration in the values.yaml of SphereEx-DBPlusEngine-cluster charts.

```yaml
automaticScaling:
  enable: true
  scaleUpWindows: 30
  scaleDownWindows: 30
  target: 20
  maxInstance: 4
  minInstance: 2
```

### Related References

- Feature Description of Auto Scaling on Cloud (HPA)
- Use Operator

## 7.2.11 Data Migration

### Introduction

DBPlusEngine provides solution of migrating data since **4.1.0**.

### Build

### Background

For systems running on a single database that urgently need to securely and simply migrate data to a horizontally sharded database.

### Prerequisites

- Proxy is developed in JAVA, and JDK version 1.8 or later is recommended.
- Data migration adopts the cluster mode, and ZooKeeper is currently supported as the registry.

### Procedure

1. Run the following command to compile the ShardingSphere-Proxy binary package:

```
git clone --depth 1 https://github.com/apache/shardingsphere.git
cd shardingsphere
mvn clean install -Dmaven.javadoc.skip=true -Dcheckstyle.skip=true -Drat.skip=true -Djacoco.skip=true -DskipITs -DskipTests -
Prelease
```

Release package： - /shardingsphere-distribution/shardingsphere-proxy-distribution/target/apache-shardingsphere-${latest.release.version}-shardingsphere-proxy-bin.tar.gz

Or you can get the installation package through the Download Page

2. Decompress the proxy release package and modify the configuration file conf/config-sharding.yaml. Please refer to proxy startup guide for details.

3. Modify the configuration file conf/server.yaml. Please refer to mode configuration for details.

Currently, mode must be Cluster, and the corresponding registry must be started in advance.

Configuration sample:

```
mode:
 type: Cluster
 repository:
  type: ZooKeeper
  props:
   namespace: governance_ds
   server-lists: localhost:2181
   retryIntervalMilliseconds: 500
   timeToLiveSeconds: 60
   maxRetries: 3
   operationTimeoutMilliseconds: 500
 overwrite: false
```

4. Introduce the JDBC driver.

Proxy includes the JDBC driver of PostgreSQL.

If the backend is connected to the following databases, download the corresponding JDBC driver jar package and put it into the ${shardingsphere-proxy}/ext-lib directory.

| Database | JDBC Driver | Reference |
|---|---|---|
| MySQL | **`mysql-co nnector-java-5.1.47.jar < https://repo1.maven.org/m aven2/mysql/mysql-connect or-java/5.1.47/mysql-conn ector-java-5.1.47.jar>`__** | Connector/J Versions |
| openGauss | opengauss-jdbc-3.0.0 .jar | |

If you are migrating to a heterogeneous database, then you could use more types of database, e.g. Oracle. Introduce JDBC driver as above too.

5. Start ShardingSphere-Proxy:

```
sh bin/start.sh
```

6. View the proxy log logs/stdout.log. If you see the following statements:

```
[INFO ] [main] o.a.s.p.frontend.ShardingSphereProxy - ShardingSphere-Proxy start success
```

The startup will have been successful.

7. Configure and migrate on demand.

7.1. Query configuration.

```
SHOW MIGRATION RULE;
```

The default configuration is as follows.

```
+--------------------------------------------------------------+---------------------------------------+------------------------------------------------------+
| read                                    | write                       | stream_channel                         |
+--------------------------------------------------------------+---------------------------------------+------------------------------------------------------+
| {"workerThread":40,"batchSize":1000,"shardingSize":10000000} | {"workerThread":40,"batchSize":1000} | {"type":"MEMORY",
"props":{"block-queue-size":10000}} |
+--------------------------------------------------------------+---------------------------------------+------------------------------------------------------+
```

7.2. Alter configuration (Optional).

Since the migration rule has default values, there is no need to create it, only the ALTER statement is provided.

A completely configured DistSQL is as follows.

```
ALTER MIGRATION RULE (
READ(
 WORKER_THREAD=40,
 BATCH_SIZE=1000,
 SHARDING_SIZE=10000000,
 RATE_LIMITER (TYPE(NAME='QPS',PROPERTIES('qps'='500')))
),
WRITE(
 WORKER_THREAD=40,
 BATCH_SIZE=1000,
 RATE_LIMITER (TYPE(NAME='TPS',PROPERTIES('tps'='2000')))
),
STREAM_CHANNEL (TYPE(NAME='MEMORY',PROPERTIES('block-queue-size'='10000')))
);
```

Configuration item description:

```
ALTER MIGRATION RULE (
READ( -- Data reading configuration. If it is not configured, part of the parameters will take effect by default.
 WORKER_THREAD=40, -- Obtain the thread pool size of all the data from the source side. If it is not configured, the default value is used.
 BATCH_SIZE=1000, -- The maximum number of records returned by a query operation. If it is not configured, the default value is used.
 SHARDING_SIZE=10000000, -- Sharding size of all the data. If it is not configured, the default value is used.
 RATE_LIMITER ( -- Traffic limit algorithm. If it is not configured, traffic is not limited.
 TYPE( -- Algorithm type. Option: QPS
 NAME='QPS',
 PROPERTIES( -- Algorithm property
 'qps'='500'
 )))
),
WRITE( -- Data writing configuration. If it is not configured, part of the parameters will take effect by default.
 WORKER_THREAD=40, -- The size of the thread pool on which data is written into the target side. If it is not configured, the default value is used.
 BATCH_SIZE=1000, -- The maximum number of records for a batch write operation. If it is not configured, the default value is used.
 RATE_LIMITER ( -- Traffic limit algorithm. If it is not configured, traffic is not limited.
 TYPE( -- Algorithm type. Option: TPS
 NAME='TPS',
 PROPERTIES( -- Algorithm property.
 'tps'='2000'
 )))
),
STREAM_CHANNEL ( -- Data channel. It connects producers and consumers, used for reading and writing procedures. If it is not configured, the MEMORY type is used by default.
TYPE( -- Algorithm type. Option: MEMORY
NAME='MEMORY',
PROPERTIES( -- Algorithm property
'block-queue-size'='10000' -- Property: blocking queue size.
)))
);
```

DistSQL sample: configure READ for traffic limit.

```
ALTER MIGRATION RULE (
READ(
 RATE_LIMITER (TYPE(NAME='QPS',PROPERTIES('qps'='500')))
)
);
```

Configure data reading for traffic limit. Other configurations use default values.

7.3. Restore configuration.

To restore the default configuration, also through the ALTER statement.

```
ALTER MIGRATION RULE (
READ(
 WORKER_THREAD=40,
 BATCH_SIZE=1000,
 SHARDING_SIZE=10000000,
 RATE_LIMITER (TYPE(NAME='QPS',PROPERTIES('qps'='500')))
),
WRITE(
 WORKER_THREAD=40,
 BATCH_SIZE=1000,
 RATE_LIMITER (TYPE(NAME='TPS',PROPERTIES('tps'='2000')))
),
STREAM_CHANNEL (TYPE(NAME='MEMORY',PROPERTIES('block-queue-size'='10000')))
);
```

## User Guide

## MySQL User Guide

### Environment

Supported MySQL versions: 5.1.15 to 8.0.x.

### Authority Required

1. Enable binlog.

MySQL 5.7 my.cnf configuration sample:

```
[mysqld]
server-id=1
log-bin=mysql-bin
binlog-format=row
binlog-row-image=full
max_connections=600
```

Run the following command and check whether binlog is enabled.

```
show variables like '%log_bin%';
show variables like '%binlog%';
```

If the following information is displayed, it means binlog is enabled.

```
+----------------------------------------+----------------------------------------+
| Variable_name                          | Value                                  |
+----------------------------------------+----------------------------------------+
| log_bin                                | ON                                     |
| binlog_format                          | ROW                                    |
| binlog_row_image                       | FULL                                   |
+----------------------------------------+----------------------------------------+
```

2. Grant Replication-related permissions for MySQL account.

Run the following command to check whether the user has migration permission.

```
SHOW GRANTS FOR 'user';
```

Result example:

```
+------------------------------------------------------------------------------+
|Grants for ${username}@${host}                                                |
+------------------------------------------------------------------------------+
|GRANT REPLICATION SLAVE, REPLICATION CLIENT ON *.* TO ${username}@${host}     |
|.......                                                                       |
+------------------------------------------------------------------------------+
```

### Complete Procedure Example

#### Prerequisite

1. Prepare the source database, table, and data in MySQL.

Sample:

```
DROP DATABASE IF EXISTS migration_ds_0;
CREATE DATABASE migration_ds_0 DEFAULT CHARSET utf8;

USE migration_ds_0

CREATE TABLE t_order (order_id INT NOT NULL, user_id INT NOT NULL, status VARCHAR(45) NULL, PRIMARY KEY (order_id));

INSERT INTO t_order (order_id, user_id, status) VALUES (1,2,'ok'),(2,4,'ok'),(3,6,'ok'),(4,1,'ok'),(5,3,'ok'),(6,5,'ok');
```

2. Prepare the target database in MySQL.

Sample:

```
DROP DATABASE IF EXISTS migration_ds_10;
CREATE DATABASE migration_ds_10 DEFAULT CHARSET utf8;

DROP DATABASE IF EXISTS migration_ds_11;
CREATE DATABASE migration_ds_11 DEFAULT CHARSET utf8;

DROP DATABASE IF EXISTS migration_ds_12;
CREATE DATABASE migration_ds_12 DEFAULT CHARSET utf8;
```

### Procedure

1. Create a new logical database in proxy and configure resources and rules.

```
CREATE DATABASE sharding_db;

USE sharding_db

REGISTER STORAGE UNIT ds_2 (
  URL="jdbc:mysql://127.0.0.1:3306/migration_ds_10?serverTimezone=UTC&useSSL=false",
  USER="root",
  PASSWORD="root",
  PROPERTIES("minPoolSize"="1","maxPoolSize"="20","idleTimeout"="60000")
), ds_3 (
  URL="jdbc:mysql://127.0.0.1:3306/migration_ds_11?serverTimezone=UTC&useSSL=false",
  USER="root",
  PASSWORD="root",
  PROPERTIES("minPoolSize"="1","maxPoolSize"="20","idleTimeout"="60000")
), ds_4 (
  URL="jdbc:mysql://127.0.0.1:3306/migration_ds_12?serverTimezone=UTC&useSSL=false",
  USER="root",
  PASSWORD="root",
  PROPERTIES("minPoolSize"="1","maxPoolSize"="20","idleTimeout"="60000")
);

CREATE SHARDING TABLE RULE t_order(
STORAGE_UNITS(ds_2,ds_3,ds_4),
SHARDING_COLUMN=order_id,
TYPE(NAME="hash_mod",PROPERTIES("sharding-count"="6")),
KEY_GENERATE_STRATEGY(COLUMN=order_id,TYPE(NAME="snowflake"))
);
```

If you are migrating to a heterogeneous database, you need to execute the table-creation statements in proxy.

2. Configure the source in proxy.

```
REGISTER MIGRATION SOURCE STORAGE UNIT ds_0 (
  URL="jdbc:mysql://127.0.0.1:3306/migration_ds_0?serverTimezone=UTC&useSSL=false",
  USER="root",
  PASSWORD="root",
  PROPERTIES("minPoolSize"="1","maxPoolSize"="20","idleTimeout"="60000")
);
```

3. Start data migration.

```
MIGRATE TABLE ds_0.t_order INTO t_order;
```

Or you can specify a target logical database.

```
MIGRATE TABLE ds_0.t_order INTO sharding_db.t_order;
```

4. Check the data migration job list.

```
SHOW MIGRATION LIST;
```

Result example:

```
+---------------------------------------+---------+----------------------+--------+---------------------+-----------+
| id                                    | tables  | sharding_total_count | active | create_time         | stop_time |
+---------------------------------------+---------+----------------------+--------+---------------------+-----------+
|j01016e501b498ed1bdb2c373a2e85e2529a6| t_order | 1                    | true   | 2022-08-22 16:37:01 | NULL      |
+---------------------------------------+---------+----------------------+--------+---------------------+-----------+
```

5. View the data migration details.

```
SHOW MIGRATION STATUS 'j01016e501b498ed1bdb2c373a2e85e2529a6';
+------+-------------+--------------------------+--------+------------------------+-----------------------------+--------------------------+---------------+
| item | data_source | status                   | active | processed_records_count | inventory_finished_percentage | incremental_idle_seconds | error_message |
+------+-------------+--------------------------+--------+------------------------+-----------------------------+--------------------------+---------------+
| 0    | ds_0        | EXECUTE_INCREMENTAL_TASK | true   | 6                      | 100                         | 81                       |               |
+------+-------------+--------------------------+--------+------------------------+-----------------------------+--------------------------+---------------+
```

6. Verify data consistency.

```
CHECK MIGRATION 'j01016e501b498ed1bdb2c373a2e85e2529a6' BY TYPE (NAME='CRC32_MATCH');
Query OK, 0 rows affected (0.09 sec)
```

Data consistency check algorithm list:

```
SHOW MIGRATION CHECK ALGORITHMS;
+-------------+--------------------------------------------------------------+----------------------------+
| type        | supported_database_types                                     | description                |
+-------------+--------------------------------------------------------------+----------------------------+
| CRC32_MATCH | MySQL                                                        | Match CRC32 of records.    |
| DATA_MATCH  | SQL92,MySQL,MariaDB,PostgreSQL,openGauss,Oracle,SQLServer,H2 | Match raw data of records. |
+-------------+--------------------------------------------------------------+----------------------------+
```

When data encryption is enabled in the target proxy, DATA_MATCH is required.

DATA_MATCH is required for heterogeneous migration.

7. Commit the job.

```
COMMIT MIGRATION 'j01016e501b498ed1bdb2c373a2e85e2529a6';
```

8. Refresh metadata.

```
REFRESH TABLE METADATA;
```

Please refer to RAL Data Migration for more DistSQL.

## PostgreSQL User Guide

### Environment

Supported PostgreSQL versions can be 9.4 or later.

### Authority Required

1. Enable test_decoding.
2. Modify WAL Configuration.

postgresql.conf configuration sample:

```
wal_level = logical
max_wal_senders = 10
max_replication_slots = 10
wal_sender_timeout = 0
max_connections = 600
```

Please refer to Write Ahead Log and Replication for more details.

3. Configure PostgreSQL and grant Proxy the replication permission.

pg_hba.conf configuration sample:

```
host replication repl_acct 0.0.0.0/0 md5
```

Please refer to The pg_hba.conf File for details.

### Complete Procedure Example

### Prerequisite

1. Prepare the source database, table, and data in PostgreSQL.

```
DROP DATABASE IF EXISTS migration_ds_0;
CREATE DATABASE migration_ds_0;

\c migration_ds_0

CREATE TABLE t_order (order_id INT NOT NULL, user_id INT NOT NULL, status VARCHAR(45) NULL, PRIMARY KEY (order_id));

INSERT INTO t_order (order_id, user_id, status) VALUES (1,2,'ok'),(2,4,'ok'),(3,6,'ok'),(4,1,'ok'),(5,3,'ok'),(6,5,'ok');
```

2. Prepare the target database in PostgreSQL.

```
DROP DATABASE IF EXISTS migration_ds_10;
CREATE DATABASE migration_ds_10;

DROP DATABASE IF EXISTS migration_ds_11;
CREATE DATABASE migration_ds_11;

DROP DATABASE IF EXISTS migration_ds_12;
CREATE DATABASE migration_ds_12;
```

### Procedure

1. Create a new logical database in proxy and configure resources and rules.

```
CREATE DATABASE sharding_db;

\c sharding_db

REGISTER STORAGE UNIT ds_2 (
   URL="jdbc:postgresql://127.0.0.1:5432/migration_ds_10",
   USER="postgres",
   PASSWORD="root",
   PROPERTIES("minPoolSize"="1","maxPoolSize"="20","idleTimeout"="60000")
), ds_3 (
   URL="jdbc:postgresql://127.0.0.1:5432/migration_ds_11",
   USER="postgres",
   PASSWORD="root",
   PROPERTIES("minPoolSize"="1","maxPoolSize"="20","idleTimeout"="60000")
), ds_4 (
   URL="jdbc:postgresql://127.0.0.1:5432/migration_ds_12",
   USER="postgres",
   PASSWORD="root",
   PROPERTIES("minPoolSize"="1","maxPoolSize"="20","idleTimeout"="60000")
);

CREATE SHARDING TABLE RULE t_order(
STORAGE_UNITS(ds_2,ds_3,ds_4),
SHARDING_COLUMN=order_id,
TYPE(NAME="hash_mod",PROPERTIES("sharding-count"="6")),
KEY_GENERATE_STRATEGY(COLUMN=order_id,TYPE(NAME="snowflake"))
);
```

If you are migrating to a heterogeneous database, you need to execute the table-creation statements in proxy.

2. Configure the source resources in proxy.

```
REGISTER MIGRATION SOURCE STORAGE UNIT ds_0 (
    URL="jdbc:postgresql://127.0.0.1:5432/migration_ds_0",
    USER="postgres",
    PASSWORD="root",
    PROPERTIES("minPoolSize"="1","maxPoolSize"="20","idleTimeout"="60000")
);
```

3. Enable Data Migration.

```
MIGRATE TABLE ds_0.t_order INTO t_order;
```

Or you can specify a target logical database.

```
MIGRATE TABLE ds_0.t_order INTO sharding_db.t_order;
```

Or you can specify a source schema.

```
MIGRATE TABLE ds_0.public.t_order INTO sharding_db.t_order;
```

4. Check the data migration job list.

```
SHOW MIGRATION LIST;
```

Result sample:

```
+---------------------------------------+---------+----------------------+--------+---------------------+-----------+
| id                                    | tables  | sharding_total_count | active | create_time         | stop_time |
+---------------------------------------+---------+----------------------+--------+---------------------+-----------+
| j01016e501b498ed1bdb2c373a2e85e2529a6 | t_order | 1                    | true   | 2022-10-13 11:16:01 | NULL      |
+---------------------------------------+---------+----------------------+--------+---------------------+-----------+
```

5. View the data migration details.

```
SHOW MIGRATION STATUS 'j01016e501b498ed1bdb2c373a2e85e2529a6';
+------+-------------+--------------------------+--------+-------------------------+-------------------------------+--------------------------+---------------+
| item | data_source | status                   | active | processed_records_count | inventory_finished_percentage | incremental_idle_seconds | error_message |
+------+-------------+--------------------------+--------+-------------------------+-------------------------------+--------------------------+---------------+
| 0    | ds_0        | EXECUTE_INCREMENTAL_TASK | true   | 6                       | 100                           | 81                       |               |
+------+-------------+--------------------------+--------+-------------------------+-------------------------------+--------------------------+---------------+
```

6. Verify data consistency.

```
CHECK MIGRATION 'j01016e501b498ed1bdb2c373a2e85e2529a6';
Query OK, 0 rows affected (0.09 sec)
```

Query the check progress.

```
SHOW MIGRATION CHECK STATUS 'j01016e501b498ed1bdb2c373a2e85e2529a6';
+---------+--------+---------------------+-------------------+-------------------------+-------------------------+------------------+---------------+
| tables  | result | finished_percentage | remaining_seconds | check_begin_time        | check_end_time          | duration_seconds | error_message |
+---------+--------+---------------------+-------------------+-------------------------+-------------------------+------------------+---------------+
| t_order | true   | 100                 | 0                 | 2022-10-13 11:18:15.171 | 2022-10-13 11:18:15.878 | 0                |               |
+---------+--------+---------------------+-------------------+-------------------------+-------------------------+------------------+---------------+
```

7. Commit the job.

```
COMMIT MIGRATION 'j01016e501b498ed1bdb2c373a2e85e2529a6';
```

8. Refresh metadata.

```
REFRESH TABLE METADATA;
```

Please refer to RAL Data Migration for more DistSQL.

### openGauss User Guide

### Environment

Supported openGauss versions are 2.0.1 to 3.0.0.

### Authority required

1. Modify WAL configuration.

postgresql.conf configuration sample:

```
wal_level = logical
max_wal_senders = 10
max_replication_slots = 10
wal_sender_timeout = 0
max_connections = 600
```

Please refer to Write Ahead Log and Replication for details.

2. Configure openGauss and grant Proxy the replication permission.

pg_hba.conf configuration sample:

```
host replication repl_acct 0.0.0.0/0 md5
```

Please refer to Configuring Client Access Authentication and Example: Logic Replication Code for details.

### Complete Procedure Example

### Prerequisite

1. Prepare the source database, table, and data in openGauss.

```
DROP DATABASE IF EXISTS migration_ds_0;
CREATE DATABASE migration_ds_0;

\c migration_ds_0

CREATE TABLE t_order (order_id INT NOT NULL, user_id INT NOT NULL, status VARCHAR(45) NULL, PRIMARY KEY (order_id));

INSERT INTO t_order (order_id, user_id, status) VALUES (1,2,'ok'),(2,4,'ok'),(3,6,'ok'),(4,1,'ok'),(5,3,'ok'),(6,5,'ok');
```

2. Prepare the target database in openGauss.

```
DROP DATABASE IF EXISTS migration_ds_10;
CREATE DATABASE migration_ds_10;

DROP DATABASE IF EXISTS migration_ds_11;
CREATE DATABASE migration_ds_11;
```

```
DROP DATABASE IF EXISTS migration_ds_12;
CREATE DATABASE migration_ds_12;
```

## Procedure

1. Create a new logical database and configure resources and rules.

```
CREATE DATABASE sharding_db;

\c sharding_db

REGISTER STORAGE UNIT ds_2 (
   URL="jdbc:opengauss://127.0.0.1:5432/migration_ds_10",
   USER="gaussdb",
   PASSWORD="Root@123",
   PROPERTIES("minPoolSize"="1","maxPoolSize"="20","idleTimeout"="60000")
), ds_3 (
   URL="jdbc:opengauss://127.0.0.1:5432/migration_ds_11",
   USER="gaussdb",
   PASSWORD="Root@123",
   PROPERTIES("minPoolSize"="1","maxPoolSize"="20","idleTimeout"="60000")
), ds_4 (
   URL="jdbc:opengauss://127.0.0.1:5432/migration_ds_12",
   USER="gaussdb",
   PASSWORD="Root@123",
   PROPERTIES("minPoolSize"="1","maxPoolSize"="20","idleTimeout"="60000")
);

CREATE SHARDING TABLE RULE t_order(
STORAGE_UNITS(ds_2,ds_3,ds_4),
SHARDING_COLUMN=order_id,
TYPE(NAME="hash_mod",PROPERTIES("sharding-count"="6")),
KEY_GENERATE_STRATEGY(COLUMN=order_id,TYPE(NAME="snowflake"))
);
```

If you are migrating to a heterogeneous database, you need to execute the table-creation statements in proxy.

2. Configure the source resources in proxy.

```
REGISTER MIGRATION SOURCE STORAGE UNIT ds_2 (
   URL="jdbc:opengauss://127.0.0.1:5432/migration_ds_0",
   USER="gaussdb",
   PASSWORD="Root@123",
   PROPERTIES("minPoolSize"="1","maxPoolSize"="20","idleTimeout"="60000")
);
```

3. Enable data migration.

```
MIGRATE TABLE ds_0.t_order INTO t_order;
```

Or you can specify a target logical database.

```
MIGRATE TABLE ds_0.t_order INTO sharding_db.t_order;
```

Or you can specify a source schema.

```
MIGRATE TABLE ds_0.public.t_order INTO sharding_db.t_order;
```

4. Check the data migration job list.

```
SHOW MIGRATION LIST;
```

Result example:

```
+---------------------------------------+---------+----------------------+--------+---------------------+-----------+
| id                                    | tables  | sharding_total_count | active | create_time         | stop_time |
+---------------------------------------+---------+----------------------+--------+---------------------+-----------+
| j01016e501b498ed1bdb2c373a2e85e2529a6 | t_order | 1                    | true   | 2022-10-13 11:16:01 | NULL      |
+---------------------------------------+---------+----------------------+--------+---------------------+-----------+
```

5. View the data migration details.

```
SHOW MIGRATION STATUS 'j01016e501b498ed1bdb2c373a2e85e2529a6';
+------+-------------+--------------------------+--------+-------------------------+-------------------------------+--------------------------+---------------+
| item | data_source | status                   | active | processed_records_count | inventory_finished_percentage | incremental_idle_
seconds | error_message |
+------+-------------+--------------------------+--------+-------------------------+-------------------------------+--------------------------+---------------+
| 0    | ds_0        | EXECUTE_INCREMENTAL_TASK | true   | 6                       | 100                           | 81                       |               |
+------+-------------+--------------------------+--------+-------------------------+-------------------------------+--------------------------+---------------+
```

6. Verify data consistency.

```
CHECK MIGRATION 'j01016e501b498ed1bdb2c373a2e85e2529a6';
Query OK, 0 rows affected (0.09 sec)
```

Query the check progress.

```
SHOW MIGRATION CHECK STATUS 'j01016e501b498ed1bdb2c373a2e85e2529a6';
+---------+--------+---------------------+-------------------+-------------------------+-------------------------+------------------+---------------+
| tables  | result | finished_percentage | remaining_seconds | check_begin_time        | check_end_time          | duration_seconds |
error_message |
+---------+--------+---------------------+-------------------+-------------------------+-------------------------+------------------+---------------+
| t_order | true   | 100                 | 0                 | 2022-10-13 11:18:15.171 | 2022-10-13 11:18:15.878 | 0                |               |
+---------+--------+---------------------+-------------------+-------------------------+-------------------------+------------------+---------------+
```

7. Commit the job.

```
COMMIT MIGRATION 'j01016e501b498ed1bdb2c373a2e85e2529a6';
```

8. Refresh metadata.

```
REFRESH TABLE METADATA;
```

Please refer to RAL Data Migration for more DistSQL.

## 7.2.12 Session Management

SphereEx-DBPlusEngine supports session management. You can view the current session or kill the session through the SQL of the native database.

Currently, this function is only available when the storage node is MySQL. MySQL SHOW PROCESSLIST and KILL commands are supported.

### Usage

### View Session

Different associated databases support different methods for viewing sessions. The show processlist command can be used to view sessions for associated MySQL databases.

SphereEx-DBPlusEngine will automatically generate a unique UUID ID as the ID and store the SQL execution information in each instance.

When this command is executed, SphereEx-DBPlusEngine will collect and synchronize the SQL execution information of each computing node through the governance center, and then summarize and return it to the user.

```
mysql> show processlist;
+----------------------------------+------+-----------+-------------+---------+------+---------------+------------------+
| Id                               | User | Host      | db          | Command | Time | State         | Info             |
+----------------------------------+------+-----------+-------------+---------+------+---------------+------------------+
| 05ede3bd584fd4a429dcaac382be2973 | root | 127.0.0.1 | sharding_db | Execute | 2    | Executing 0/1 | select sleep(10) |
| f9e5c97431567415fe10badc5fa46378 | root | 127.0.0.1 | sharding_db | Sleep   | 690  |               |                  |
+----------------------------------+------+-----------+-------------+---------+------+---------------+------------------+
```

- Output Description

  Here it simulates the output of native MySQL, but the Id field is a special random string.

### Kill Session

The user determines whether the KILL statement needs to be executed according to the results returned by SHOW PROCESSLIST. SphereEx-DBPlusEngine will cancel the SQL in execution according to the ID in the KILL statement.

```
mysql> kill 05ede3bd584fd4a429dcaac382be2973;
Query OK, 0 rows affected (0.04 sec)

mysql> show processlist;
Empty set (0.02 sec)
```

## 7.2.13 Observability

The SphereEx-DBPlusEngine has a built-in observability plugin SphereEx-Agent, which provides users with functions such as extended logs, monitoring indicators, and link tracking.

## Agent Configuration

### Directory structure

Agent related files are located in the sphereex-dbplusengine-proxy/sphereex-agent directory.

```
.
├── conf
│   └── agent.yaml
├── lib
│   ├── ...
├── plugins
│   ├── ...
├── template
│   └── dbplusengine-grafana-template.json
└── tool
  └── ...
```

### Configuration Description

- conf/agent.yaml is used to manage Agent configuration.
- Built-in plugins include Jaeger, OpenTracing, Zipkin, OpenTelemetry, BaseLogging and Prometheus.
- The BaseLogging plug-in is enabled by default, and the slow log query function is enabled.

agent.yaml the default configuration contents are as follows:

```yaml
plugins:
 logging:
  BaseLogging:
   props:
    slow-query-log: true
    long-query-time: 5000
    general-query-log: false
#  metrics:
#    Prometheus:
#      host: "0.0.0.0"
#      port: 9090
#      props:
#        jvm-information-collector-enabled: "true"
#  tracing:
#    Jaeger:
#      host: "localhost"
#      port: 5775
#      props:
#        service-name: "dbplusengine"
#        jaeger-sampler-type: "const"
#        jaeger-sampler-param: "1"
#    Zipkin:
#      host: "localhost"
#      port: 9411
#      props:
#        service-name: "dbplusengine"
#        url-version: "/api/v2/spans"
#        sampler-type: "const"
#        sampler-param: "1"
#    SkyWalking:
#      props:
#        opentracing-tracer-class-name: "org.apache.skywalking.apm.toolkit.opentracing.SkywalkingTracer"
#    OpenTelemetry:
#      props:
```

```
#     otel-resource-attributes: "service.name=dbplusengine"
#     otel-traces-exporter: "zipkin"
```

### Parameter Description

| Name | Description | Value range | Default value |
|---|---|---|---|
| jvm-information-collector-enabled | Start JVM collector | true, false | true |
| service-name | Tracking service name | Custom | dbplusengine |
| jaeger-sampler-type | Jaeger sample rate type | const, probabilistic, ratelimiting, remote | const |
| jaeger-sampler-param | Jaeger sample rate parameter | const :0, 1, probabilistic: 0.0 - 1.0, ratelimiting: > 0, Customize the number of acquisitions per second, remote：need to customize the remote service addres ,JAEGER_SAMPLER_MANAGER_HOST_PORT | 1 (const type) |
| url-version | Zipkin url address | Custom | /api/v2/spans |
| sampler-type | Zipkin sample rate type | const、counting、ratelimiting、boundary | const |
| sampler-param | Zipkin sampling rate parameter | const：0, 1, counting：0.01 - 1.0, ratelimiting: > 0, customize the number of collections per second, boundary: 0.0001 - 1.0 | 1 (const type) |
| otel-resource-attributes | open-telemetry properties | String key value pair (, split) | service.name=dbplusengine-agent |
| otel-traces-exporter | Tracing expoter | zipkin, jaeger | zipkin |
| otel-traces-sampler | Open-telemetry sample rate type | always_on, always_off, traceidratio | always_on |
| otel-traces-sampler-arg | Open-telemetry sample rate parameter | traceidratio：0.0 - 1.0 | 1.0 |

### Used in DBPlusEngine-Proxy

In the sphereex-dbplusengine-proxy/bin/ directory, two startup scripts are provided for users: - start.sh - start-with-agent.sh

When the user needs the Agent function, execute start-with-agent.sh to start DBPlusEngine-Proxy.

```
bin/start-with-agent.sh
```

After successful startup, you can find the loading information of the plugin in the DBPlusEngine-Proxy log.

If the ‘metrics’ or ‘tracing’ configuration is enabled, relevant data can be viewed through the configured monitoring address after accessing the proxy.

## 7.3 Error Code

This chapter lists the error codes of DBPlusEngine. They include SQL error codes and server error codes.

**All contents of this chapter are a draft, the error codes may be adjusted.**

### 7.3.1 SQL Error Code

SQL error codes provide by standard SQL State, Vendor Code and Reason, which return to the client when SQL execute error.

**The error codes are draft, they may be adjusted.**

#### Kernel Exception

#### Meta data

| SQL State | Vendor Code | Reason |
|---|---|---|
| 42000 | 10000 | Resource does not exist. |
| 08000 | 10001 | The URL %s is not recognized, please refer to the pattern %s. |
| 42000 | 10002 | Cannot support 3-tier structure for actual data node %s with JDBC %s. |
| HY004 | 10003 | Invalid format for actual data node %s. |
| 42000 | 10004 | Unsupported SQL node conversion for SQL statement %s. |
| 42000 | 10010 | Rule does not exist. |
| 42S02 | 10020 | Schema %s does not exist. |
| 42S02 | 10021 | Single table %s does not exist. |
| HY000 | 10022 | Cannot load table with database name %s and data source name %s. |
| 0A000 | 10030 | Cannot drop schema %s because of contains tables. |

#### Data

| SQL State | Vendor Code | Reason |
|---|---|---|
| HY004 | 11000 | Invalid value %s. |
| HY004 | 11001 | Unsupported conversion data type %s for value %s. |
| HY004 | 11010 | Unsupported conversion stream charset %s. |

#### Syntax

| SQL State | Vendor Code | Reason |
|---|---|---|
| 42000 | 12000 | You have an error in your SQL syntax: %s |
| 42000 | 12001 | Cannot accept SQL type %s. |
| 42000 | 12002 | SQL String can not be NULL or empty. |
| 42000 | 12010 | Cannot support variable %s. |
| 42S02 | 12011 | Cannot find column label %s. |
| HV008 | 12020 | Column index %d is out of range. |
| 0A000 | 12100 | DROP TABLE …CASCADE is not supported. |

### Connection

| SQL State | Vendor Code | Reason |
|---|---|---|
| 08000 | 13000 | Cannot register driver, reason is: %s |
| 01000 | 13010 | Circuit break open, the request has been ignored. |
| 08000 | 13020 | Cannot get %d connections one time, partition succeed c onnection(%d) have released. Please consider increasing the ` maxPoolSize` of the data sources or decreasing the max-co nnections-siz e-per-query in properties. |
| 08000 | 13030 | Connection has been closed. |
| 08000 | 13031 | Result set has been closed. |
| HY000 | 13090 | Load datetime from database failed, reason: %s |

### Transaction

| SQL State | Vendor Code | Reason |
|---|---|---|
| 25000 | 14000 | Switch transaction type failed, please terminate the current transaction. |
| 25000 | 14100 | JDBC does not support operations across multiple logical databases in transaction. |
| 25000 | 14200 | Cannot start new XA transaction in a active transaction. |
| 25000 | 14201 | Failed to create %s XA data source. |

### Lock

| SQL State | Vendor Code | Reason |
|---|---|---|
| HY000 | 15000 | The table %s of schema %s is locked. |
| HY000 | 15001 | The table %s of schema %s lock wait timeout of %s milliseconds exceeded. |

### Audit

| SQL State | Vendor Code | Reason |
|---|---|---|
| 44000 | 16000 | SQL check failed, error message: %s |

### Cluster

| SQL State | Vendor Code | Reason |
|---|---|---|
| HY000 | 17000 | Work ID assigned failed, which can not exceed 1024. |
| HY000 | 17001 | Cannot find %s file for datetime initialize. |
| HY000 | 17002 | File access failed, reason is: %s |
| HY000 | 17010 | Cluster persist repository error, reason is: %s |

## Migration

| SQLState | Vendor Code | Reason |
|---|---|---|
| 44000 | 18001 | Created rule already existed. |
| 44000 | 18002 | Altered rule does not exist. |
| HY000 | 18020 | Failed to get DDL for table %s. |
| 42S01 | 18030 | Duplicate resource names %s. |
| 42S02 | 18031 | Resource names %s do not exist. |
| 0A000 | 18032 | Unsupported data type %s of unique key for pipeline job. |
| HY000 | 18050 | Before data record is %s, after data record is %s. |
| 08000 | 18051 | Data check table %s failed. |
| 0A000 | 18052 | Unsupported pipeline database type %s. |
| 0A000 | 18053 | Unsupported CRC32 data consistency calculate algorithm with database type %s. |
| HY000 | 18080 | Cannot find pipeline job %s. |
| HY000 | 18081 | Job has already started. |
| HY000 | 18082 | Sharding count of job %s is 0. |
| HY000 | 18083 | Cannot split by range for table %s, reason is: %s |
| HY000 | 18084 | Cannot split by unique key %s for table %s, reason is: %s |
| HY000 | 18085 | Target table %s is not empty. |
| 01007 | 18086 | Source data source lacks %s privilege(s). |
| HY000 | 18087 | Source data source required %s = %s, now is %s. |
| HY000 | 18088 | User %s does exist. |
| 08000 | 18089 | Check privileges failed on source data source, reason is: %s |
| 08000 | 18090 | Data sources can not connect, reason is: %s |
| HY000 | 18091 | Importer job write data failed. |
| 08000 | 18092 | Get binlog position failed by job %s, reason is: %s |
| HY000 | 18093 | Cannot poll event because of binlog sync channel already closed. |
| HY000 | 18094 | Task %s execute failed. |
| HY000 | 18095 | Job has already finished, please run CHECK MIGRATION %s to start a new data consistency check job. |
| HY000 | 18096 | Incomplete consistency check job %s exists. |

## DistSQL

| SQL State | Vendor Code | Reason |
|---|---|---|
| 44000 | 19000 | Cannot process invalid resources, error message is: %s |
| 44000 | 19001 | Resources %s do not exist in database %s. |
| 44000 | 19002 | There is no resource in the database %s. |
| 44000 | 19003 | Resource %s is still used by %s. |
| 44000 | 19004 | Duplicate resource names %s. |
| 44000 | 19100 | Invalid %s rule %s, error message is: %s |
| 44000 | 19101 | %s rules %s do not exist in database %s. |
| 44000 | 19102 | %s rules %s in database %s are still in used. |
| 44000 | 19103 | %s rule %s has been enabled in database %s. |
| 44000 | 19104 | %s rule %s has been disabled in database %s. |
| 44000 | 19105 | Duplicate %s rule names %s in database %s. |
| 44000 | 19150 | Invalid %s algorithm(s) %s. |
| 44000 | 19151 | %s algorithm(s) %s do not exist in database %s. |
| 44000 | 19152 | %s algorithms %s in database %s are still in used. |
| 44000 | 19153 | Duplicate %s algorithms %s in database %s. |

## Feature Exception

### Data Sharding

| SQL State | Vendor Code | Reason |
|---|---|---|
| 44000 | 20000 | Cannot find table rule with logic tables %s. |
| 44000 | 20001 | Cannot get uniformed table structure for logic table %s, it has different meta data of actual tab |
| 42S02 | 20002 | Cannot find data source in sharding rule, invalid actual data node %s. |
| 44000 | 20003 | Data nodes must be configured for sharding table %s. |
| 44000 | 20004 | Actual table %s.%s is not in table rule c onfiguration. |
| 44000 | 20005 | Cannot find binding actual table, data source is %s, logic table is %s, other actual table is %s. |
| 44000 | 20006 | Actual tables %s are in use. |
| 42S01 | 20007 | Index %s already exists. |
| 42S02 | 20008 | Index %s does not exist. |
| 42S01 | 20009 | View name has to bind to %s tables. |
| 44000 | 20020 | Sharding value can’ t be null in insert statement. |
| HY004 | 20021 | Found different types for sharding value %s. |
| HY004 | 20022 | Invalid %s, datetime pattern should be %s, value is %s. |
| 0A000 | 20040 | Cannot support operation %s with sharding table %s. |
| 44000 | 20041 | Cannot update sharding value for table %s. |
| 0A000 | 20042 | The CREATE VIEW statement contains unsupported query statement. |
| 44000 | 20043 | PREPARE statement cannot support sharding tables route to same data sources. |
| 44000 | 20044 | The table inserted and the table selected must be the same or bind tables. |
| 0A000 | 20045 | Cannot support DML operation with multiple tables %s. |
| 42000 | 20046 | %s …LIMIT cannot support route to multiple data nodes. |
| 44000 | 20047 | Cannot find actual data source intersection for logic tables %s. |
| 42000 | 20048 | INSERT INTO …SELECT can not support applying key generator with absent generate key colu |
| 0A000 | 20049 | Alter view rename .. to .. statement should have same config for %s and %s. |
| HY000 | 20060 | %s %s can not route correctly for %s %s. |
| 42S02 | 20061 | Cannot get route result, please check your sharding rule c onfiguration. |
| 34000 | 20062 | Cannot get cursor name from fetch statement. |
| HY000 | 20080 | Sharding algorithm class %s should be implement %s. |
| HY000 | 20081 | Routed target %s does not exist, available targets are %s. |
| 44000 | 20082 | Inline sharding algorithms expression %s and sharding column %s do not match. |
| 44000 | 20090 | Cannot find strategy for generate keys with table %s. |
| HY000 | 20099 | Sharding plugin error, reason is: %s |

### Readwrite Splitting

| SQL State | Vendor Code | Reason |
|---|---|---|
| HY004 | 20280 | Invalid read database weight %s. |

### Database HA

| SQL State | Vendor Code | Reason |
|---|---|---|
| HY000 | 20380 | MGR plugin is not active in database %s. |
| 44000 | 20381 | MGR is not in single primary mode in database %s. |
| 44000 | 20382 | %s is not in MGR replication group member in database %s. |
| 44000 | 20383 | Group name in MGR is not same with configured one %s in database %s. |

## SQL Dialect Translator

| SQL State | Vendor Code | Reason |
|---|---|---|
| 42000 | 20440 | Cannot support database %s in SQL translation. |
| 42000 | 20441 | Translation error, SQL is: %s |

## Traffic Management

| SQL State | Vendor Code | Reason |
|---|---|---|
| 42S02 | 20500 | Cannot get traffic execution unit. |

## Data Encrypt

| SQL State | Vendor Code | Reason |
|---|---|---|
| 44000 | 20700 | Cannot find logic encrypt column by %s. |
| 44000 | 20701 | Failure to find encrypt column %s from table %s. |
| 44000 | 20702 | Altered column %s must use same encrypt algorithm with previous column %s in table %s. |
| 42000 | 20740 | Insert value of index %s can not support for encrypt. |
| 0A000 | 20741 | The SQL clause %s is unsupported in encrypt rule. |
| HY004 | 20780 | Encrypt algorithm %s i nitialization failed, reason is: %s |

## Shadow Database

| SQL State | Vendor Code | Reason |
|---|---|---|
| HY004 | 20820 | Shadow column %s of table %s does not support %s type. |
| 42000 | 20840 | Insert value of index %s can not support for shadow. |

## Other Exception

| SQL State | Vendor Code | Reason |
|---|---|---|
| HY004 | 30000 | Unknown exception: %s |
| 0A000 | 30001 | Unsupported SQL operation: %s |
| 0A000 | 30002 | Database protocol exception: %s |
| 0A000 | 30003 | Unsupported command: %s |

### 7.3.2 Server Error Code

Unique codes provided when server exception occur, which printed by Proxy backend or Driver startup logs.

| Error Code | Reason |
|---|---|
| SPI-00001 | No implementation class load from SPI %s with type %s. |
| DATA-SOURCE-00001 | Data source unavailable. |
| PROPS-00001 | Value %s of %s cannot convert to type %s. |
| PROXY-00001 | Load database server info failed. |
| SPRING-00001 | Cannot find JNDI data source. |
| SPRING-SHARDING-00001 | Cannot support type %s. |

# 8 Dev Manual

DB Plus Engine provides dozens of SPI based extensions. It is very convenient for developers to customize the features.

This chapter lists all SPI extensions of DB Plus Engine. If there is no special requirement, users can use the built-in implementation provided by DB Plus Engine; advanced users can refer to the interfaces for customized implementation.

- Traffic Dual Routing (Commercial Edition)

## 8.1 Mode

### 8.1.1 ClusterPersistRepository

| SPI Name | Description |
|---|---|
| ClusterPersistRepository | Registry center repository |

| Implementation Class | Description |
|---|---|
| ZooKeeper | ZooKeeper registry center repository |
| Etcd | Etcd registry center repository |
| SphereEx:MATE | MATE registry center repository |

### 8.1.3 GovernanceWatcher

| SPI Name | Description |
|---|---|
| GovernanceWatcher | Governance watcher |

| Implementation Class | Description |
|---|---|
| StorageNodeStateChangedWatcher | Storage node changed watcher |
| ComputeNodeStateChangedWatcher | Compute node changed watcher |
| PropertiesChangedWatcher | Properties changed watcher |
| PrivilegeNodeChangedWatcher | Privilege changed watcher |
| GlobalRuleChangedWatcher | Global rule changed watcher |
| MetaDataChangedWatcher | Meta data changed watcher |

## 8.2 Configuration

### 8.2.1 RuleBuilder

| SPI Name | Description |
|---|---|
| RuleBuilder | Used to convert user configurations to rule objects |

| Implementation Class | Description |
|---|---|
| AlgorithmPro videdReadwriteSpl ittingRuleBuilder | Used to convert algorithm-based read-write separation user configuration into read-write separation rule objects |
| AlgorithmPr ovidedDatabaseDis coveryRuleBuilder | Used to convert algorithm-based database discovery user configuration into database discovery rule objects |
| Al gorithmProvidedSh ardingRuleBuilder | Used to convert algorithm-based sharding user configuration into sharding rule objects |
| A lgorithmProvidedE ncryptRuleBuilder | Used to convert algorithm-based encryption user configuration into encryption rule objects |
| AlgorithmProvided ShadowRuleBuilder | Used to convert algorithm-based shadow database user configuration into shadow database rule objects |
| ReadwriteSpl ittingRuleBuilder | Used to convert read-write separation user configuration into read-write separation rule objects |
| DatabaseDis coveryRuleBuilder | Used to convert database discovery user configuration into database discovery rule objects |
| Singl eTableRuleBuilder | Used to convert single-table user configuration into a single-table rule objects |
| Aut horityRuleBuilder | Used to convert permission user configuration into permission rule objects |
| Sh ardingRuleBuilder | Used to convert sharding user configuration into sharding rule objects |
| E ncryptRuleBuilder | Used to convert encrypted user configuration into encryption rule objects |
| ShadowRuleBuilder | Used to convert shadow database user configuration into shadow database rule objects |
| Trans actionRuleBuilder | Used to convert transaction user configuration into transaction rule objects |
| SQL ParserRuleBuilder | Used to convert SQL parser user configuration into SQL parser rule objects |

### 8.2.2 YamlRuleConfigurationSwapper

| SPI Name | Description |
|---|---|
| YamlRuleConfigurationSwapper | Used to convert YAML configuration to standard user configuration |

| Implementation Class | Description |
|---|---|
| ReadwriteSplittingRuleAlgorithmProviderConfigurationYamlSwapper | Used to convert algorithm-based read-write separation configuration into read-write separation standard configuration |
| DatabaseDiscoveryRuleAlgorithmProviderConfigurationYamlSwapper | Used to convert algorithm-based database discovery configuration into database discovery standard configuration |
| ShardingRuleAlgorithmProviderConfigurationYamlSwapper | Used to convert algorithm-based sharding configuration into sharding standard configuration |
| EncryptRuleAlgorithmProviderConfigurationYamlSwapper | Used to convert algorithm-based encryption configuration into encryption standard configuration |
| ShadowRuleAlgorithmProviderConfigurationYamlSwapper | Used to convert algorithm-based shadow database configuration into shadow database standard configuration |
| ReadwriteSplittingRuleConfigurationYamlSwapper | Used to convert the YAML configuration of read-write separation into the standard configuration of read-write separation |
| DatabaseDiscoveryRuleConfigurationYamlSwapper | Used to convert the YAML configuration of database discovery into the standard configuration of database discovery |
| AuthorityRuleConfigurationYamlSwapper | Used to convert the YAML configuration of permission rules into standard configuration of permission rules |
| ShardingRuleConfigurationYamlSwapper | Used to convert the YAML configuration of the shard into the standard configuration of the shard |
| EncryptRuleConfigurationYamlSwapper | Used to convert encrypted YAML configuration into encrypted standard configuration |
| ShadowRuleConfigurationYamlSwapper | Used to convert the YAML configuration of the shadow database into the standard configuration of the shadow database |
| TransactionRuleConfigurationYamlSwapper | Used to convert the YAML configuration of the transaction into the standard configuration of the transaction |
| SingleTableRuleConfigurationYamlSwapper | Used to convert the YAML configuration of the single table into the standard configuration of the single table |
| SQLParserRuleConfigurationYamlSwapper | Used to convert the YAML configuration of the SQL parser into the standard configuration of the SQL parser |

### 8.2.3 ShardingSphereYamlConstruct

| SPI Name | Description |
|---|---|
| ShardingSphereYamlConstruct | Used to convert customized objects and YAML to each other |

| Implementation Class | Description |
|---|---|
| NoneShardingStrategyConfigurationYamlConstruct | Used to convert non sharding strategy and YAML to each other |

## 8.3 Kernel

### 8.3.1 SQLRouter

| SPI Name | Description |
|---|---|
| SQLRouter | Used to process routing results |

| Implementation Class | Description |
|---|---|
| Re adwriteSplittingSQLRouter | Used to process read-write separation routing results |
| D atabaseDiscoverySQLRouter | Used to process database discovery routing results |
| SingleTableSQLRouter | Used to process single-table routing results |
| ShardingSQLRouter | Used to process sharding routing results |
| ShadowSQLRouter | Used to process shadow database routing results |

### 8.3.2 SQLRewriteContextDecorator

| SPI Name | Description |
|---|---|
| SQLRewriteContextDecorator | Used to process SQL rewrite results |

| SPI Name | Description |
|---|---|
| Shardin gSQLRewriteContextDecorator | Used to process sharding SQL rewrite results |
| Encryp tSQLRewriteContextDecorator | Used to process encryption SQL rewrite results |

### 8.3.3 SQLExecutionHook

| SPI Name | Description |
|---|---|
| SQLExecutionHook | Hook of SQL execution |

| Implementation Class | Description |
|---|---|
| TransactionalSQLExecutionHook | Transaction hook of SQL execution |

### 8.3.4 ResultProcessEngine

| SPI Name | Description |
|---|---|
| ResultProcessEngine | Used by merge engine to process result set |

| Implementation Class | Description |
|---|---|
| Shard ingResultMergerEngine | Used by merge engine to process sharding result set |
| Encrypt ResultDecoratorEngine | Used by merge engine to process encryption result set |

### 8.3.5 StoragePrivilegeHandler

| SPI Name | Description |
|---|---|
| StoragePrivilegeHandler | Use SQL dialect to process privilege metadata |

| Implementation Class | Description |
|---|---|
| Postg reSQLPrivilegeHandler | Use PostgreSQL dialect to process privilege metadata |
| SQLS erverPrivilegeHandler | Use SQLServer dialect to process privilege metadata |
| O raclePrivilegeHandler | Use Oracle dialect to process privilege metadata |
| MySQLPrivilegeHandler | Use MySQL dialect to process privilege metadata |

## 8.4 DataSource

### 8.4.1 DatabaseType

| SPI Name | Description |
|---|---|
| DatabaseType | Supported database type |

| Implementation Class | Description |
|---|---|
| SQL92DatabaseType | SQL92 database type |
| MySQLDatabaseType | MySQL database |
| MariaDBDatabaseType | MariaDB database |
| PostgreSQLDatabaseType | PostgreSQL database |
| OracleDatabaseType | Oracle database |
| SQLServerDatabaseType | SQLServer database |
| H2DatabaseType | H2 database |
| OpenGaussDatabaseType | OpenGauss database |

### 8.4.2 DialectTableMetaDataLoader

| SPI Name | Description |
|---|---|
| DialectTableMetaDataLoader | Use SQL dialect to load meta data rapidly |

| Implementation Class | Description |
|---|---|
| MySQLTableMetaDataLoader | Use MySQL dialect to load meta data |
| OracleTableMetaDataLoader | Use Oracle dialect to load meta data |
| PostgreSQLTableMetaDataLoader | Use PostgreSQL dialect to load meta data |
| SQLServerTableMetaDataLoader | Use SQLServer dialect to load meta data |
| H2TableMetaDataLoader | Use H2 dialect to load meta data |
| OpenGaussTableMetaDataLoader | Use OpenGauss dialect to load meta data |

### 8.4.3 DataSourcePoolMetaData

| SPI Name | Description |
|---|---|
| DataSourcePoolMetaData | Data source pool meta data |

| Implementation Class | Description |
|---|---|
| DBCPDataSourcePoolMetaData | DBCP data source pool meta data |
| HikariDataSourcePoolMetaData | Hikari data source pool meta data |
| TomcatDBCPDataSourcePoolMetaData | Tomcat DBCP data source pool meta data |

### 8.4.4 DataSourcePoolDestroyer

| SPI Name | Description |
|---|---|
| DataSourcePoolDestroyer | Data source pool destroyer |

| Implementation Class | Description |
|---|---|
| DefaultDataSourcePoolDestroyer | Default data source pool destroyer |
| HikariDataSourcePoolDestroyer | Hikari data source pool destroyer |

## 8.5 SQL Parser

### 8.5.1 DatabaseTypedSQLParserFacade

| SPI Name | Description |
|---|---|
| DatabaseTypedSQLParserFacade | SQL parser facade for lexer and parser |

| Implementation Class | Description |
|---|---|
| MySQLParserFacade | SQL parser facade for MySQL |
| PostgreSQLParserFacade | SQL parser facade for PostgreSQL |
| SQLServerParserFacade | SQL parser facade for SQLServer |
| OracleParserFacade | SQL parser facade for Oracle |
| SQL92ParserFacade | SQL parser facade for SQL92 |
| OpenGaussParserFacade | SQL parser facade for openGauss |

### 8.5.2 SQLVisitorFacade

| SPI Name | Description |
|---|---|
| SQLVisitorFacade | SQL AST visitor facade |

| Implementation Class | Description |
|---|---|
| MySQLS tatementSQLVisitorFacade | SQL visitor of statement extracted facade for MySQL |
| PostgreSQLS tatementSQLVisitorFacade | SQL visitor of statement extracted facade for PostgreSQL |
| SQLServerS tatementSQLVisitorFacade | SQL visitor of statement extracted facade for SQLServer |
| OracleS tatementSQLVisitorFacade | SQL visitor of statement extracted facade for Oracle |
| SQL92S tatementSQLVisitorFacade | SQL visitor of statement extracted facade for SQL92 |

## 8.6 Proxy

### 8.6.1 DatabaseProtocolFrontendEngine

| SPI Name | Description |
|---|---|
| DatabaseProtocolFrontendEngine | Regulate parse and adapter protocol of database access for ShardingSphere-Proxy |

| Implementation Class | Description |
|---|---|
| MySQLFrontendEngine | Base on MySQL database protocol |
| PostgreSQLFrontendEngine | Base on PostgreSQL database protocol |
| OpenGaussFrontendEngine | Base on openGauss database protocol |

### 8.6.2 JDBCDriverURLRecognizer

| SPI Name | Description |
|---|---|
| JDBCDriverURLRecognizer | Use JDBC driver to execute SQL |

| Implementation Class | Description |
|---|---|
| MySQLRecognizer | Use MySQL JDBC driver to execute SQL |
| PostgreSQLRecognizer | Use PostgreSQL JDBC driver to execute SQL |
| OracleRecognizer | Use Oracle JDBC driver to execute SQL |
| SQLServerRecognizer | Use SQLServer JDBC driver to execute SQL |
| H2Recognizer | Use H2 JDBC driver to execute SQL |
| P6SpyDriverRecognizer | Use P6Spy JDBC driver to execute SQL |
| OpenGaussRecognizer | Use openGauss JDBC driver to execute SQL |

### 8.6.3 AuthorityProviderAlgorithm

| SPI Name | Description |
|---|---|
| AuthorityProviderAlgorithm | User authority loading logic |

| Implementation Class | Type | Description |
|---|---|---|
| AllPermittedPrivilegesProviderAlgorithm | A LL_PERMITTED | All privileges granted to user by default (No authentication). Will not interact with the actual database. |
| DatabasePermittedPrivilegesP roviderAlgorithm | DATABASE_PERMITTED | Permissions configured through the attribute user-database-mappings. |

## 8.7 Data Sharding

### 8.7.1 ShardingAlgorithm

| SPI Name | Description |
|---|---|
| ShardingAlgorithm | Sharding algorithm |

| Implementation Class | Description |
|---|---|
| Boundar yBasedRangeShardingAlgorithm | Boundary based range sharding algorithm |
| Volum eBasedRangeShardingAlgorithm | Volume based range sharding algorithm |
| Co mplexInlineShardingAlgorithm | Complex inline sharding algorithm |
| A utoIntervalShardingAlgorithm | Mutable interval sharding algorithm |
| ClassBasedShardingAlgorithm | Class based sharding algorithm |
| HintInlineShardingAlgorithm | Hint inline sharding algorithm |
| IntervalShardingAlgorithm | Fixed interval sharding algorithm |
| HashModShardingAlgorithm | Hash modulo sharding algorithm |
| InlineShardingAlgorithm | Inline sharding algorithm |
| ModShardingAlgorithm | Modulo sharding algorithm |

### 8.7.2 KeyGenerateAlgorithm

| SPI Name | Description |
|---|---|
| KeyGenerateAlgorithm | Key generate algorithm |

| Implementation Class | Description |
|---|---|
| SnowflakeKeyGenerateAlgorithm | Snowflake key generate algorithm |
| UUIDKeyGenerateAlgorithm | UUID key generate algorithm |

### 8.7.3 DatetimeService

| SPI Name | Description |
|---|---|
| DatetimeService | Use current time for routing |

| Implementation Class | Description |
|---|---|
| DatabaseDa tetimeServiceDelegate | Get the current time from the database for routing |
| SystemDatetimeService | Get the current time from the application system for routing |

### 8.7.4 DatabaseSQLEntry

| SPI Name | Description |
|---|---|
| DatabaseSQLEntry | Database dialect for get current time |

| Implementation Class | Description |
|---|---|
| MySQLDatabaseSQLEntry | MySQL dialect for get current time |
| PostgreSQLDatabaseSQLEntry | PostgreSQL dialect for get current time |
| OracleDatabaseSQLEntry | Oracle dialect for get current time |
| SQLServerDatabaseSQLEntry | SQLServer dialect for get current time |

# 8.8 Read/write splitting

## 8.8.1 ReadwriteSplittingType

| SPI Name | Description |
|---|---|
| ReadwriteSplittingType | Readwrite-splitting type |

| Implementation Class | Description |
|---|---|
| StaticReadwriteSplittingType | Static readwrite-splitting type |
| DynamicReadwriteSplittingType | Dynamic readwrite-splitting type |

## 8.8.2 ReadQueryLoadBalanceAlgorithm

| SPI Name | Description |
|---|---|
| ReadQueryLoadBalanceAlgorithm | Load balance algorithm of replica databases |

| Implementation Class | Description |
|---|---|
| RoundRobinRe plicaLoadBalanceAlgorithm | Round robin load balance algorithm of replica databases |
| RandomRe plicaLoadBalanceAlgorithm | Random load balance algorithm of replica databases |
| WeightRe plicaLoadBalanceAlgorithm | Weight load balance algorithm of replica databases |
| DelayRe plicaLoadBalanceAlgorithm | Delay load balance algorithm of replica databases |

# 8.9 HA

## 8.9.1 DatabaseDiscoveryType

| SPI Name | Description |
|---|---|
| DatabaseDiscoveryType | Database discovery type |

| Implementation Class | Description |
|---|---|
| MGRDatabaseDiscoveryType | Database discovery of MySQL’s MGR |
| OpenGaussDatabaseDiscoveryType | Database discovery of openGauss |

## 8.10 Distributed Transaction

### 8.10.1 ShardingSphereTransactionManager

| SPI Name | Description |
|---|---|
| ShardingSphereTransactionManager | Distributed transaction manager |

| Implementation Class | Description |
|---|---|
| X AShardingSphereTransactionManager | XA distributed transaction manager |
| SeataA TShardingSphereTransactionManager | Seata distributed transaction manager |

### 8.10.2 XATransactionManagerProvider

| SPI Name | Description |
|---|---|
| XATransactionManagerProvider | XA distributed transaction manager |

| Implementation Class | Description |
|---|---|
| Atomikos TransactionManagerProvider | XA distributed transaction manager based on Atomikos |
| NarayanaXA TransactionManagerProvider | XA distributed transaction manager based on Narayana |
| BitronixXA TransactionManagerProvider | XA distributed transaction manager based on Bitronix |

### 8.10.3 XADataSourceDefinition

| SPI Name | Description |
|---|---|
| XADataSourceDefinition | Auto convert Non XA data source to XA data source |

| Implementation Class | Description |
|---|---|
| MySQLXAD ataSourceDefinition | Auto convert Non XA MySQL data source to XA MySQL data source |
| MariaDBXAD ataSourceDefinition | Auto convert Non XA MariaDB data source to XA MariaDB data source |
| PostgreSQLXAD ataSourceDefinition | Auto convert Non XA PostgreSQL data source to XA PostgreSQL data source |
| OracleXAD ataSourceDefinition | Auto convert Non XA Oracle data source to XA Oracle data source |
| SQLServerXAD ataSourceDefinition | Auto convert Non XA SQLServer data source to XA SQLServer data source |
| H2XAD ataSourceDefinition | Auto convert Non XA H2 data source to XA H2 data source |

### 8.10.4 DataSourcePropertyProvider

| SPI Name | Description |
|---|---|
| DataS ourcePropertyProvider | Used to get standard properties of data source pool |

| Implementation Class | Description |
|---|---|
| HikariCPPropertyProvider | Used to get standard properties of HikariCP |

## 8.11 Scaling

### 8.11.1 ScalingEntry

| SPI Name | Description |
|---|---|
| ScalingEntry | Entry of scaling |

| Implementation Class | Description |
|---|---|
| MySQLScalingEntry | MySQL entry of scaling |
| PostgreSQLScalingEntry | PostgreSQL entry of scaling |
| OpenGaussScalingEntry | openGauss entry of scaling |

### 8.11.2 JobCompletionDetectAlgorithm

| SPI Name | Description |
|---|---|
| JobCompletionDetectAlgorithm | Job completion check algorithm |

| Implementation Class | Description |
|---|---|
| IdleRuleAl teredJobCompletionDetectAlgorithm | Incremental task idle time based algorithm |

### 8.11.3 DataConsistencyCheckAlgorithm

| SPI Name | Description |
|---|---|
| DataConsistencyCheckAlgorithm | Data consistency check algorithm on source and target database cluster |

| Implementation Class | Description |
|---|---|
| DataMatchDataC onsistencyCheckAlgorithm | Records content match implementation. Type name: DATA_MATCH. |
| CRC32MatchDataC onsistencyCheckAlgorithm | Records CRC32 match implementation. Type name: CRC32_MATCH. |

### 8.11.4 SingleTableDataCalculator

| SPI Name | Description |
|---|---|
| S ingleTableDataCalculator | Single table data calculator for data consistency check |

| Implementation Class | Description |
|---|---|
| DataMatchS ingleTableDataCalculator | Single table data calculator for DATA_MATCH data consistency check |
| CRC32MatchMySQLS ingleTableDataCalculator | Single table data calculator for CRC32_MATCH data consistency check |

## 8.12 SQL Checker

### 8.12.1 SQLChecker

| SPI Name | Description |
|---|---|
| SQLChecker | SQL checker |

| Implementation Class | Description |
|---|---|
| AuthorityChecker | Authority checker |

## 8.13 Encryption

### 8.13.1 EncryptAlgorithm

| SPI Name | Description |
|---|---|
| EncryptAlgorithm | Data encrypt algorithm |

| Implementation Class | Description |
|---|---|
| MD5EncryptAlgorithm | MD5 data encrypt algorithm |
| AESEncryptAlgorithm | AES data encrypt algorithm |
| RC4EncryptAlgorithm | RC4 data encrypt algorithm |
| SM3EncryptAlgorithm | SM3 data encrypt algorithm |
| SM4EncryptAlgorithm | SM4 data encrypt algorithm |

### 8.13.2 QueryAssistedEncryptAlgorithm

| SPI Name | Description |
|---|---|
| QueryAss istedEncryptAlgorithm | Data encrypt algorithm which include query assisted column |

| Implementation Class | Description |
|---|---|
| None | |

## 8.14 Shadow DB

### 8.14.1 ShadowAlgorithm

| SPI Name | Description |
|---|---|
| ShadowAlgorithm | shadow routing algorithm |

| Implementation Class | Description |
|---|---|
| ColumnValueMatchShadowAlgorithm | Column value match shadow algorithm |
| ColumnRegexMatchShadowAlgorithm | Column regex match shadow algorithm |
| SimpleHintShadowAlgorithm | Simple hint shadow algorithm |

# 8.15 Observability

## 8.15.1 PluginDefinitionService

| SPI Name | Description |
|---|---|
| PluginDefinitionService | Agent plugin definition |

| Implementation Class | Description |
|---|---|
| PrometheusPluginDefinitionService | Prometheus plugin |
| BaseLoggingPluginDefinitionService | Logging plugin |
| JaegerPluginDefinitionService | Jaeger plugin |
| OpenTelemetryTracingPluginDefinitionService | OpenTelemetryTracing plugin |
| OpenTracingPluginDefinitionService | OpenTracing plugin |
| ZipkinPluginDefinitionService | Zipkin plugin |

## 8.15.2 PluginBootService

| SPI Name | Description |
|---|---|
| PluginBootService | Plugin startup service definition |

| Implementation Class | Description |
|---|---|
| PrometheusPluginBootService | Prometheus plugin startup class |
| BaseLoggingPluginBootService | Logging plugin startup class |
| JaegerTracingPluginBootService | Jaeger plugin startup class |
| OpenTelemetryTracingPluginBootService | OpenTelemetryTracing plugin startup class |
| OpenTracingPluginBootService | OpenTracing plugin startup class |
| ZipkinTracingPluginBootService | Zipkin plugin startup class |

## 8.15.3 Proxy Agent Monitoring Metrics

| Metric | Description |
|---|---|
| proxy_request_total | total number of requests |
| p roxy_connection_total | total number of connections |
| proxy_e xecute_latency_millis | request duration (MS) |
| prox y_execute_error_total | Number of execution exceptions |
| proxy_info | Proxy information. Different values of the name tag represent different informati |
| proxy_tra nsaction_commit_total | Transaction commit times |
| proxy_trans action_rollback_total | Transaction rollback times |
| build_info | Proxy build information. The name tag represents different components, and th |
| meta_data_info | Metadata information. Different name tag values represent different informatior |
| par se_dist_sql_rql_total | Total number of DistSQL RQL types parsed |
| par se_dist_sql_rdl_total | Total number of DistSQL RDL types parsed |
| par se_dist_sql_ral_total | Total number of DistSQL RAL types parsed |
| r oute_sql_select_total | Total number of select SQL statements executed by route |
| r oute_sql_insert_total | Total number of insert SQL statements executed by route |
| r oute_sql_update_total | Total number of update SQL statements executed by route |
| r oute_sql_delete_total | Total number of delete SQL statements executed by route |
| r oute_datasource_total | The number of data source routes. The name tag represents the added data sou |

| Metric | Description |
|---|---|
| route_table_total | The number of table routes. The name tag represents the table name |
| parse _sql_dml_insert_total | Total number of insert SQL statements parsed |
| parse _sql_dml_delete_total | Total number of delete SQL statements parsed |
| parse _sql_dml_update_total | Total number of update SQL statements parsed |
| parse _sql_dml_select_total | Total number of select SQL statements parsed |
| parse_sql_ddl_total | Total number of parsing DDL SQL statements |
| parse_sql_dcl_total | Total number of parsing DCL SQL statements |
| parse_sql_dal_total | Total number of parsing DAL SQL statements |
| parse_sql_tcl_total | Total number of parsing TCL SQL statements |
| proxy_transac tion_autocommit_total | Total number of auto commit transactions |
| proxy _connection_usage_sec | Connection duration |
| proxy_request_bytes | Number of requested bytes |
| proxy_response_bytes | Number of response bytes |
| rou te_sql_latency_millis | Routing SQL takes time |
| par se_sql_latency_millis | Parsing duration |
| parse_sql_in_commit | Number of SQL parsed in commit |
| parse_sql_in_rollback | Number of SQL parsed in rollback |
| proxy_exec ute_error_typed_total | The total number of execution errors. The name tag represents the exception cla |
| parse_sql_total | Total number of SQL parsed |
| route_sql_total | Total number of SQL routes |
| proxy_execute_total | Total number of tasks executed |
| proxy_backend _executor_thread_info | The backend executes thread pool information. Labels of different names repres |

## 8.16 Traffic Dual Routing

### 8.16.1 TrafficAlgorithm

| SPI Name | Description |
|---|---|
| TrafficAlgorithm | Forwarding matching algorithm |

| Implementation Class | Description |
|---|---|
| SQLHintTrafficAlgorithm | Forwarding matching algorithm based on SQL Hint |
| SQLMatchTrafficAlgorithm | Forwarding matching algorithm based on SQL string |
| SQLRegexTrafficAlgorithm | Regular forwarding matching algorithm based on SQL string |
| FirstSQLTrafficAlgorithm | Transaction forwarding matching algorithm based on the first SQL forwarding result |
| JDBCTrafficAlgorithm | Transaction forwarding matching algorithm for unified JDBC |
| ProxyTrafficAlgorithm | Transaction forwarding matching algorithm based on unified forwarding proxy |

### 8.16.2 TrafficLoadBalanceAlgorithm

| SPI Name | Description |
|---|---|
| TrafficLoadBalanceAlgorithm | Proxy instance load balancing algorithm |

| Implementation Class | Description |
|---|---|
| RandomTrafficLoadBalanceAlgorithm | Proxy instance load balancing algorithm |
| RoundRobinTrafficLoadBalanceAlgorithm | Load balancing algorithm for polling proxy instances |

# 9

# Best Practices

## 9.1 Authority Control

### 9.1.1 Authority Configuration

**Scenarios**

The authority engine performs system initialization according to the authority rules configured in the server.yaml.

**Data Planning**

- users are used to specify the initial user. For example, set root@% as the initial user.
- The type in the privilege is used to specify the selected service provider. For example, the enterprise authority provider SphereEx:PERMITTED is configured here.

**Notes**

1. The initial user has SUPER authority by default.
2. If the initial user is given non SUPER authorization through DistSQL, the initial user will lose SUPER authorization.
3. To grant SUPER authorization again, you need to use GRANT DIST SUPER TO user statement.

**Procedure**

The configuration format is as follows:

```yaml
authority:
 users:
  - user: root@%
    password: root
 privilege:
  type: SphereEx:PERMITTED
```

### 9.1.2 Do not Use Role Management

**Scenario**

An application system provides different levels of DBPlusEngine accounts for developers and operation and maintenance personnel. Among them, developers can only execute DML instructions, operation and maintenance personnel can execute DML + DDL instructions, and another root user is the top manager.

**Data Planning**

All account requirements are as follows:

| User Name | User | Required Authorities |
|---|---|---|
| root | Top Administrator | SUPER |
| zhangsan | Developer - Zhang San | DML |
| wangwu | Developer - Wang Wu | DML |
| develop_test | Developer and tester | DML |
| operator_1 | Operation and maintenance personnel-1 | DML + DDL |
| operator_2 | Operation and maintenance personnel-2 | DML + DDL |

The root user is the initial user.

**Procedure**

1. Create each developer and operation and maintenance user in turn, and set the password according to the actual situation.

```
-- The login host is not limited, and the host configuration is omitted.
CREATE DIST USER zhangsan IDENTIFIED BY '123456';
CREATE DIST USER wangwu IDENTIFIED BY '123456';
CREATE DIST USER develop_test IDENTIFIED BY '123456';
CREATE DIST USER operator_1 IDENTIFIED BY '123456';
CREATE DIST USER operator_2 IDENTIFIED BY '123456';
```

2. Authorize development users.

```
GRANT DIST INSERT,SELECT,UPDATE,DELETE TO zhangsan;
GRANT DIST INSERT,SELECT,UPDATE,DELETE TO wangwu;
GRANT DIST INSERT,SELECT,UPDATE,DELETE TO develop_test;
```

3. Authorize operation and maintenance users.

```
GRANT DIST INSERT,SELECT,UPDATE,DELETE,CREATE,ALTER,DROP,TRUNCATE TO operator_1;
GRANT DIST INSERT,SELECT,UPDATE,DELETE,CREATE,ALTER,DROP,TRUNCATE TO operator_2;
```

4. If you need to add new development users, repeat the following two steps.

```
-- Create new user.
CREATE DIST USER new_developer IDENTIFIED BY '123456';
-- Authorize
GRANT DIST INSERT,SELECT,UPDATE,DELETE TO new_developer;
```

5. If you need to add new operation and maintenance users, repeat the following two steps.

```
-- Create new user.
CREATE DIST USER new_operator IDENTIFIED BY '123456';
-- Authorize
GRANT DIST INSERT,SELECT,UPDATE,DELETE,CREATE,ALTER,DROP,TRUNCATE TO new_operator;
```

## 9.1.3 Using Role Management

### Scenarios

An application system provides different levels of DBPlusEngine accounts for developers and operation and maintenance personnel. Among them, developers can only execute DML instructions, operation and maintenance personnel can execute DML + DDL instructions, and another root user is the top manager.

### Data Planning

All account requirements are as follows:

| User Name | User | Required Authorities |
|---|---|---|
| root | Top Administrator | SUPER |
| zhangsan | Developer - Zhang San | DML |
| wangwu | Developer - Wang Wu | DML |
| develop_test | Developer and tester | DML |
| operator_1 | Operation and maintenance personnel-1 | DML + DDL |
| operator_2 | Operation and maintenance personnel-2 | DML + DDL |

### Procedure

1. Create each developer and operation and maintenance user in turn, and set the password according to the actual situation.

```
-- The login host is not limited, and the host configuration is omitted.
CREATE DIST USER zhangsan IDENTIFIED BY '123456';
CREATE DIST USER wangwu IDENTIFIED BY '123456';
CREATE DIST USER develop_test IDENTIFIED BY '123456';
CREATE DIST USER operator_1 IDENTIFIED BY '123456';
CREATE DIST USER operator_2 IDENTIFIED BY '123456';
```

2. Create two roles: develop_dml and operate_ddl.

```
CREATE DIST ROLE develop_dml;
CREATE DIST ROLE operate_ddl;
```

3. Authorize roles.

```
GRANT DIST INSERT,SELECT,UPDATE,DELETE TO develop_dml;
GRANT DIST INSERT,SELECT,UPDATE,DELETE,CREATE,ALTER,DROP,TRUNCATE TO operate_ddl;
```

4. Assign the developer to the user, so that the user has the authorizations owned by the role.

```
GRANT DIST develop_dml TO zhangsan;
GRANT DIST develop_dml TO wangwu;
GRANT DIST develop_dml TO develop_test;
```

5. Assign users to the operation and maintenance role.

```
GRANT DIST operate_ddl TO operator_1;
GRANT DIST operate_ddl TO operator_2;
```

6. If you need to add new developmers, repeat the following two steps.

```
-- Create new user.
CREATE DIST USER new_developer IDENTIFIED BY '123456';
-- Authorize
GRANT DIST develop_dml TO new_developer;
```

7. If you need to add new operation and maintenance users, repeat the following two steps.

```
-- Create new user.
CREATE DIST USER new_operator IDENTIFIED BY '123456';
-- Authorize
GRANT DIST operate_ddl TO new_operator;
```

## 9.2 Data Sharding

### Scenarios

In the context of the current Internet era, business data shows a rapid growth trend. In the case of the storage and access of massive data, there are many problems in the solution of single node storage of traditional relational database. It is difficult to meet the scenario of massive data in terms of performance, availability and operation and maintenance cost. Data sharding can split the data in a single database into multiple databases or tables according to a certain dimension, so as to improve performance and availability.

### Prerequisites

Take DBPlusEngine-Proxy as an example, download and unzip the proxy, refer to the following configuration, configure the corresponding configuration file in the conf directory, and then start the proxy.

### Configuration Example

config-sharding.yaml

```
rules:
- !SHARDING
 tables:
  t_order:
   databaseStrategy:
    standard:
     shardingColumn: user_id
     shardingAlgorithmName: database_inline
   tableStrategy:
    standard:
     shardingColumn: order_id
     shardingAlgorithmName: table_inline

 shardingAlgorithms:
  database_inline:
   type: INLINE
   props:
    algorithm-expression: ds_${user_id % 2}
  table_inline:
   type: INLINE
   props:
    algorithm-expression: t_order_${order_id % 2}
```

### Relevant Reference

Sharding

## 9.3 Distributed Transaction

### Scenarios

The DBPlusEngine distributed database solution provides support for distributed transactions, with commonly used modes being XA and BASE. Distributed transactions provide the semantics of final consistency. In XA mode, if the isolation level of storage DB is serializable, it can achieve strong consistency semantics. Here is how to use XA.

### Prerequisites

Taking DBPlusEngine-Proxy as an example, start by downloading and unzipping the proxy. Refer to the following configuration, configure the corresponding configuration file in the conf directory and start the proxy.

### Configuration Example

- Configuring with Atomikos

server.yaml

```yaml
rules:
 - !TRANSACTION
   defaultType: XA
   providerType: Atomikos
```

- Configuring with Narayana

Because the Narayana configuration is complicated, DBPlusEngine-Proxy provides automatic configuration. Now users do not need to manually configure jbossts-properties.xml configuration file.

DBPlusEngine will automatically configure Narayana specified in server.yaml, and generate the corresponding jbossts-properties.xml configuration file.

server.yaml

```yaml
rules:
- !TRANSACTION
  defaultType: XA
  providerType: Narayana
```

When Narayana is used as the XA transaction manager and the DB mode is configured to store XA recovery information, the DBPlusEngine-Proxy supports transferring the unrecovered transactions on the failed proxy instance to other proxies for recovery.

The configuration is as follows:

```yaml
- !TRANSACTION
  defaultType: XA
  providerType: Narayana
  props:
    recoveryStoreUrl: jdbc:mysql://127.0.0.1:3306/jbossts?serverTimezone=UTC&useSSL=false&allowPublicKeyRetrieval=true
    # mysql8 using com.mysql.cj.jdbc.MysqlDataSource
    recoveryStoreDataSource: com.mysql.jdbc.jdbc2.optional.MysqlDataSource
    recoveryStoreUser: databaseUser
    recoveryStorePassword: databasePwd
```

## 9.4 Read/Write Splitting

### Scenarios

With the growth of business volume, many applications will encounter the bottleneck of database throughput. It is difficult for a single database to carry a large number of concurrent queries and modifications. Currently, the database cluster with primary-secondary configuration has become an effective scheme. Primary-secondary configuration, that is, the master database is responsible for transactional operations such as data writing, modification and deletion, and the slave database is responsible for the database architecture of query operation. The database with master-slave configuration can limit the row lock brought by write operation to the master database, and support a large number of queries through the slave database, so as to greatly improve the performance of the application. In addition, the multi master and multi slave database configuration can be adopted to ensure that the system is still available even if the data node is down or even in the case of physical damage to the database.

The read/write splitting can seprately route query and write operations of different users to diffrent databases, so as to improve the read and write performance of the database. Read write splitting supports load balancing strategy, which can distribute requests evenly to different database nodes.

### Prerequisites

Take DBPlusEngine-Proxy as an example, download and unzip the proxy, refer to the following configuration, configure the corresponding configuration file in the conf directory, and then start the proxy.

### Configuration Example

config-readwrite-splitting.yaml

```yaml
rules:
- !READWRITE_SPLITTING
  dataSources:
    pr_ds:
      writeDataSourceName: write_ds
      readDataSourceNames: [read_ds_0, read_ds_1]
      loadBalancerName: weight_lb
  loadBalancers:
    weight_lb:
      type: WEIGHT
      props:
        read_ds_0: 2
        read_ds_1: 1
```

### Relevant Reference

Readwrite-splitting

## 9.5 Elastic Scaling

### Scenarios

Efficient capacity expansion and contraction. Scaling may be possible without moving any data. Shrinking may allow you to move only the necessary portion of the data.

### Prerequisites

Use the autoTables range sharding algorithm, for example: VOLUME_RANGE.

### Procedure

1. Add a database resource.

```sql
REGISTER STORAGE UNIT ds_0 (
  URL="jdbc:postgresql://host1:5432/scaling_ds_0",
  USER="postgres",
  PASSWORD="root",
  PROPERTIES("minPoolSize"="1","maxPoolSize"="20","idleTimeout"="60000")
), ds_1 (
  URL="jdbc:postgresql://host2:5432/scaling_ds_1",
  USER="postgres",
  PASSWORD="root",
  PROPERTIES("minPoolSize"="1","maxPoolSize"="20","idleTimeout"="60000")
);
```

2. Create table rules.

```
CREATE SHARDING TABLE RULE t_order(
STORAGE_UNITS(ds_0,ds_1),
SHARDING_COLUMN=order_id,
TYPE(NAME="VOLUME_RANGE",PROPERTIES("range-lower"="1","range-upper"="100000000","sharding-volume"="10000000")),
KEY_GENERATE_STRATEGY(COLUMN=order_id,TYPE(NAME="snowflake"))
);
```

3. Create a new table.

```
CREATE TABLE t_order (order_id INT NOT NULL, user_id INT NOT NULL, status VARCHAR(45) NULL, PRIMARY KEY (order_id));
```

4. Insert into some data.

For example:

```
INSERT INTO t_order (order_id, user_id, status) VALUES
(1,2,'ok'),
(101,2,'ok'),
(201,2,'ok');
```

5. Add new database resources.

```
REGISTER STORAGE UNIT ds_2 (
  URL="jdbc:postgresql://host3:5432/scaling_ds_10",
  USER="postgres",
  PASSWORD="root",
  PROPERTIES("minPoolSize"="1","maxPoolSize"="20","idleTimeout"="60000")
);
```

6. Trigger scaling.

```
RESHARD TABLE t_order BY(
STORAGE_UNITS(ds_0,ds_1,ds_2),
SHARDING_COLUMN=order_id,
TYPE(NAME="VOLUME_RANGE",PROPERTIES("range-lower"="1","range-upper"="150000000","sharding-volume"="10000000")),
KEY_GENERATE_STRATEGY(COLUMN=order_id,TYPE(NAME="snowflake"))
);
```

Only new sub-tables are created, no data is moved.

## 9.6 Data Encryption

### Scenarios

With the spread of information technology, more and more companies realize the important value of data assets, resulting in data security being paid more and more attention. As an important means to protect data security, data encryption has naturally become the basic demand of various companies. Data encryption is a data protection method that uses encryption rules to deform data to achieve the purpose of security control.

In reality, there are often two business scenarios of data encryption. One is that new businesses need data encryption. Because they are new businesses, everything is new. Business teams often simply implement data encryption based on the basic requirements of the company’ s encryption. However, with the rapid development of business, the original encryption scheme is difficult to meet the requirements of new business scenarios, resulting in the need for large-scale transformation of business systems, and the cost of upgrading is huge. The other is the mature business that has been launched and the data is stored in plaintext. Now that the company has the requirements of data encryption, it will involve the migration and encryption of old data (washing number) and the related transformation of business SQL. The overall complexity is high. For the core business, it also needs to be transformed without shutdown, which will involve the construction of pre release environment and the preparation of rollback scheme, which will cost a lot.

Based on this background, SphereEx-DBPlusEngine has proposed a complete, safe, transparent and low transformation cost data encryption integration scheme to meet the encryption and decryption needs of various companies from the needs of the industry and the pain points of business transformation.

### Prerequisites

Take DBPlusEngine-Proxy as an example, download and unzip the proxy, refer to the following configuration, configure the corresponding configuration file in the conf directory, and then start the proxy.

### Configuration Example

config-encrypt.yaml

```
-!ENCRYPT
   encryptors:
    aes_encryptor:
      type: AES
      props:
        aes-key-value: 123456abc
   tables:
    t_user:
      columns:
        pwd:
          cipherColumn: pwd_cipher
          encryptorName: aes_encryptor
```

### Relevant Reference

Encryption

## 9.7 Shadow DB

### Scenarios

SphereEx-DBPlusEngine focuses on solutions at the database level in the scenario of full link online pressure test. Launch the shadow DB function of pressure test, with the help of the powerful SQL parsing ability of SphereEx-DBPlusEngine. Perform shadow judgment on executing SQL. At the same time, the shadow algorithm is combined with flexible configuration. To meet the online pressure test requirements of complex business scenarios, the pressure test flow is routed to the shadow DB, and the online normal flow is routed to the production database.

### Prerequisites

Take DBPlusEngine-Proxy as an example, download and unzip the proxy, refer to the following configuration, configure the corresponding configuration file in the conf directory, and then start the proxy.

### Configuration Example

config-shadow.yaml

```
rules:
- !SHADOW
  dataSources:
    shadowDataSource:
      sourceDataSourceName: ds
      shadowDataSourceName: ds_shadow
  tables:
```

```
  t_user:
   dataSourceNames:
    - shadowDataSource
   shadowAlgorithmNames:
    - user_id_insert_value_match-algorithm
    - simple-hint-algorithm
 shadowAlgorithms:
  user_id_insert_value_match-algorithm:
   type: VALUE_MATCH
   props:
    operation: insert
    column: user_id
    value: 1
  simple-hint-algorithm:
   type: SIMPLE_HINT
   props:
    foo: bar

- !SQL_PARSER
 sqlCommentParseEnabled: true
```

### Relevant Reference

Shadow DB

## 9.8 Cluster Deployment

### Scenarios

Considering the deployment scenarios of users, SphereEx-DBPlusEngine provides three operation modes, and cluster mode is one of them. Cluster mode is the recommended production deployment mode for SphereEx-DBPlusEngine. In the cluster mode, the horizontal capacity expansion can be achieved by adding computing nodes. At the same time, multi-node deployment is also the basis to ensure the high availability of services.

- Operation Mode

Cluster mode is one of the operation modes of SphereEx-DBPlusEngine. Cluster mode is applicable to the deployment of production environment. In addition to cluster mode, SphereEx-DBPlusEngine also provides memory mode and stand-alone mode, which are used for integration test and local development test respectively. Unlike stand-alone mode, memory mode does not persist any metadata and configuration information, all modifications take effect in the current process. The operation mode of SphereEx-DBPlusEngine covers all scenarios of users from development to test and production deployment.

- Registration Center

The registry center is the basis for the implementation of the cluster mode. SphereEx-DBPlusEngine realizes the sharing of metadata and configuration in the cluster environment by integrating the third-party registry components ZooKeeper and etcd. At the same time, with the help of the notification and coordination ability of the registry center, it ensures the real-time synchronization of the cluster when the shared data changes.

### Prerequisites

Take DBPlusEngine-Proxy as an example, download and unzip the proxy, refer to the following configuration, configure the corresponding configuration file in the conf directory, and then start the proxy.

### Configuration Example

If the cluster mode needs to be enabled in the production environment, it needs to be enabled by configuring mode tag in server.yaml:

```
mode:
 type: Cluster
 repository:
  type: ZooKeeper
  props:
   namespace: governance_ds
   server-lists: localhost:2181
   retryIntervalMilliseconds: 500
   timeToLiveSeconds: 60
   maxRetries: 3
   operationTimeoutMilliseconds: 500
 overwrite: false
```

At the same time, when multiple compute nodes need to be added to the cluster, it is necessary to ensure that the configurations of ‘namespace’ and ‘server lists’ are the same to ensure that these compute nodes work in the same cluster.

If users need to use local configuration to initialize or overwrite the configuration in the cluster, they can configure ‘overwrite: true’ .

## 9.9 Proxy + LDAP & LDAPS Application Case

### 9.9.1 Background

SphereEx-DBPlusEngine adds support for LDAP login authentication. The following application case shows the use process of LDAP login authentication.

In the process of case display, Wireshark tool is used to capture packets to more vividly show the difference between LDAP and LDAPS protocols.

LDAPS is an LDAP communication mode based on SSL/TLS.

### 9.9.2 Basic Environment

| Name | Version |
|---|---|
| MySQL | 5.7 or 8.0 |
| SphereEx DBPlusEngine | 1.0 |
| Wireshark | 3.6 |
| ApacheDS | 2.0.0 |

- config-sharding-databases.yaml

```
schemaName: sharding_db

dataSources:
 ds_0:
  url: jdbc:mysql://127.0.0.1:3306/demo_ds_0?serverTimezone=UTC&useSSL=false
  username: root
  password:
  connectionTimeoutMilliseconds: 30000
  idleTimeoutMilliseconds: 60000
  maxLifetimeMilliseconds: 1800000
  maxPoolSize: 10
  minPoolSize: 1
 ds_1:
  url: jdbc:mysql://127.0.0.1:3306/demo_ds_1?serverTimezone=UTC&useSSL=false
  username: root
  password:
  connectionTimeoutMilliseconds: 30000
  idleTimeoutMilliseconds: 60000
  maxLifetimeMilliseconds: 1800000
  maxPoolSize: 10
  minPoolSize: 1

rules:
- !SHARDING
 tables:
  t_order:
   actualDataNodes: ds_${0..1}.t_order
   keyGenerateStrategy:
    column: order_id
    keyGeneratorName: snowflake
 defaultDatabaseStrategy:
  standard:
   shardingColumn: user_id
   shardingAlgorithmName: database_inline
 defaultTableStrategy:
  none:
 shardingAlgorithms:
  database_inline:
   type: INLINE
   props:
    algorithm-expression: ds_${user_id % 2}
 keyGenerators:
  snowflake:
   type: SNOWFLAKE
```

### 9.9.3 LDAP Server Configuration

Adopt Docker image of ApacheDS™: tremolosecurity/apacheds。

a. Pull image.

```
docker pull tremolosecurity/apacheds:latest
```

b. Generate SSL certificate.

Description page: https://directory.apache.org/apacheds/basic-ug/3.3-enabling-ssl.html

Use keytool to generate the certificate according to the method in the page.

Take Common name as localhost as an example to generate two files:

```
.
├── localhost.ks
└── localhost.cer
```

Because the container used has special requirements for file names, rename localhost.ks as apacheds.jks.

```
.
├── apacheds.jks
└── localhost.cer
```

Note: The certificate file can be saved in any path, such as:

- /Users/${yourname}/apacheds/apacheds.jks
- /Users/${yourname}/apacheds/localhost.cer

c. Start container.

```
docker run --detach --rm --name apacheds \
 -p 10389:10389 \
 -p 10636:10636 \
 -v /Users/${yourname}/apacheds:/etc/apacheds \
 -e APACHEDS_ROOT_PASSWORD=secret \
 -e APACHEDS_TLS_KS_PWD=secret \
 tremolosecurity/apacheds:latest
```

Notes:

- The container maps two ports, of which 10389 is used as LDAP non encrypted connection and 10636 is used as LDAPS encrypted connection.
- ApacheDS service contains a default user uid=admin, ou=system, which can be accessed through the parameter APACHEDS_ROOT_PASSWORD and its password is secret.

After starting the operation, check whether the log is normal:

```
docker logs -f apacheds
```

d. ldapsearch test

The ldapsearch command can easily initiate access to the LDAP service and verify whether the LDAP service is normal:

```
docker exec -it apacheds ldapsearch -x -H ldap://localhost:10389 -b ou=system -D "uid=admin,ou=system" -w secret
```

At this point, the LDAP server configuration is complete.

### 9.9.4 JDK Import Certificate

Since the LDAP server uses a self signed certificate, you need to import it into the keystore of JRE before accessing the client.

Note: During the import process, you need to enter the key of the certificate: secret.

```
keytool -import -alias localhost -keystore $JAVA_HOME/jre/lib/security/cacerts -file /Users/${yourname}/apacheds/localhost.cer
```

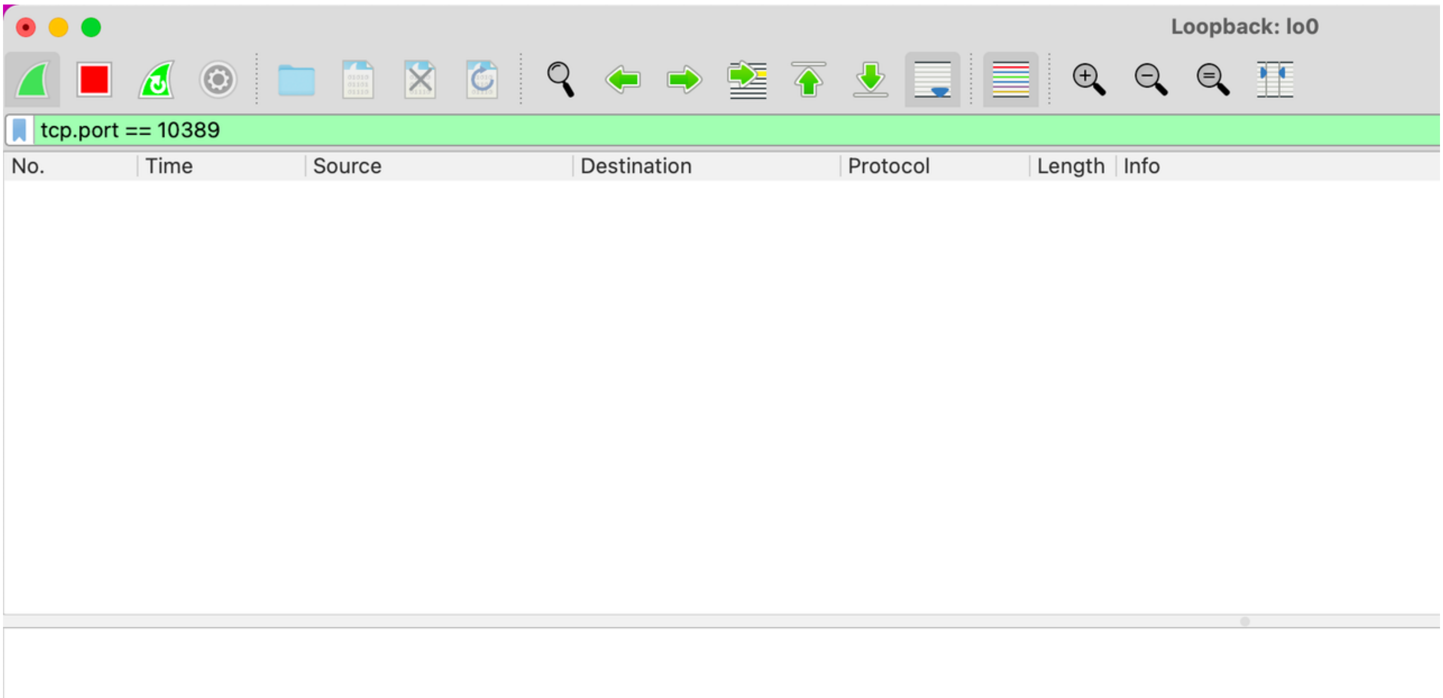
### 9.9.5 Proxy-LDAP Test

a. server.yaml

```
authority:
 users:
  - user: root@%
  - user: admin
  - user: sharding
 authenticators:
  auth_ldap:
   type: LDAP
   props:
    ldap_server_url: ldap://localhost:10389
    ldap_dn_template: uid={0},ou=system
 defaultAuthenticator: auth_ldap
```

b. Start Proxy.

c. Start Wireshark and start capturing packets on port 10389.

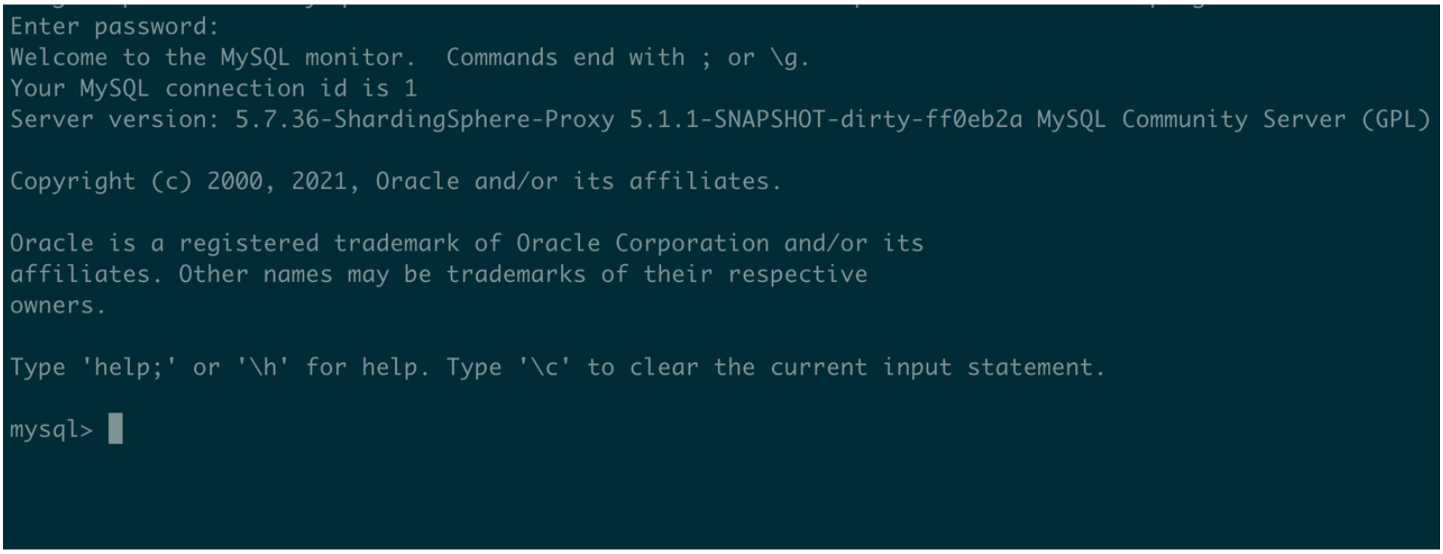


d. MySQL client login test.

Note that the parameter –enable-cleartext-plugin is specified.

```
# Since there is only one admin user in the LDAP server, use admin to log in.
# If you try another user, the result is login failure.
mysql -h 127.0.0.1 -P 3307 -A -u admin -p --enable-cleartext-plugin
```

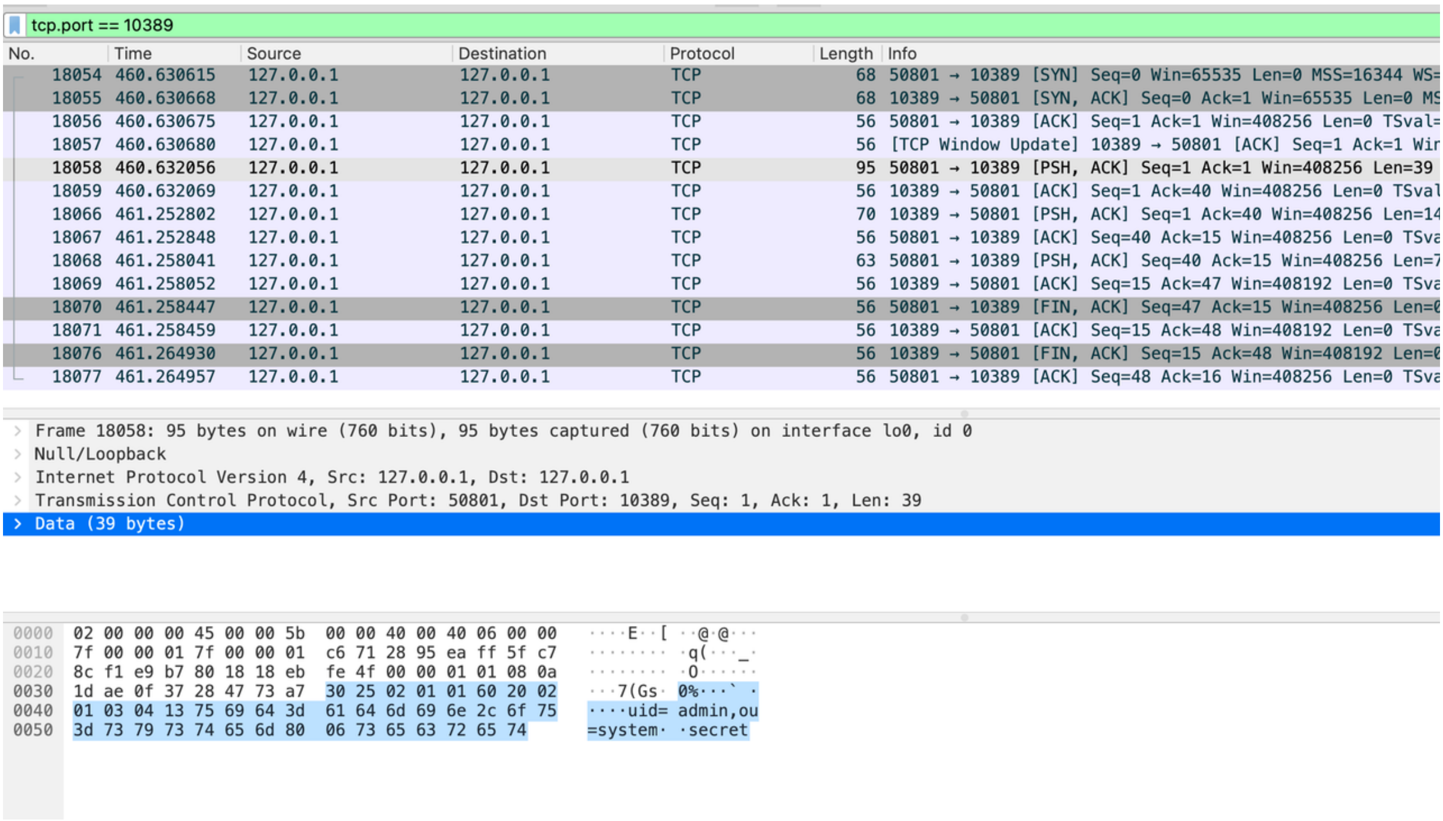
Login succeeded after entering the password secret:



```
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1
Server version: 5.7.36-ShardingSphere-Proxy 5.1.1-SNAPSHOT-dirty-ff0eb2a MySQL Community Server (GPL)

Copyright (c) 2000, 2021, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> 
```

e. View packet capture data.

From the captured TCP packet, we can easily find a message containing user DN and password:



```
tcp.port == 10389
No.    Time        Source     Destination  Protocol  Length Info
18054 460.630615  127.0.0.1  127.0.0.1    TCP       68 50801 → 10389 [SYN] Seq=0 Win=65535 Len=0 MSS=16344 WS=
18055 460.630668  127.0.0.1  127.0.0.1    TCP       68 10389 → 50801 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MS
18056 460.630675  127.0.0.1  127.0.0.1    TCP       56 50801 → 10389 [ACK] Seq=1 Ack=1 Win=408256 Len=0 TSval=
18057 460.630680  127.0.0.1  127.0.0.1    TCP       56 [TCP Window Update] 10389 → 50801 [ACK] Seq=1 Ack=1 Wir
18058 460.632056  127.0.0.1  127.0.0.1    TCP       95 50801 → 10389 [PSH, ACK] Seq=1 Ack=1 Win=408256 Len=39
18059 460.632069  127.0.0.1  127.0.0.1    TCP       56 10389 → 50801 [ACK] Seq=1 Ack=40 Win=408256 Len=0 TSval
18066 461.252802  127.0.0.1  127.0.0.1    TCP       70 10389 → 50801 [PSH, ACK] Seq=1 Ack=40 Win=408256 Len=14
18067 461.252848  127.0.0.1  127.0.0.1    TCP       56 50801 → 10389 [ACK] Seq=40 Ack=15 Win=408256 Len=0 TSva
18068 461.258041  127.0.0.1  127.0.0.1    TCP       63 50801 → 10389 [PSH, ACK] Seq=40 Ack=15 Win=408256 Len=7
18069 461.258052  127.0.0.1  127.0.0.1    TCP       56 10389 → 50801 [ACK] Seq=15 Ack=47 Win=408192 Len=0 TSva
18070 461.258447  127.0.0.1  127.0.0.1    TCP       56 50801 → 10389 [FIN, ACK] Seq=47 Ack=15 Win=408256 Len=0
18071 461.258459  127.0.0.1  127.0.0.1    TCP       56 10389 → 50801 [ACK] Seq=15 Ack=48 Win=408192 Len=0 TSva
18076 461.264930  127.0.0.1  127.0.0.1    TCP       56 10389 → 50801 [FIN, ACK] Seq=15 Ack=48 Win=408192 Len=0
18077 461.264957  127.0.0.1  127.0.0.1    TCP       56 50801 → 10389 [ACK] Seq=48 Ack=16 Win=408256 Len=0 TSva

> Frame 18058: 95 bytes on wire (760 bits), 95 bytes captured (760 bits) on interface lo0, id 0
> Null/Loopback
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
> Transmission Control Protocol, Src Port: 50801, Dst Port: 10389, Seq: 1, Ack: 1, Len: 39
> Data (39 bytes)

0000  02 00 00 00 45 00 00 5b  00 00 40 00 40 06 00 00   ····E··[ ··@·@···
0010  7f 00 00 01 7f 00 00 01  c6 71 28 95 ea ff 5f c7   ········ ·q(···_·
0020  8c f1 e9 b7 80 18 18 eb  fe 4f 00 00 01 01 08 0a   ········ ·O······
0030  1d ae 0f 37 28 47 73 a7  30 25 02 01 01 60 20 02   ···7(Gs· 0%···` ·
0040  01 03 04 13 75 69 64 3d  61 64 6d 69 6e 2c 6f 75   ····uid= admin,ou
0050  3d 73 79 73 74 65 6d 80  06 73 65 63 72 65 74      =system· ·secret
```

f. Summary

From server.yaml configuration, we can see that under the premise of existing LDAP server, using LDAP for login authentication is not complicated, and only simple configuration is required.

On the other hand, because LDAP protocol is unencrypted, there is a risk of password disclosure when using LDAP authentication in public networks.

### 9.9.6 Proxy-LDAPS Test

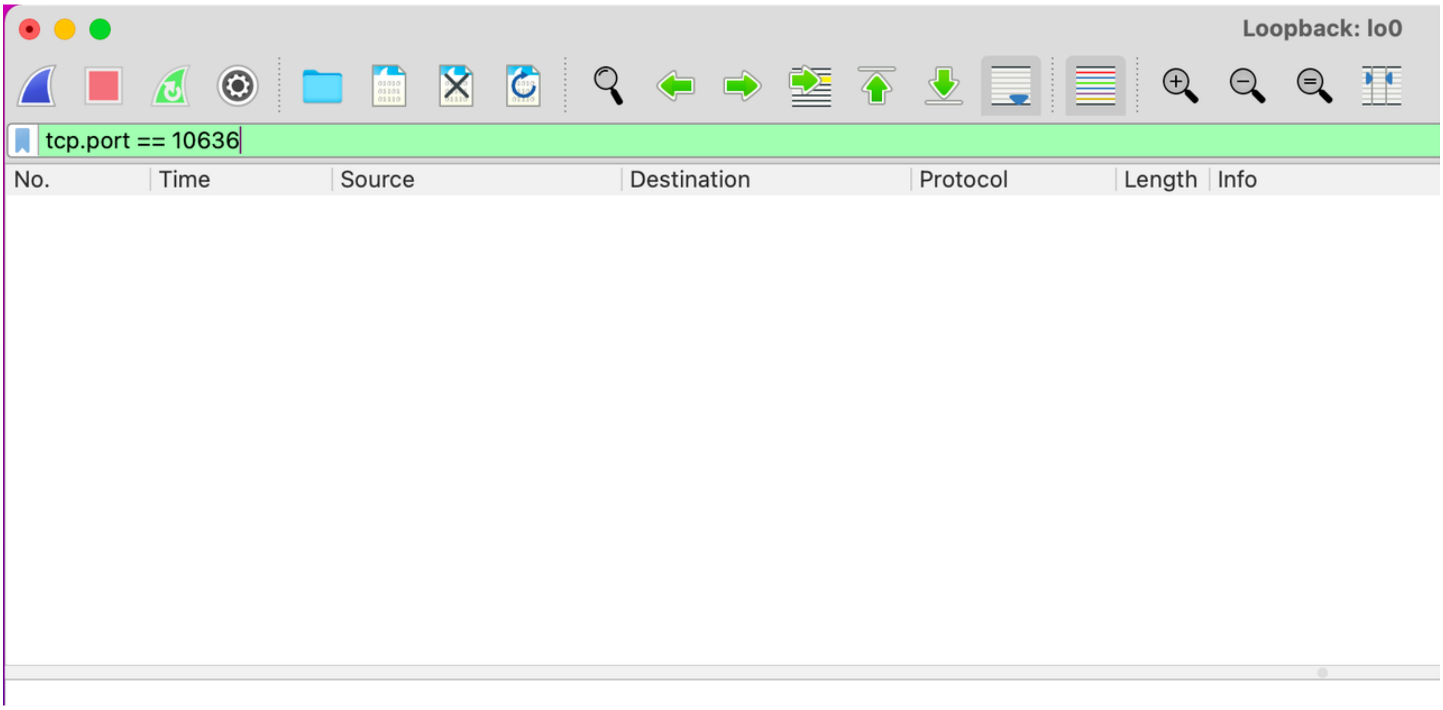
a. server.yaml

The only difference from the LDAP case is that the URL of the LDAP server is changed.

```yaml
authority:
 users:
  - user: root@%
  - user: admin
  - user: sharding
 authenticators:
  auth_ldap:
   type: LDAP
   props:
    ldap_server_url: ldaps://localhost:10636
    ldap_dn_template: uid={0},ou=system
 defaultAuthenticator: auth_ldap
```

b. Start Proxy .

c. Start Wireshark and start capturing packets on port 10636.



tcp.port == 10636

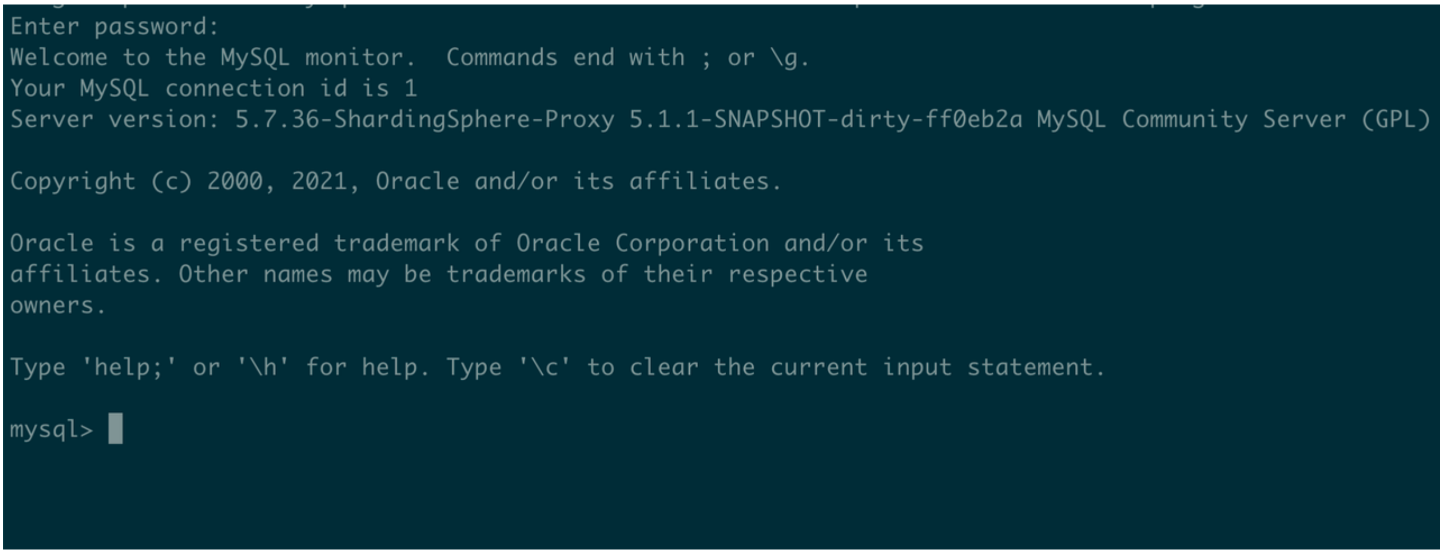
| No. | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|

d. MySQL client login test.

Note: need to specify the parameter –enable-cleartext-plugin.

```
# Since there is only one admin user in the LDAP server, use admin to log in.
# If you try another user, the result is login failure.
mysql -h 127.0.0.1 -P 3307 -A -u admin -p --enable-cleartext-plugin
```

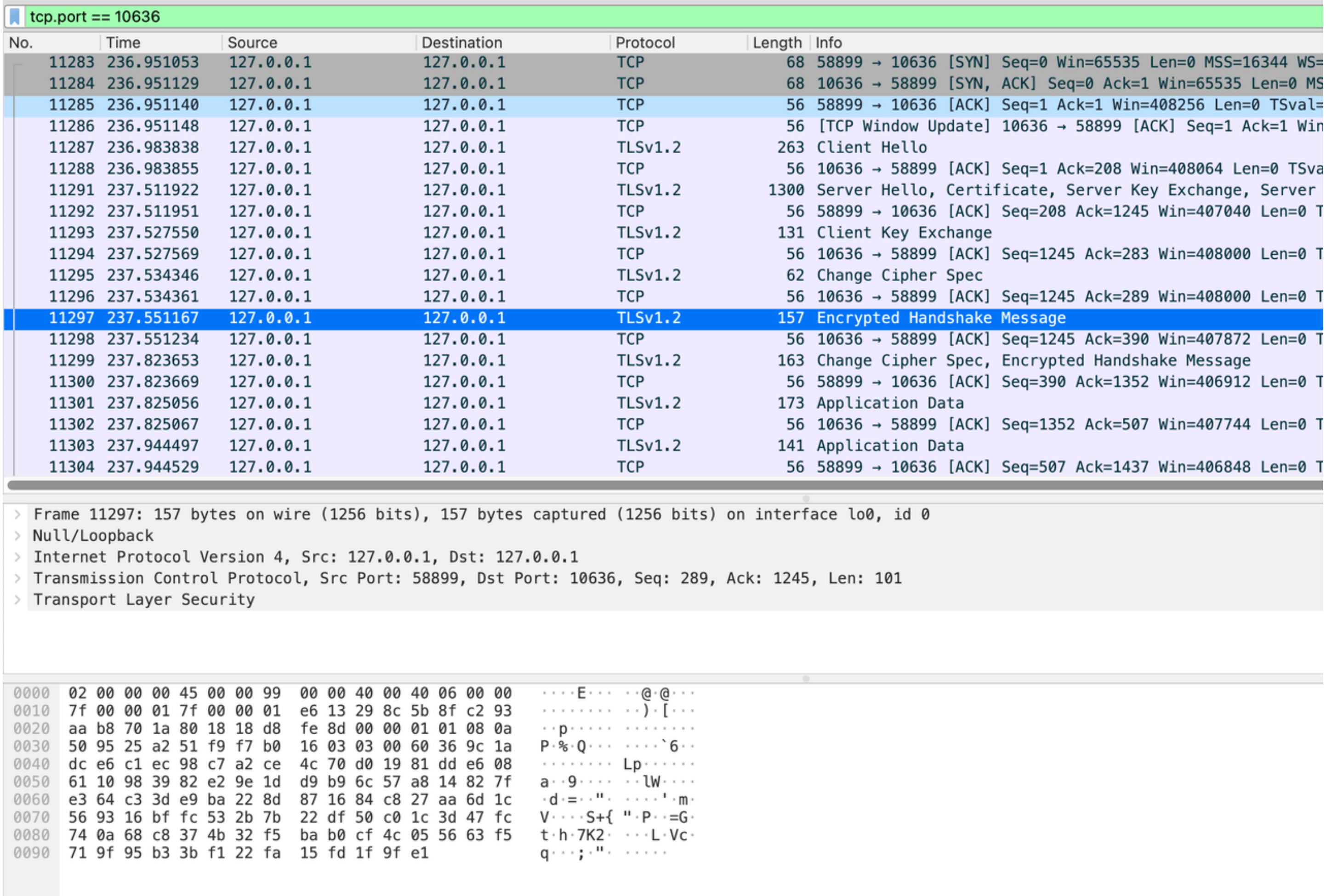
Login succeeded after entering the password secret:



```
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1
Server version: 5.7.36-ShardingSphere-Proxy 5.1.1-SNAPSHOT-dirty-ff0eb2a MySQL Community Server (GPL)

Copyright (c) 2000, 2021, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

e. View packet capture data.

It can be seen from the data packet that TLS communication has been established between proxy and LDAP server, and the content of communication cannot be obtained through packet capturing:



tcp.port == 10636

```
No.    Time        Source      Destination  Protocol  Length Info
11283  236.951053  127.0.0.1   127.0.0.1    TCP           68 58899 → 10636 [SYN] Seq=0 Win=65535 Len=0 MSS=16344 WS=
11284  236.951129  127.0.0.1   127.0.0.1    TCP           68 10636 → 58899 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MS
11285  236.951140  127.0.0.1   127.0.0.1    TCP           56 58899 → 10636 [ACK] Seq=1 Ack=1 Win=408256 Len=0 TSval=
11286  236.951148  127.0.0.1   127.0.0.1    TCP           56 [TCP Window Update] 10636 → 58899 [ACK] Seq=1 Ack=1 Win
11287  236.983838  127.0.0.1   127.0.0.1    TLSv1.2      263 Client Hello
11288  236.983855  127.0.0.1   127.0.0.1    TCP           56 10636 → 58899 [ACK] Seq=1 Ack=208 Win=408064 Len=0 TSva
11291  237.511922  127.0.0.1   127.0.0.1    TLSv1.2     1300 Server Hello, Certificate, Server Key Exchange, Server
11292  237.511951  127.0.0.1   127.0.0.1    TCP           56 58899 → 10636 [ACK] Seq=208 Ack=1245 Win=407040 Len=0 T
11293  237.527550  127.0.0.1   127.0.0.1    TLSv1.2      131 Client Key Exchange
11294  237.527569  127.0.0.1   127.0.0.1    TCP           56 10636 → 58899 [ACK] Seq=1245 Ack=283 Win=408000 Len=0 T
11295  237.534346  127.0.0.1   127.0.0.1    TLSv1.2       62 Change Cipher Spec
11296  237.534361  127.0.0.1   127.0.0.1    TCP           56 10636 → 58899 [ACK] Seq=1245 Ack=289 Win=408000 Len=0 T
11297  237.551167  127.0.0.1   127.0.0.1    TLSv1.2      157 Encrypted Handshake Message
11298  237.551234  127.0.0.1   127.0.0.1    TCP           56 10636 → 58899 [ACK] Seq=1245 Ack=390 Win=407872 Len=0 T
11299  237.823653  127.0.0.1   127.0.0.1    TLSv1.2      163 Change Cipher Spec, Encrypted Handshake Message
11300  237.823669  127.0.0.1   127.0.0.1    TCP           56 58899 → 10636 [ACK] Seq=390 Ack=1352 Win=406912 Len=0 T
11301  237.825056  127.0.0.1   127.0.0.1    TLSv1.2      173 Application Data
11302  237.825067  127.0.0.1   127.0.0.1    TCP           56 10636 → 58899 [ACK] Seq=1352 Ack=507 Win=407744 Len=0 T
11303  237.944497  127.0.0.1   127.0.0.1    TLSv1.2      141 Application Data
11304  237.944529  127.0.0.1   127.0.0.1    TCP           56 58899 → 10636 [ACK] Seq=507 Ack=1437 Win=406848 Len=0 T
```

```
> Frame 11297: 157 bytes on wire (1256 bits), 157 bytes captured (1256 bits) on interface lo0, id 0
> Null/Loopback
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
> Transmission Control Protocol, Src Port: 58899, Dst Port: 10636, Seq: 289, Ack: 1245, Len: 101
> Transport Layer Security
```

```
0000  02 00 00 00 45 00 00 99  00 00 40 00 40 06 00 00   ····E···  ··@·@···
0010  7f 00 00 01 7f 00 00 01  e6 13 29 8c 5b 8f c2 93   ········  ··)·[···
0020  aa b8 70 1a 80 18 18 d8  fe 8d 00 00 01 01 08 0a   ··p·····  ········
0030  50 95 25 a2 51 f9 f7 b0  16 03 03 00 60 36 9c 1a   P·%·Q···  ····`6··
0040  dc e6 c1 ec 98 c7 a2 ce  4c 70 d0 19 81 dd e6 08   ········  Lp······
0050  61 10 98 39 82 e2 9e 1d  d9 b9 6c 57 a8 14 82 7f   a··9····  ··lW····
0060  e3 64 c3 3d e9 ba 22 8d  87 16 84 c8 27 aa 6d 1c   ·d·=··"·  ····'·m·
0070  56 93 16 bf fc 53 2b 7b  22 df 50 c0 1c 3d 47 fc   V····S+{  "·P··=G·
0080  74 0a 68 c8 37 4b 32 f5  ba b0 cf 4c 05 56 63 f5   t·h·7K2·  ···L·Vc·
0090  71 9f 95 b3 3b f1 22 fa  15 fd 1f 9f e1            q···;·"·  ·····
```

f. Summary

SSL/TLS encryption can effectively protect user login information.  The SphereEx-DBPlusEngine is very friendly to LDAPS. You can switch from LDAP to LDAPS by importing certificates and replacing URLs.

# 10 Test Manual

DBPlusEngine provides test engines for integration, module and performance.

## 10.1 Integration Test

Provide point to point test which connect real DBPlusEngine and database instances.

They define SQLs in XML files, engine run for each database independently. All test engines designed to modify the configuration files to execute all assertions without any **Java code** modification. It does not depend on any third-party environment, DBPlusEngine-Proxy and database used for testing are provided by docker image.

## 10.2 Module Test

Provide module test engine for complex modules.

They define SQLs in XML files, engine run for each database independently too. It includes SQL parser and SQL rewriter modules.

## 10.3 Performance Test

Provide multiple performance test methods, includes Sysbench, JMH or TPCC and so on.

## 10.4 Integration Test

The SQL parsing unit test covers both SQL placeholder and literal dimension. Integration test can be further divided into two dimensions of strategy and JDBC. The former one includes strategies as Sharding, table Sharding, database Sharding, and read/write splitting while the latter one includes Statement and PreparedStatement.

Therefore, one SQL can drive 5 kinds of database parsing * 2 kinds of parameter transmission modes + 5 kinds of databases * 5 kinds of Sharding strategies * 2 kinds of JDBC operation modes = 60 test cases, to enable DBPlusEngine to achieve the pursuit of high quality.

### 10.4.1 Process

The Parameterized in JUnit will collect all test data, and pass to test method to assert one by one. The process of handling test data is just like a leaking hourglass:

#### Configuration

- environment type
  - /shardingsphere-integration-test-suite/src/test/resources/env-native.properties
  - /shardingsphere-integration-test-suite/src/test/resources/env/SQL-TYPE/dataset.xml
  - /shardingsphere-integration-test-suite/src/test/resources/env/SQL-TYPE/schema.xml
- test case type
  - /shardingsphere-integration-test-suite/src/test/resources/cases/SQL-TYPE/SQL-TYPE-integration-test-cases.xml
  - /shardingsphere-integration-test-suite/src/test/resources/cases/SQL-TYPE/dataset/FEATURE-TYPE/*.xml
- sql-case
  - /sharding-sql-test/src/main/resources/sql/sharding/SQL-TYPE/*.xml

#### Environment Configuration

Integration test depends on existed database environment, developers need to setup the configuration file for corresponding database to test:

Firstly, setup configuration file /shardingsphere-integration-test-suite/src/test/resources/env-native.properties, for example:

```
# the switch for PK, concurrent, column index testing and so on
it.run.additional.cases=false

# test scenarios, could define multiple rules
it.scenarios=db,tbl,dbtbl_with_replica_query,replica_query

# database type, could define multiple databases(H2,MySQL,Oracle,SQLServer,PostgreSQL)
it.cluster.databases=MySQL,PostgreSQL

# MySQL configuration
it.mysql.host=127.0.0.1
it.mysql.port=13306
it.mysql.username=root
it.mysql.password=root

## PostgreSQL configuration
it.postgresql.host=db.psql
it.postgresql.port=5432
it.postgresql.username=postgres
it.postgresql.password=postgres

## SQLServer configuration
it.sqlserver.host=db.mssql
it.sqlserver.port=1433
it.sqlserver.username=sa
it.sqlserver.password=Jdbc1234

## Oracle configuration
it.oracle.host=db.oracle
it.oracle.port=1521
it.oracle.username=jdbc
it.oracle.password=jdbc
```

Secondly, setup configuration file /shardingsphere-integration-test-suite/src/test/resources/env/SQL-TYPE/dataset.xml. Developers can set up metadata and expected data to start the data initialization in dataset.xml. For example:

```
<dataset>
  <metadata data-nodes="tbl.t_order_${0..9}">
    <column name="order_id" type="numeric" />
    <column name="user_id" type="numeric" />
    <column name="status" type="varchar" />
  </metadata>
  <row data-node="tbl.t_order_0" values="1000, 10, init" />
  <row data-node="tbl.t_order_1" values="1001, 10, init" />
  <row data-node="tbl.t_order_2" values="1002, 10, init" />
  <row data-node="tbl.t_order_3" values="1003, 10, init" />
  <row data-node="tbl.t_order_4" values="1004, 10, init" />
  <row data-node="tbl.t_order_5" values="1005, 10, init" />
  <row data-node="tbl.t_order_6" values="1006, 10, init" />
  <row data-node="tbl.t_order_7" values="1007, 10, init" />
  <row data-node="tbl.t_order_8" values="1008, 10, init" />
  <row data-node="tbl.t_order_9" values="1009, 10, init" />
</dataset>
```

Developers can customize DDL to create databases and tables in schema.xml.

### Assertion Configuration

So far we have confirmed what kind of SQL execute in which environment in upon configuration, here we define the data for assert. There are two kinds of config for assert, one is at /shardingsphere-integration-test-suite/src/test/resources/cases/SQL-TYPE/SQL-TYPE-integration-test-cases.xml. This file just like an index, defined the sql, parameters and expected index position for execution. the SQL is the value for sql-case-id. For example:

```
<integration-test-cases>
  <dml-test-case sql-case-id="insert_with_all_placeholders">
    <assertion parameters="1:int, 1:int, insert:String" expected-data-file="insert_for_order_1.xml" />
    <assertion parameters="2:int, 2:int, insert:String" expected-data-file="insert_for_order_2.xml" />
  </dml-test-case>
</integration-test-cases>
```

Another kind of config for assert is the data, as known as the corresponding expected-data-file in SQL-TYPE-integration-test-cases.xml, which is at /shardingsphere-integration-test-suite/src/test/resources/cases/SQL-TYPE/dataset/FEATURE-TYPE/*.xml.
This file is very like the dataset.xml mentioned before, and the difference is that expected-data-file contains some other assert data, such as the return value after a sql execution. For examples:

```
<dataset update-count="1">
  <metadata data-nodes="db_${0..9}.t_order">
    <column name="order_id" type="numeric" />
    <column name="user_id" type="numeric" />
    <column name="status" type="varchar" />
  </metadata>
  <row data-node="db_0.t_order" values="1000, 10, update" />
  <row data-node="db_0.t_order" values="1001, 10, init" />
  <row data-node="db_0.t_order" values="2000, 20, init" />
  <row data-node="db_0.t_order" values="2001, 20, init" />
</dataset>
```

Util now, all config files are ready, just launch the corresponding test case is fine.With no need to modify any Java code, only set up some config files. This will reduce the difficulty for DBPlusEngine testing.

### 10.4.2 Notice

1. If Oracle needs to be tested, please add Oracle driver dependencies to the pom.xml.
2. 10 splitting-databases and 10 splitting-tables are used in the integrated test to ensure the test data is full, so it will take a relatively long time to run the test cases.

## 10.5 Performance Test

Provides result for each performance test tools.

### 10.5.1 SysBench DBPlusEngine-Proxy Empty Rule Performance Test

#### Objectives

Compare the performance of DBPlusEngine-Proxy and MySQL 1. Sysbench directly carries out stress testing on the performance of MySQL. 2. Sysbench directly carries out stress testing on DBPlusEngine-Proxy (directly connect MySQL).

Based on the above two groups of experiments, we can figure out the loss of MySQL when using DBPlusEngine-Proxy.

#### Set up the test environment

#### Server information

1. Db-related configuration: it is recommended that the memory is larger than the amount of data to be tested, so that the data is stored in the memory hot block, and the rest can be adjusted.
2. DBPlusEngine-Proxy-related configuration: it is recommended to use a high-performance, multi-core CPU, and other configurations can be customized.
3. Disable swap partitions on all servers involved in the stress testing.

#### Database

```
[mysqld]
innodb_buffer_pool_size=${MORE_THAN_DATA_SIZE}
innodb-log-file-size=3000000000
innodb-log-files-in-group=5
innodb-flush-log-at-trx-commit=0
innodb-change-buffer-max-size=40
back_log=900
innodb_max_dirty_pages_pct=75
innodb_open_files=20480
innodb_buffer_pool_instances=8
innodb_page_cleaners=8
innodb_purge_threads=2
innodb_read_io_threads=8
innodb_write_io_threads=8
table_open_cache=102400
log_timestamps=system
thread_cache_size=16384
```

```
transaction_isolation=READ-COMMITTED

# Appropriate tuning can be considered to magnify the underlying DB performance, so that the experiment doesn't subject to DB performance bottleneck.
```

### Stress testing tool

Refer to sysbench’ s GitHub

### DBPlusEngine-Proxy

#### bin/start.sh

```
-Xmx16g -Xms16g -Xmn8g  # Adjust JVM parameters
```

#### config.yaml

```
databaseName: sharding_db

dataSources:
 ds_0:
  url: jdbc:mysql://***.***.***.***:****/test?serverTimezone=UTC&useSSL=false # Parameters can be adjusted appropriately
  username: test
  password:
  connectionTimeoutMilliseconds: 30000
  idleTimeoutMilliseconds: 60000
  maxLifetimeMilliseconds: 1800000
  maxPoolSize: 200 # The maximum ConnPool is set to ${the number of concurrencies in stress testing}, which is consistent with the number of concurrencies in stress testing to shield the impact of additional connections in the process of stress testing.
  minPoolSize: 200 # The minimum ConnPool is set to ${the number of concurrencies in stress testing}, which is consistent with the number of concurrencies in stress testing to shield the impact of connections initialization in the process of stress testing.

rules: []
```

### Test phase

#### Environment setup

```
sysbench oltp_read_write --mysql-host=${DB_IP} --mysql-port=${DB_PORT} --mysql-user=${USER} --mysql-password=${PASSWD} --mysql-db=test --tables=10 --table-size=1000000 --report-interval=10 --time=100 --threads=200  cleanup
sysbench oltp_read_write --mysql-host=${DB_IP} --mysql-port=${DB_PORT} --mysql-user=${USER} --mysql-password=${PASSWD} --mysql-db=test --tables=10 --table-size=1000000 --report-interval=10 --time=100 --threads=200  prepare
```

### Stress testing command

```
sysbench oltp_read_write --mysql-host=${DB/PROXY_IP} --mysql-port=${DB/PROXY_PORT} --mysql-user=${USER} --mysql-
password=${PASSWD} --mysql-db=test --tables=10 --table-size=1000000 --report-interval=10 --time=100 --threads=200  run
```

### Stress testing report analysis

```
sysbench 1.0.20 (using bundled LuaJIT 2.1.0-beta2)
Running the test with following options:
Number of threads: 200
Report intermediate results every 10 second(s)
Initializing random number generator from current time
Initializing worker threads...
Threads started!
# Report test results every 10 seconds, and the number of tps, reads per second, writes per second, and the total response time of
more than 95th percentile.
[ 10s ] thds: 200 tps: 11161.70 qps: 223453.06 (r/w/o: 156451.76/44658.51/22342.80) lat (ms,95%): 27.17 err/s: 0.00 reconn/s: 0.00
...
[ 120s ] thds: 200 tps: 11731.00 qps: 234638.36 (r/w/o: 164251.67/46924.69/23462.00) lat (ms,95%): 24.38 err/s: 0.00 reconn/s: 0.00
SQL statistics:
    queries performed:
        read:                    19560590               # number of reads
        write:                   5588740               # number of writes
        other:                    27943700              # number of other operations (COMMIT etc.)
        total:                   27943700              # the total number
    transactions:                1397185 (11638.59 per sec.)    # number of transactions (per second)
    queries:                  27943700 (232771.76 per sec.)  # number of statements executed (per second)
    ignored errors:              0     (0.00 per sec.)       # number of ignored errors (per second)
    reconnects:                  0     (0.00 per sec.)       # number of reconnections (per second)

General statistics:
    total time:                  120.0463s               # total time
    total number of events:          1397185               # toal number of transactions

Latency (ms):
         min:                        5.37            # minimum latency
         avg:                       17.13            # average latency
         max:                       109.75             # maximum latency
         95th percentile:                24.83               # average response time of over 95th percentile.
         sum:                  23999546.19

Threads fairness:
    events (avg/stddev):         6985.9250/34.74              # On average, 6985.9250 events were completed per thread, and the
standard deviation is 34.74
    execution time (avg/stddev):   119.9977/0.01              # The average time of each thread is 119.9977 seconds, and the
standard deviation is 0.01
```

### Noticeable features

1. CPU utilization ratio of the server where DBPlusEngine-Proxy resides. It is better to make full use of CPU.
2. I/O of the server disk where the DB resides. The lower the physical read value is, the better.
3. Network IO of the server involved in the stress testing.

### 10.5.2 BenchmarkSQL DBPlusEngine-Proxy Sharding Performance Test

#### Objective

BenchmarkSQL tool is used to test the sharding performance of DBPlusEngine-Proxy.

#### Method

DBPlusEngine-Proxy supports the TPC-C test through BenchmarkSQL 5.0. In addition to the content described in this document, BenchmarkSQL is operated according to the original document HOW-TO-RUN.txt.

#### Fine tuning to test tools

Unlike stand-alone database stress testing, distributed database solutions inevitably face trade-offs in functions. It is recommended to make the following adjustments when using BenchmarkSQL to carry out stress testing on DBPlusEngine-Proxy.

#### Remove the foreign key and extraHistID

Modify run/runDatabaseBuild.sh in the BenchmarkSQL directory at line 17.

Before modification:

```
AFTER_LOAD="indexCreates foreignKeys extraHistID buildFinish"
```

After modification:

```
AFTER_LOAD="indexCreates buildFinish"
```

#### Stress testing environment or parameter recommendations

**Note: None of the parameters mentioned in this section are absolute values and need to be adjusted based on actual test results.**

#### It is recommended to run DBPlusEngine using Java 17

DBPlusEngine can be compiled using Java 8.

When using Java 17, maximize the DBPlusEngine performance by default.

#### DBPlusEngine data sharding recommendations

The data sharding of BenchmarkSQL can use the warehouse id in each table as the sharding key.

One of the tables bmsql_item has no warehouse id and has a fixed data volume of 100,000 rows: - You can take i_id as a sharding key. However, the same Proxy connection may hold connections to multiple different data sources at the same time. - Or you can give up sharding and store it in a single data source. But a data source may be under great pressure. - Or you may choose range-based sharding for i_id, such as 1-50000 for data source 0 and 50001-100000 for data source 1.

BenchmarkSQL has the following SQL involving multiple tables:

```
SELECT c_discount, c_last, c_credit, w_tax
FROM bmsql_customer
  JOIN bmsql_warehouse ON (w_id = c_w_id)
WHERE c_w_id = ? AND c_d_id = ? AND c_id = ?
```

```
SELECT o_id, o_entry_d, o_carrier_id
FROM bmsql_oorder
WHERE o_w_id = ? AND o_d_id = ? AND o_c_id = ?
 AND o_id = (
    SELECT max(o_id)
    FROM bmsql_oorder
    WHERE o_w_id = ? AND o_d_id = ? AND o_c_id = ?
    )
```

If the warehouse id is used as the sharding key, the tables involved in the above SQL can be configured as bindingTable:

```
rules:
 - !SHARDING
  bindingTables:
   - bmsql_warehouse, bmsql_customer
   - bmsql_stock, bmsql_district, bmsql_order_line
```

For the data sharding configuration with warehouse id as the sharding key, refer to the appendix of this document.

### PostgreSQL JDBC URL parameter recommendations

Adjust the JDBC URL in the configuration file used by BenchmarkSQL, that is, the value of the parameter name conn: - Adding the parameter defaultRowFetchSize=50 may reduce the number of fetch for multi-row result sets. You need to increase or decrease the number according to actual test results. - Adding the parameter reWriteBatchedInserts=true may reduce the time spent on bulk inserts, such as preparing data or bulk inserts for the New Order business. Whether to enable the operation depends on actual test results.

props.pg file excerpt. It is suggested to change the parameter value of conn in line 3.

```
db=postgres
driver=org.postgresql.Driver
conn=jdbc:postgresql://localhost:5432/postgres?defaultRowFetchSize=50&reWriteBatchedInserts=true
user=benchmarksql
password=PWbmsql
```

### DBPlusEngine Proxy server.yaml parameter recommendations

The default value of proxy-backend-query-fetch-size is -1. Changing it to about 50 can minimize the number of fetch for multi-row result sets.

The default value of proxy-frontend-executor-size is CPU * 2 and can be reduced to about CPU * 0.5 based on actual test results. If NUMA is involved, set this parameter to the number of physical cores per CPU based on actual test results.

server.yaml file excerpt:

```
props:
 proxy-backend-query-fetch-size: 50
 # proxy-frontend-executor-size: 32 # 4*32C aarch64
 # proxy-frontend-executor-size: 12 # 2*12C24T x86
```

## Appendix

### BenchmarkSQL data sharding reference configuration

Adjust pool size according to the actual stress testing process.

```
databaseName: bmsql_sharding
dataSources:
 ds_0:
  url: jdbc:postgresql://db0.ip:5432/bmsql
  username: postgres
  password: postgres
  connectionTimeoutMilliseconds: 3000
  idleTimeoutMilliseconds: 60000
  maxLifetimeMilliseconds: 1800000
  maxPoolSize: 1000
  minPoolSize: 1000
 ds_1:
  url: jdbc:postgresql://db1.ip:5432/bmsql
  username: postgres
  password: postgres
  connectionTimeoutMilliseconds: 3000
  idleTimeoutMilliseconds: 60000
  maxLifetimeMilliseconds: 1800000
  maxPoolSize: 1000
  minPoolSize: 1000
 ds_2:
  url: jdbc:postgresql://db2.ip:5432/bmsql
  username: postgres
  password: postgres
  connectionTimeoutMilliseconds: 3000
  idleTimeoutMilliseconds: 60000
  maxLifetimeMilliseconds: 1800000
  maxPoolSize: 1000
  minPoolSize: 1000
 ds_3:
  url: jdbc:postgresql://db3.ip:5432/bmsql
  username: postgres
  password: postgres
  connectionTimeoutMilliseconds: 3000
  idleTimeoutMilliseconds: 60000
  maxLifetimeMilliseconds: 1800000
  maxPoolSize: 1000
  minPoolSize: 1000

rules:
 - !SHARDING
  bindingTables:
   - bmsql_warehouse, bmsql_customer
   - bmsql_stock, bmsql_district, bmsql_order_line
  defaultDatabaseStrategy:
   none:
  defaultTableStrategy:
   none:
  keyGenerators:
   snowflake:
    type: SNOWFLAKE
  tables:
   bmsql_config:
    actualDataNodes: ds_0.bmsql_config

   bmsql_warehouse:
    actualDataNodes: ds_${0..3}.bmsql_warehouse
    databaseStrategy:
```

```yaml
     standard:
      shardingColumn: w_id
      shardingAlgorithmName: mod_4

   bmsql_district:
    actualDataNodes: ds_${0..3}.bmsql_district
    databaseStrategy:
     standard:
      shardingColumn: d_w_id
      shardingAlgorithmName: mod_4

   bmsql_customer:
    actualDataNodes: ds_${0..3}.bmsql_customer
    databaseStrategy:
     standard:
      shardingColumn: c_w_id
      shardingAlgorithmName: mod_4

   bmsql_item:
    actualDataNodes: ds_${0..3}.bmsql_item
    databaseStrategy:
     standard:
      shardingColumn: i_id
      shardingAlgorithmName: mod_4

   bmsql_history:
    actualDataNodes: ds_${0..3}.bmsql_history
    databaseStrategy:
     standard:
      shardingColumn: h_w_id
      shardingAlgorithmName: mod_4

   bmsql_oorder:
    actualDataNodes: ds_${0..3}.bmsql_oorder
    databaseStrategy:
     standard:
      shardingColumn: o_w_id
      shardingAlgorithmName: mod_4

   bmsql_stock:
    actualDataNodes: ds_${0..3}.bmsql_stock
    databaseStrategy:
     standard:
      shardingColumn: s_w_id
      shardingAlgorithmName: mod_4

   bmsql_new_order:
    actualDataNodes: ds_${0..3}.bmsql_new_order
    databaseStrategy:
     standard:
      shardingColumn: no_w_id
      shardingAlgorithmName: mod_4

   bmsql_order_line:
    actualDataNodes: ds_${0..3}.bmsql_order_line
    databaseStrategy:
     standard:
      shardingColumn: ol_w_id
      shardingAlgorithmName: mod_4

  shardingAlgorithms:
   mod_4:
    type: MOD
    props:
      sharding-count: 4
```

## BenchmarkSQL 5.0 PostgreSQL statement list

### Create tables

```
create table bmsql_config (
  cfg_name    varchar(30) primary key,
  cfg_value   varchar(50)
);

create table bmsql_warehouse (
  w_id        integer   not null,
  w_ytd       decimal(12,2),
  w_tax       decimal(4,4),
  w_name      varchar(10),
  w_street_1  varchar(20),
  w_street_2  varchar(20),
  w_city      varchar(20),
  w_state     char(2),
  w_zip       char(9)
);

create table bmsql_district (
  d_w_id       integer       not null,
  d_id         integer       not null,
  d_ytd        decimal(12,2),
  d_tax        decimal(4,4),
  d_next_o_id  integer,
  d_name       varchar(10),
  d_street_1   varchar(20),
  d_street_2   varchar(20),
  d_city       varchar(20),
  d_state      char(2),
  d_zip        char(9)
);

create table bmsql_customer (
  c_w_id         integer        not null,
  c_d_id         integer        not null,
  c_id           integer        not null,
  c_discount     decimal(4,4),
  c_credit       char(2),
  c_last         varchar(16),
  c_first        varchar(16),
  c_credit_lim   decimal(12,2),
  c_balance      decimal(12,2),
  c_ytd_payment  decimal(12,2),
  c_payment_cnt  integer,
  c_delivery_cnt integer,
  c_street_1     varchar(20),
  c_street_2     varchar(20),
  c_city         varchar(20),
  c_state        char(2),
  c_zip          char(9),
  c_phone        char(16),
  c_since        timestamp,
  c_middle       char(2),
  c_data         varchar(500)
);
```

```
create sequence bmsql_hist_id_seq;

create table bmsql_history (
  hist_id  integer,
  h_c_id   integer,
  h_c_d_id integer,
  h_c_w_id integer,
  h_d_id   integer,
  h_w_id   integer,
  h_date   timestamp,
  h_amount decimal(6,2),
  h_data   varchar(24)
);

create table bmsql_new_order (
  no_w_id  integer   not null,
  no_d_id  integer   not null,
  no_o_id  integer   not null
);

create table bmsql_oorder (
  o_w_id       integer      not null,
  o_d_id       integer      not null,
  o_id         integer      not null,
  o_c_id       integer,
  o_carrier_id integer,
  o_ol_cnt     integer,
  o_all_local  integer,
  o_entry_d    timestamp
);

create table bmsql_order_line (
  ol_w_id         integer   not null,
  ol_d_id         integer   not null,
  ol_o_id         integer   not null,
  ol_number       integer   not null,
  ol_i_id         integer   not null,
  ol_delivery_d   timestamp,
  ol_amount       decimal(6,2),
  ol_supply_w_id  integer,
  ol_quantity     integer,
  ol_dist_info    char(24)
);

create table bmsql_item (
  i_id     integer      not null,
  i_name   varchar(24),
  i_price  decimal(5,2),
  i_data   varchar(50),
  i_im_id  integer
);

create table bmsql_stock (
  s_w_id       integer       not null,
  s_i_id       integer       not null,
  s_quantity   integer,
  s_ytd        integer,
  s_order_cnt  integer,
  s_remote_cnt integer,
  s_data       varchar(50),
  s_dist_01    char(24),
  s_dist_02    char(24),
  s_dist_03    char(24),
  s_dist_04    char(24),
  s_dist_05   char(24),
  s_dist_06   char(24),
  s_dist_07   char(24),
  s_dist_08   char(24),
  s_dist_09   char(24),
  s_dist_10   char(24)
);
```

### Create indexes

```
alter table bmsql_warehouse add constraint bmsql_warehouse_pkey
  primary key (w_id);

alter table bmsql_district add constraint bmsql_district_pkey
  primary key (d_w_id, d_id);

alter table bmsql_customer add constraint bmsql_customer_pkey
  primary key (c_w_id, c_d_id, c_id);

create index bmsql_customer_idx1
  on  bmsql_customer (c_w_id, c_d_id, c_last, c_first);

alter table bmsql_oorder add constraint bmsql_oorder_pkey
  primary key (o_w_id, o_d_id, o_id);

create unique index bmsql_oorder_idx1
  on  bmsql_oorder (o_w_id, o_d_id, o_carrier_id, o_id);

alter table bmsql_new_order add constraint bmsql_new_order_pkey
  primary key (no_w_id, no_d_id, no_o_id);

alter table bmsql_order_line add constraint bmsql_order_line_pkey
  primary key (ol_w_id, ol_d_id, ol_o_id, ol_number);

alter table bmsql_stock add constraint bmsql_stock_pkey
  primary key (s_w_id, s_i_id);

alter table bmsql_item add constraint bmsql_item_pkey
  primary key (i_id);
```

### New Order business

stmtNewOrderSelectWhseCust

```
UPDATE bmsql_district
    SET d_next_o_id = d_next_o_id + 1
    WHERE d_w_id = ? AND d_id = ?
```

stmtNewOrderSelectDist

```
SELECT d_tax, d_next_o_id
    FROM bmsql_district
    WHERE d_w_id = ? AND d_id = ?
    FOR UPDATE
```

stmtNewOrderUpdateDist

```
UPDATE bmsql_district
    SET d_next_o_id = d_next_o_id + 1
    WHERE d_w_id = ? AND d_id = ?
```

### stmtNewOrderInsertOrder

```
INSERT INTO bmsql_oorder (
   o_id, o_d_id, o_w_id, o_c_id, o_entry_d,
   o_ol_cnt, o_all_local)
VALUES (?, ?, ?, ?, ?, ?, ?)
```

### stmtNewOrderInsertNewOrder

```
INSERT INTO bmsql_new_order (
   no_o_id, no_d_id, no_w_id)
VALUES (?, ?, ?)
```

### stmtNewOrderSelectStock

```
SELECT s_quantity, s_data,
      s_dist_01, s_dist_02, s_dist_03, s_dist_04,
      s_dist_05, s_dist_06, s_dist_07, s_dist_08,
      s_dist_09, s_dist_10
   FROM bmsql_stock
   WHERE s_w_id = ? AND s_i_id = ?
   FOR UPDATE
```

### stmtNewOrderSelectItem

```
SELECT i_price, i_name, i_data
   FROM bmsql_item
   WHERE i_id = ?
```

### stmtNewOrderUpdateStock

```
UPDATE bmsql_stock
   SET s_quantity = ?, s_ytd = s_ytd + ?,
      s_order_cnt = s_order_cnt + 1,
      s_remote_cnt = s_remote_cnt + ?
   WHERE s_w_id = ? AND s_i_id = ?
```

### stmtNewOrderInsertOrderLine

```
INSERT INTO bmsql_order_line (
   ol_o_id, ol_d_id, ol_w_id, ol_number,
   ol_i_id, ol_supply_w_id, ol_quantity,
   ol_amount, ol_dist_info)
VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)
```

## Payment business

### stmtPaymentSelectWarehouse

```
SELECT w_name, w_street_1, w_street_2, w_city,
      w_state, w_zip
   FROM bmsql_warehouse
   WHERE w_id = ?
```

### stmtPaymentSelectDistrict

```
SELECT d_name, d_street_1, d_street_2, d_city,
      d_state, d_zip
   FROM bmsql_district
   WHERE d_w_id = ? AND d_id = ?
```

stmtPaymentSelectCustomerListByLast

```
SELECT c_id
  FROM bmsql_customer
  WHERE c_w_id = ? AND c_d_id = ? AND c_last = ?
  ORDER BY c_first
```

stmtPaymentSelectCustomer

```
SELECT c_first, c_middle, c_last, c_street_1, c_street_2,
    c_city, c_state, c_zip, c_phone, c_since, c_credit,
    c_credit_lim, c_discount, c_balance
  FROM bmsql_customer
  WHERE c_w_id = ? AND c_d_id = ? AND c_id = ?
  FOR UPDATE
```

stmtPaymentSelectCustomerData

```
SELECT c_data
  FROM bmsql_customer
  WHERE c_w_id = ? AND c_d_id = ? AND c_id = ?
```

stmtPaymentUpdateWarehouse

```
UPDATE bmsql_warehouse
  SET w_ytd = w_ytd + ?
  WHERE w_id = ?
```

stmtPaymentUpdateDistrict

```
UPDATE bmsql_district
  SET d_ytd = d_ytd + ?
  WHERE d_w_id = ? AND d_id = ?
```

stmtPaymentUpdateCustomer

```
UPDATE bmsql_customer
  SET c_balance = c_balance - ?,
    c_ytd_payment = c_ytd_payment + ?,
    c_payment_cnt = c_payment_cnt + 1
  WHERE c_w_id = ? AND c_d_id = ? AND c_id = ?
```

stmtPaymentUpdateCustomerWithData

```
UPDATE bmsql_customer
  SET c_balance = c_balance - ?,
    c_ytd_payment = c_ytd_payment + ?,
    c_payment_cnt = c_payment_cnt + 1,
    c_data = ?
  WHERE c_w_id = ? AND c_d_id = ? AND c_id = ?
```

stmtPaymentInsertHistory

```
INSERT INTO bmsql_history (
  h_c_id, h_c_d_id, h_c_w_id, h_d_id, h_w_id,
  h_date, h_amount, h_data)
VALUES (?, ?, ?, ?, ?, ?, ?, ?)
```

## Order Status business

### stmtOrderStatusSelectCustomerListByLast

```
SELECT c_id
  FROM bmsql_customer
  WHERE c_w_id = ? AND c_d_id = ? AND c_last = ?
  ORDER BY c_first
```

### stmtOrderStatusSelectCustomer

```
SELECT c_first, c_middle, c_last, c_balance
  FROM bmsql_customer
  WHERE c_w_id = ? AND c_d_id = ? AND c_id = ?
```

### stmtOrderStatusSelectLastOrder

```
SELECT o_id, o_entry_d, o_carrier_id
  FROM bmsql_oorder
  WHERE o_w_id = ? AND o_d_id = ? AND o_c_id = ?
   AND o_id = (
     SELECT max(o_id)
       FROM bmsql_oorder
       WHERE o_w_id = ? AND o_d_id = ? AND o_c_id = ?
     )
```

### stmtOrderStatusSelectOrderLine

```
SELECT ol_i_id, ol_supply_w_id, ol_quantity,
    ol_amount, ol_delivery_d
  FROM bmsql_order_line
  WHERE ol_w_id = ? AND ol_d_id = ? AND ol_o_id = ?
  ORDER BY ol_w_id, ol_d_id, ol_o_id, ol_number
```

## Stock level business

### stmtStockLevelSelectLow

```
SELECT count(*) AS low_stock FROM (
  SELECT s_w_id, s_i_id, s_quantity
    FROM bmsql_stock
    WHERE s_w_id = ? AND s_quantity < ? AND s_i_id IN (
      SELECT ol_i_id
        FROM bmsql_district
        JOIN bmsql_order_line ON ol_w_id = d_w_id
         AND ol_d_id = d_id
         AND ol_o_id >= d_next_o_id - 20
         AND ol_o_id < d_next_o_id
        WHERE d_w_id = ? AND d_id = ?
    )
  ) AS L
```

## Delivery BG business

### stmtDeliveryBGSelectOldestNewOrder

```
SELECT no_o_id
  FROM bmsql_new_order
  WHERE no_w_id = ? AND no_d_id = ?
  ORDER BY no_o_id ASC
```

### stmtDeliveryBGDeleteOldestNewOrder

```
DELETE FROM bmsql_new_order
  WHERE no_w_id = ? AND no_d_id = ? AND no_o_id = ?
```

### stmtDeliveryBGSelectOrder

```
SELECT o_c_id
  FROM bmsql_oorder
  WHERE o_w_id = ? AND o_d_id = ? AND o_id = ?
```

### stmtDeliveryBGUpdateOrder

```
UPDATE bmsql_oorder
  SET o_carrier_id = ?
  WHERE o_w_id = ? AND o_d_id = ? AND o_id = ?
```

### stmtDeliveryBGSelectSumOLAmount

```
SELECT sum(ol_amount) AS sum_ol_amount
  FROM bmsql_order_line
  WHERE ol_w_id = ? AND ol_d_id = ? AND ol_o_id = ?
```

### stmtDeliveryBGUpdateOrderLine

```
UPDATE bmsql_order_line
SET ol_delivery_d = ?
WHERE ol_w_id = ? AND ol_d_id = ? AND ol_o_id = ?
```

### stmtDeliveryBGUpdateCustomer

```
UPDATE bmsql_customer
  SET c_balance = c_balance + ?,
    c_delivery_cnt = c_delivery_cnt + 1
  WHERE c_w_id = ? AND c_d_id = ? AND c_id = ?
```

## 10.6 Module Test

Provides test engine with each complex modules.

### 10.6.1 SQL Parser Test

#### Prepare Data

Unlike the Integration test, the SQL parse test does not need a specific database environment, just define the sql to parse, and the assert data:

#### SQL Data

As mentioned sql-case-id in Integration test，test-case-id could be shared in different module to test, and the file is at shardingsphere-sql-parser/shardingsphere-sql-parser-test/src/main/resources/sql/supported/${SQL-TYPE}/*.xml

#### Assert Data

The assert data is at shardingsphere-sql-parser/shardingsphere-sql-parser-test/src/main/resources/case/${SQL-TYPE}/*.xml in that xml file, it could assert against the table name, token or sql condition and so on. For example:

```
<parser-result-sets>
  <parser-result sql-case-id="insert_with_multiple_values">
    <tables>
      <table name="t_order" />
    </tables>
    <tokens>
      <table-token start-index="12" table-name="t_order" length="7" />
    </tokens>
    <sharding-conditions>
      <and-condition>
        <condition column-name="order_id" table-name="t_order" operator="EQUAL">
          <value literal="1" type="int" />
        </condition>
        <condition column-name="user_id" table-name="t_order" operator="EQUAL">
          <value literal="1" type="int" />
        </condition>
      </and-condition>
      <and-condition>
        <condition column-name="order_id" table-name="t_order" operator="EQUAL">
          <value literal="2" type="int" />
        </condition>
        <condition column-name="user_id" table-name="t_order" operator="EQUAL">
          <value literal="2" type="int" />
        </condition>
      </and-condition>
    </sharding-conditions>
  </parser-result>
</parser-result-sets>
```

When these configs are ready, launch the test engine in shardingsphere-sql-parser/shardingsphere-sql-parser-test to test SQL parse.

## 10.6.2 SQL Rewrite Test

### Target

Facing logic databases and tables cannot be executed directly in actual databases. SQL rewrite is used to rewrite logic SQL into rightly executable ones in actual databases, including two parts, correctness rewrite and optimization rewrite. Rewrite tests are for these targets.

### Test

The rewrite tests are in the test folder under sharding-core/sharding-core-rewrite. Followings are the main part for rewrite tests:

- test engine
- environment configuration
- assert data

Test engine is the entrance of rewrite tests, just like other test engines, through Junit Parameterized, read every and each data in the xml file under the target test type in test\resources, and then assert by the engine one by one

Environment configuration is the yaml file under test type under test\resources\yaml. The configuration file contains dataSources，shardingRule，encryptRule and other info. for example:

```
dataSources:
  db: !!com.zaxxer.hikari.HikariDataSource
    driverClassName: org.h2.Driver
    jdbcUrl: jdbc:h2:mem:db;DB_CLOSE_DELAY=-1;DATABASE_TO_UPPER=false;MODE=MYSQL
    username: sa
    password:

## sharding Rules
rules:
- !SHARDING
  tables:
    t_account:
      actualDataNodes: db.t_account_${0..1}
      tableStrategy:
        standard:
          shardingColumn: account_id
          shardingAlgorithmName: account_table_inline
      keyGenerateStrategy:
        column: account_id
        keyGeneratorName: snowflake
    t_account_detail:
      actualDataNodes: db.t_account_detail_${0..1}
      tableStrategy:
        standard:
          shardingColumn: order_id
          shardingAlgorithmName: account_detail_table_inline
  bindingTables:
    - t_account, t_account_detail
  shardingAlgorithms:
    account_table_inline:
      type: INLINE
      props:
        algorithm-expression: t_account_${account_id % 2}
    account_detail_table_inline:
      type: INLINE
      props:
        algorithm-expression: t_account_detail_${account_id % 2}
  keyGenerators:
```

```
  snowflake:
    type: SNOWFLAKE
```

Assert data are in the xml under test type in test\resources. In the xml file, yaml-rule means the environment configuration file path, input contains the target SQL and parameters, output contains the expected SQL and parameters. The db-type described the type for SQL parse, default is SQL92. For example:

```
<rewrite-assertions yaml-rule="yaml/sharding/sharding-rule.yaml">
  <!-- to change SQL parse type, change db-type -->
  <rewrite-assertion id="create_index_for_mysql" db-type="MySQL">
    <input sql="CREATE INDEX index_name ON t_account ('status')" />
    <output sql="CREATE INDEX index_name ON t_account_0 ('status')" />
    <output sql="CREATE INDEX index_name ON t_account_1 ('status')" />
  </rewrite-assertion>
</rewrite-assertions>
```

After setting up the assert data and environment configuration, rewrite test engine will assert the corresponding SQL without any Java code modification.

# 11 Reference

This chapter contains a section of technical implementation and test process with DBPlusEngine, which provide the reference with users and developers.

## 11.1 Management

### 11.1.1 Data Structure in Registry Center

Under defined namespace, rules, props and metadata nodes persist in YAML, and modifying nodes can dynamically refresh configurations. nodes node persist the runtime node of database access object, to distinguish different database access instances.

```
namespace
   ├────rules                        # Global rule configuration
   ├────props                        # Properties configuration
   ├────metadata                     # Metadata configuration
   ├    ├────${schema_1}                # Schema name 1
   ├    ├    ├────dataSources             # Datasource configuration
   ├    ├    ├────rules              # Rule configuration
   ├    ├    ├────tables              # Table configuration
   ├    ├    ├    ├────t_1
   ├    ├    ├    ├────t_2
   ├    ├    ├────views              # View configuration
   ├    ├    ├    ├────v_1
   ├    ├    ├    ├────v_2
   ├    ├────${schema_2}                # Schema name 2
   ├    ├    ├────dataSources             # Datasource configuration
   ├    ├    ├────rules              # Rule configuration
   ├    ├    ├────tables              # Table configuration
   ├────nodes
   ├    ├────compute_nodes
   ├    ├    ├────online
   ├    ├    ├    ├────proxy
   ├    ├    ├    ├    ├────${your_instance_ip_a}@${your_instance_port_x}
   ├    ├    ├    ├    ├────${your_instance_ip_b}@${your_instance_port_y}
   ├    ├    ├    ├    ├────....
   ├    ├    ├    ├────jdbc
   ├    ├    ├    ├    ├────${your_instance_ip_a}@${your_instance_pid_x}
   ├    ├    ├    ├    ├────${your_instance_ip_b}@${your_instance_pid_y}
   ├    ├    ├    ├    ├────....
   ├    ├    ├────attributies
   ├    ├    ├    ├────${your_instance_ip_a}@${your_instance_port_x}
```

```
 ├   ├    ├    ├    ├─────status
 ├   ├    ├    ├    ├─────label
 ├   ├    ├    ├─────${your_instance_ip_b}@${your_instance_pid_y}
 ├   ├    ├    ├    ├─────status
 ├   ├    ├    ├─────....
 ├   ├─────storage_nodes
 ├   ├    ├─────disable
 ├   ├    ├     ├─────${schema_1.ds_0}
 ├   ├    ├     ├─────${schema_1.ds_1}
 ├   ├    ├     ├─────....
 ├   ├    ├─────primary
 ├   ├    ├     ├─────${schema_2.ds_0}
 ├   ├    ├     ├─────${schema_2.ds_1}
 ├   ├    ├     ├─────....
```

### /rules

Global rule configuration, which can include transaction configuration, SQL parser configuration, etc.

```yaml
- !TRANSACTION
  defaultType: XA
  providerType: Atomikos
- !SQL_PARSER
  sqlCommentParseEnabled: true
```

### /props

Properties configuration. Please refer to Configuration Manual for more details.

```yaml
kernel-executor-size: 20
sql-show: true
```

### /metadata/${schemaName}/dataSources

A collection of multiple database connection pools, whose properties (e.g. DBCP, C3P0, Druid and HikariCP) are configured by users themselves.

```yaml
ds_0:
  initializationFailTimeout: 1
  validationTimeout: 5000
  maxLifetime: 1800000
  leakDetectionThreshold: 0
  minimumIdle: 1
  password: root
  idleTimeout: 60000
  jdbcUrl: jdbc:mysql://127.0.0.1:3306/ds_0?serverTimezone=UTC&useSSL=false
  dataSourceClassName: com.zaxxer.hikari.HikariDataSource
  maximumPoolSize: 50
  connectionTimeout: 30000
  username: root
  poolName: HikariPool-1
ds_1:
  initializationFailTimeout: 1
  validationTimeout: 5000
  maxLifetime: 1800000
  leakDetectionThreshold: 0
  minimumIdle: 1
  password: root
  idleTimeout: 60000
  jdbcUrl: jdbc:mysql://127.0.0.1:3306/ds_1?serverTimezone=UTC&useSSL=false
  dataSourceClassName: com.zaxxer.hikari.HikariDataSource
  maximumPoolSize: 50
  connectionTimeout: 30000
  username: root
  poolName: HikariPool-2
```

### /metadata/${schemaName}/rules

Rule configurations, including sharding, read/write splitting, data encryption, shadow DB configurations.

```
- !SHARDING
  xxx

- !READWRITE_SPLITTING
  xxx

- !ENCRYPT
  xxx
```

### /metadata/${schemaName}/tables

Use separate node storage for each table, dynamic modification of metadata content is not supported currently.

```
name: t_order                    # Table name
columns:                        # Columns
  id:                         # Column name
    caseSensitive: false
    dataType: 0
    generated: false
    name: id
    primaryKey: trues
  order_id:
    caseSensitive: false
    dataType: 0
    generated: false
    name: order_id
    primaryKey: false
indexs:                        # Index
  t_user_order_id_index:           # Index name
    name: t_user_order_id_index
```

### /nodes/compute_nodes

It includes running instance information of database access object, with sub-nodes as the identifiers of currently running instance, which consist of IP and PORT.

Those identifiers are temporary nodes, which are registered when instances are online and cleared when instances are offline. The registry center monitors the change of those nodes to govern the database access of running instances and other things.

### /nodes/storage_nodes

It is able to orchestrate replica database, delete or disable data dynamically.

## 11.2 Sharding

The major sharding processes of all the three DBPlusEngine products are identical. According to whether query optimization is performed, they can be divided into standard kernel process and federation executor engine process. The standard kernel process consists of SQL Parse => SQL Route => SQL Rewrite => SQL Execute => Result Merge, which is used to process SQL execution in standard sharding scenarios. The federation executor engine process consists of SQL Parse => Logical Plan Optimize => Physical Plan Optimize => Plan Execute => Standard Kernel Process. The federation executor engine perform logical plan optimization and physical plan optimization. In the optimization execution phase, it relies on the standard kernel process to route, rewrite, execute, and merge the optimized logical SQL.

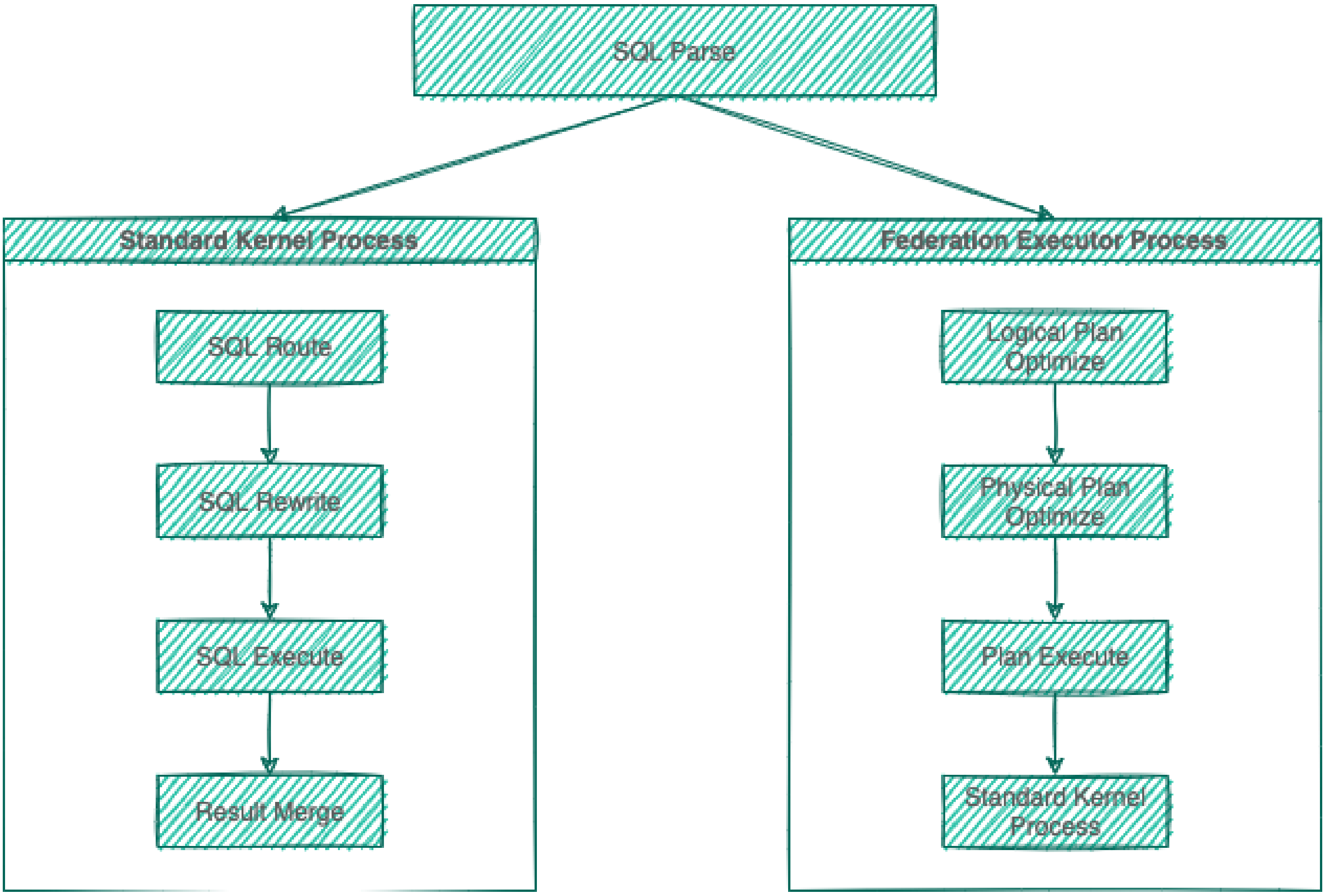


Fig. 1: Sharding Architecture Diagram

### 11.2.1 SQL Parsing

It is divided into lexical parsing and syntactic parsing. The lexical parser will split SQL into inseparable words, and then the syntactic parser will analyze SQL and extract the parsing context, which can include tables, options, ordering items, grouping items, aggregation functions, pagination information, query conditions and placeholders that may be revised.

### 11.2.2 SQL Route

It is the sharding strategy that matches users' configurations according to the parsing context and the route path can be generated. It supports sharding route and broadcast route currently.

### 11.2.3 SQL Rewrite

It rewrites SQL as statement that can be rightly executed in the real database, and can be divided into correctness rewrite and optimization rewrite.

### 11.2.4 SQL Execution

Through multi-thread executor, it executes asynchronously.

### 11.2.5 Result Merger

It merges multiple execution result sets to output through unified JDBC interface. Result merger includes methods as stream merger, memory merger and addition merger using decorator merger.

### 11.2.6 Query Optimization

Supported by federation executor engine(under development), optimization is performed on complex query such as join query and subquery. It also supports distributed query across multiple database instances. It uses relational algebra internally to optimize query plan, and then get query result through the best query plan.

### 11.2.7 Parse Engine

Compared to other programming languages, SQL is relatively simple, but it is still a complete set of programming language, so there is no essential difference between parsing SQL grammar and parsing other languages (Java, C and Go, etc.).

#### Abstract Syntax Tree

The parsing process can be divided into lexical parsing and syntactic parsing. Lexical parser is used to divide SQL into indivisible atomic signs, i.e., Token. According to the dictionary provided by different database dialect, it is categorized into keyword, expression, literal value and operator. SQL is then converted into abstract syntax tree by syntactic parser.

For example, the following SQL:

```sql
SELECT id, name FROM t_user WHERE status = 'ACTIVE' AND age > 18
```

Its parsing AST (Abstract Syntax Tree) is this:

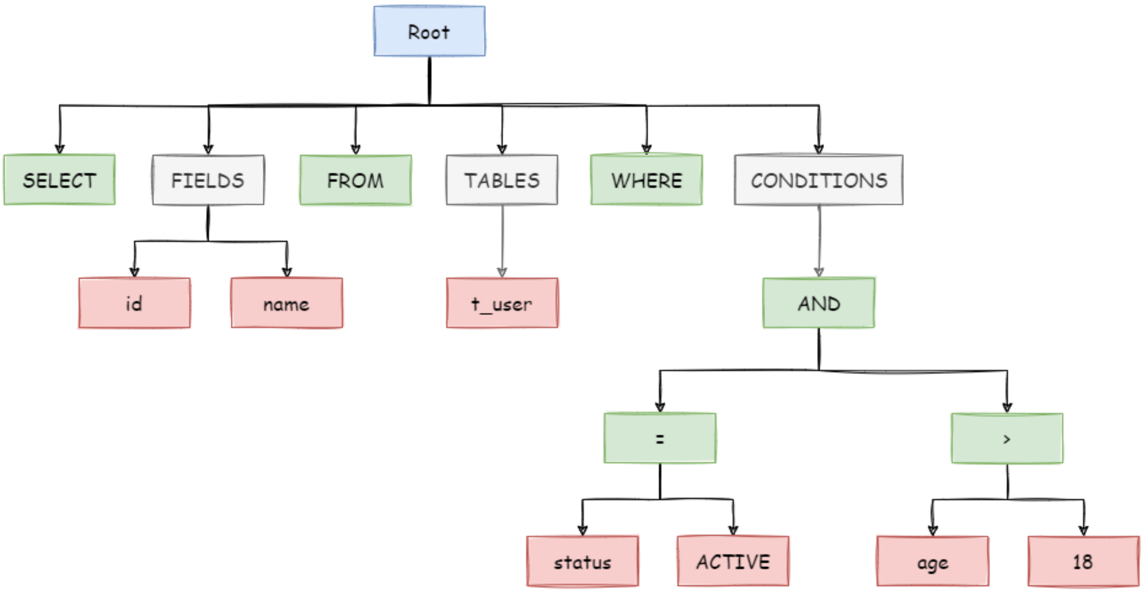


Fig. 2: SQL AST

To better understand, the Token of keywords in abstract syntax tree is shown in green; that of variables is shown in red; what ' s to be further divided is shown in grey.

At last, through traversing the abstract syntax tree, the context needed by sharding is extracted and the place that may need to be rewritten is also marked out. Parsing context for the use of sharding includes select items, table information, sharding conditions, auto-increment primary key information, Order By information, Group By information, and pagination information (Limit, Rownum and Top). One-time SQL parsing process is irreversible, each Token is parsed according to the original order of SQL in a high performance. Considering similarities and differences between SQL of all kinds of database dialect, SQL dialect dictionaries of different types of databases are provided in the parsing module.

## SQL Parser

### Features

- Independent SQL parsing engine
- The syntax rules can be easily expanded and modified (using ANTLR)
- Support multiple dialects

| DB | Status |
|---|---|
| MySQL | supported |
| PostgreSQL | supported |
| SQLServer | supported |
| Oracle | supported |
| SQL92 | supported |
| openGauss | supported |

- SQL format (developing)
- SQL parameterize (developing)

### API Usage

Maven config

```
<dependency>
   <groupId>org.apache.shardingsphere</groupId>
   <artifactId>shardingsphere-sql-parser-engine</artifactId>
   <version>${project.version}</version>
</dependency>
// According to the needs, introduce the parsing module of the specified dialect (take MySQL as an example), you can add all the supported dialects, or just what you need
<dependency>
   <groupId>org.apache.shardingsphere</groupId>
   <artifactId>shardingsphere-sql-parser-mysql</artifactId>
   <version>${project.version}</version>
</dependency>
```

demo:

- Get AST

```
/**
 * databaseType type:String values: MySQL, Oracle, PostgreSQL, SQL92, SQLServer, openGauss
 * sql type:String SQL to be parsed
 * useCache type:boolean whether use cache
 * @return parse context
 */
ParseContext parseContext = new SQLParserEngine(databaseType).parse(sql, useCache)
```

- GET SQLStatement

```
/**
 * databaseType type:String values: MySQL, Oracle, PostgreSQL, SQL92, SQLServer, openGauss
 * useCache type:boolean whether use cache
 * @return SQLStatement
 */
ParseContext parseContext = new SQLParserEngine(databaseType).parse(sql, useCache);
SQLVisitorEngine sqlVisitorEngine = new SQLVisitorEngine(databaseType, "STATEMENT");
SQLStatement sqlStatement = sqlVisitorEngine.visit(parseContext);
```

- SQL Format

```
/**
 * databaseType type:String values MySQL
 * useCache type:boolean whether use cache
 * @return String
 */
ParseContext parseContext = new SQLParserEngine(databaseType).parse(sql, useCache);
SQLVisitorEngine sqlVisitorEngine = new SQLVisitorEngine(databaseType, "FORMAT", new Properties());
String formatedSql = sqlVisitorEngine.visit(parseContext);
```

example:

### 11.2.8 Route Engine

It refers to the sharding strategy that matches databases and tables according to the parsing context and generates route path. SQL with sharding keys can be divided into single-sharding route (equal mark as the operator of sharding key), multiple-sharding route (IN as the operator of sharding key) and range sharding route (BETWEEN as the operator of sharding key). SQL without sharding key adopts broadcast route.

Sharding strategies can usually be set in the database or by users. Strategies built in the database are relatively simple and can generally be divided into last number modulo, hash, range, tag, time and so on. More flexible, sharding strategies set by users can be customized according to their needs. Together with automatic data migration, database middle layer can automatically shard and balance the data without users paying attention to sharding strategies, and thereby the distributed database can have the elastic scaling-out ability.

#### Sharding Route

It is used in the situation to route according to the sharding key, and can be sub-divided into 3 types, direct route, standard route and Cartesian product route.

#### Direct Route

The conditions for direct route are relatively strict. It requires to shard through Hint (use HintAPI to appoint the route to databases and tables directly). On the premise of having database sharding but not table sharding, SQL parsing and the following result merging can be avoided. Therefore, with the highest compatibility, it can execute any SQL in complex situations, including sub-queries, self-defined functions. Direct route can also be used in the situation where sharding keys are not in SQL. For example, set sharding key as 3.

```
hintManager.setDatabaseShardingValue(3);
```

If the routing algorithm is value % 2, when a logical database t_order corresponds to two physical databasest_order_0 and t_order_1, the SQL will be executed on t_order_1 after routing. The following is a sample code using the API.

```
String sql = "SELECT * FROM t_order";
try (
     HintManager hintManager = HintManager.getInstance();
     Connection conn = dataSource.getConnection();
     PreparedStatement pstmt = conn.prepareStatement(sql)) {
  hintManager.setDatabaseShardingValue(3);
  try (ResultSet rs = pstmt.executeQuery()) {
     while (rs.next()) {
        //...
     }
  }
}
```

### Standard Route

Standard route is DB Plus Engine’ s most recommended sharding method. Its application range is the SQL that does not include joint query or only includes joint query between binding tables. When the sharding operator is equal mark, the route result will fall into a single database (table); when sharding operators are BETWEEN or IN, the route result will not necessarily fall into the only database (table). So one logic SQL can finally be split into multiple real SQL to execute. For example, if sharding is according to the odd number or even number of order_id, a single table query SQL is as the following:

```
SELECT * FROM t_order WHERE order_id IN (1, 2);
```

The route result will be:

```
SELECT * FROM t_order_0 WHERE order_id IN (1, 2);
SELECT * FROM t_order_1 WHERE order_id IN (1, 2);
```

The complexity and performance of the joint query are comparable with those of single-table query. For instance, if a joint query SQL that contains binding tables is as this:

```
SELECT * FROM t_order o JOIN t_order_item i ON o.order_id=i.order_id  WHERE order_id IN (1, 2);
```

Then, the route result will be:

```
SELECT * FROM t_order_0 o JOIN t_order_item_0 i ON o.order_id=i.order_id  WHERE order_id IN (1, 2);
SELECT * FROM t_order_1 o JOIN t_order_item_1 i ON o.order_id=i.order_id  WHERE order_id IN (1, 2);
```

It can be seen that, the number of divided SQL is the same as the number of single tables.

### Cartesian Route

Cartesian route has the most complex situation, it cannot locate sharding rules according to the binding table relationship, so the joint query between non-binding tables needs to be split into Cartesian product combination to execute. If SQL in the last case is not configured with binding table relationship, the route result will be:

```
SELECT * FROM t_order_0 o JOIN t_order_item_0 i ON o.order_id=i.order_id  WHERE order_id IN (1, 2);
SELECT * FROM t_order_0 o JOIN t_order_item_1 i ON o.order_id=i.order_id  WHERE order_id IN (1, 2);
SELECT * FROM t_order_1 o JOIN t_order_item_0 i ON o.order_id=i.order_id  WHERE order_id IN (1, 2);
SELECT * FROM t_order_1 o JOIN t_order_item_1 i ON o.order_id=i.order_id  WHERE order_id IN (1, 2);
```

Cartesian product route has a relatively low performance, so it should be careful to use.

### Broadcast Route

For SQL without sharding key, broadcast route is used. According to SQL types, it can be divided into five types, schema & table route, database schema route, database instance route, unicast route and ignore route.

### Schema & Table Route

Schema & table route is used to deal with all the operations of physical tables related to its logic table, including DQL and DML without sharding key and DDL, etc. For example.

```
SELECT * FROM t_order WHERE good_prority IN (1, 10);
```

It will traverse all the tables in all the databases, match the logical table and the physical table name one by one and execute them if succeeded. After routing, they are:

```
SELECT * FROM t_order_0 WHERE good_prority IN (1, 10);
SELECT * FROM t_order_1 WHERE good_prority IN (1, 10);
SELECT * FROM t_order_2 WHERE good_prority IN (1, 10);
SELECT * FROM t_order_3 WHERE good_prority IN (1, 10);
```

### Database Schema Route

Database schema route is used to deal with database operations, including the SET database management order used to set the database and transaction control statement as TCL. In this case, all physical databases matched with the name are traversed according to logical database name, and the command is executed in the physical database. For example:

```
SET autocommit=0;
```

If this command is executed in t_order, t_order will have 2 physical databases. And it will actually be executed in both t_order_0 and t_order_1.

### Database Instance Route

Database instance route is used in DCL operation, whose authorization statement aims at database instances. No matter how many schemas are included in one instance, each one of them can only be executed once. For example:

```
CREATE USER customer@127.0.0.1 identified BY '123';
```

This command will be executed in all the physical database instances to ensure customer users have access to each instance.

### Unicast Route

Unicast route is used in the scenario of acquiring the information from some certain physical table. It only requires to acquire data from any physical table in any database. For example:

```
DESCRIBE t_order;
```

The descriptions of the two physical tables, t_order_0 and t_order_1 of t_order have the same structure, so this command is executed once on any physical table.

### Ignore Route

Ignore route is used to block the operation of SQL to the database. For example:

```
USE order_db;
```

This command will not be executed in physical database. Because DB Plus Engine uses logic Schema, there is no need to send the Schema shift order to the database.

The overall structure of route engine is as the following:

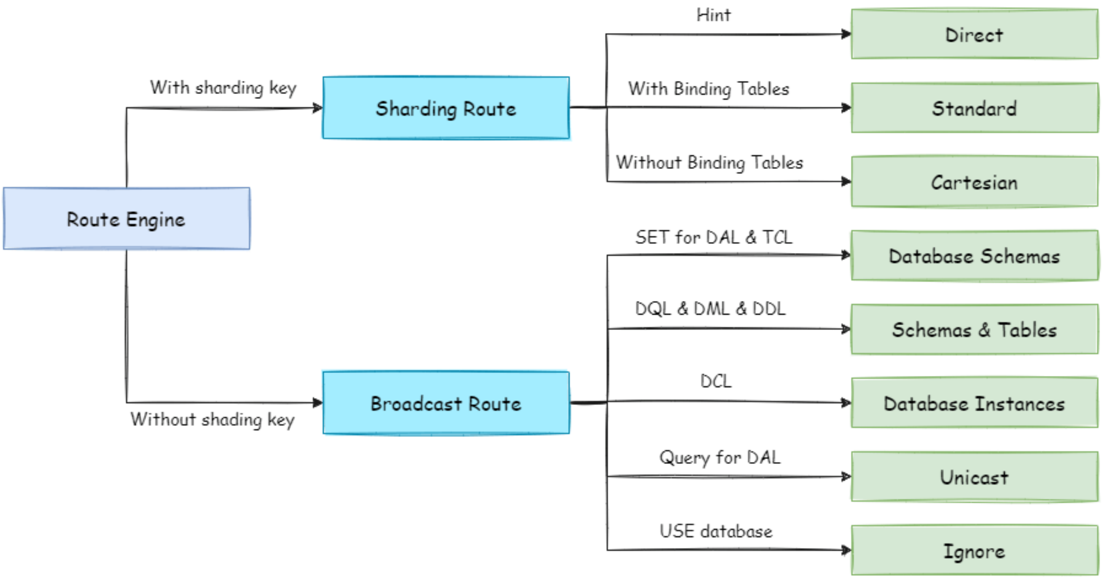


Fig. 3: Route Engine

### 11.2.9 Rewrite Engine

The SQL written by engineers facing logic databases and tables cannot be executed directly in actual databases. SQL rewrite is used to rewrite logic SQL into rightly executable ones in actual databases, including two parts, correctness rewrite and optimization rewrite.

### Correctness Rewrite

In situations with sharding tables, it requires to rewrite logic table names in sharding settings into actual table names acquired after routing. Database sharding does not require to rewrite table names. In addition to that, there are also column derivation, pagination information revision and other content.

### Identifier Rewrite

Identifiers that need to be rewritten include table name, index name and schema name. Table name rewrite refers to the process to locate the position of logic tables in the original SQL and rewrite it as the physical table. Table name rewrite is one typical situation that requires to parse SQL. From a most plain case, if the logic SQL is as follow:

```
SELECT order_id FROM t_order WHERE order_id=1;
```

If the SQL is configured with sharding key order_id=1, it will be routed to Sharding Table 1. Then, the SQL after rewrite should be:

```
SELECT order_id FROM t_order_1 WHERE order_id=1;
```

In this most simple kind of SQL, whether parsing SQL to abstract syntax tree seems unimportant, SQL can be rewritten only by searching for and substituting characters. But in the following situation, it is unable to rewrite SQL rightly merely by searching for and substituting characters:

```
SELECT order_id FROM t_order WHERE order_id=1 AND remarks=' t_order xxx';
```

The SQL rightly rewritten is supposed to be:

```
SELECT order_id FROM t_order_1 WHERE order_id=1 AND remarks=' t_order xxx';
```

Rather than:

```
SELECT order_id FROM t_order_1 WHERE order_id=1 AND remarks=' t_order_1 xxx';
```

Because there may be similar characters besides the table name, the simple character substitute method cannot be used to rewrite SQL. Here is another more complex SQL rewrite situation:

```
SELECT t_order.order_id FROM t_order WHERE t_order.order_id=1 AND remarks=' t_order xxx';
```

The SQL above takes table name as the identifier of the field, so it should also be revised when SQL is rewritten:

```
SELECT t_order_1.order_id FROM t_order_1 WHERE t_order_1.order_id=1 AND remarks=' t_order xxx';
```

But if there is another table name defined in SQL, it is not necessary to revise that, even though that name is the same as the table name. For example:

```
SELECT t_order.order_id FROM t_order AS t_order WHERE t_order.order_id=1 AND remarks=' t_order xxx';
```

SQL rewrite only requires to revise its table name:

```
SELECT t_order.order_id FROM t_order_1 AS t_order WHERE t_order.order_id=1 AND remarks=' t_order xxx';
```

Index name is another identifier that can be rewritten. In some databases (such as MySQL/SQLServer), the index is created according to the table dimension, and its names in different tables can repeat. In some other databases (such as PostgreSQL/Oracle), however, the index is created according to the database dimension, index names in different tables are required to be one and the only.

In DB Plus Engine, schema management method is similar to that of the table. It uses logic schema to manage a set of data sources, so it requires to replace the logic schema written by users in SQL with physical database schema.

DB Plus Engine only supports to use schema in database management statements but not in DQL and DML statements, for example:

```sql
SHOW COLUMNS FROM t_order FROM order_ds;
```

Schema rewrite refers to rewriting logic schema as a right and real schema found arbitrarily with unicast route.

### Column Derivation

Column derivation in query statements usually results from two situations. First, DB Plus Engine needs to acquire the corresponding data when merging results, but it is not returned through the query SQL. This kind of situation aims mainly at GROUP BY and ORDER BY. Result merger requires sorting and ranking according to items of GROUP BY and ORDER BY field. But if sorting and ranking items are not included in the original SQL, it should be rewritten. Look at the situation where the original SQL has the information required by result merger:

```sql
SELECT order_id, user_id FROM t_order ORDER BY user_id;
```

Since user_id is used in ranking, the result merger needs the data able to acquire user_id. The SQL above is able to acquire user_id data, so there is no need to add columns.

If the selected item does not contain the column required by result merger, it will need to add column, as the following SQL:

```sql
SELECT order_id FROM t_order ORDER BY user_id;
```

Since the original SQL does not contain user_id needed by result merger, the SQL needs to be rewritten by adding columns, and after that, it will be:

```sql
SELECT order_id, user_id AS ORDER_BY_DERIVED_0 FROM t_order ORDER BY user_id;
```

What’ s to be mentioned, column derivation will only add the missing column rather than all of them; the SQL that includes * in SELECT will also selectively add columns according to the meta-data information of tables. Here is a relatively complex SQL column derivation case:

```sql
SELECT o.* FROM t_order o, t_order_item i WHERE o.order_id=i.order_id ORDER BY user_id, order_item_id;
```

Suppose only t_order_item table contains order_item_id column, according to the meta-data information of tables, the user_id in sorting item exists in table t_order as merging result, but order_item_id does not exist in t_order, so it needs to add columns. The SQL after that will be:

```sql
SELECT o.*, order_item_id AS ORDER_BY_DERIVED_0 FROM t_order o, t_order_item i WHERE o.order_id=i.order_id ORDER BY
user_id, order_item_id;
```

Another situation of column derivation is using AVG aggregation function. In distributed situations, it is not right to calculate the average value with avg1 + avg2 + avg3 / 3, and it should be rewritten as (sum1 + sum2 + sum3) / (count1 + count2 + count3). This requires to rewrite the SQL that contains AVG as SUM and COUNT and recalculate the average value in result merger. Such as the following SQL:

```sql
SELECT AVG(price) FROM t_order WHERE user_id=1;
```

Should be rewritten as:

```sql
SELECT COUNT(price) AS AVG_DERIVED_COUNT_0, SUM(price) AS AVG_DERIVED_SUM_0 FROM t_order WHERE user_id=1;
```

Then it can calculate the right average value through result merger.

The last kind of column derivation is in SQL with INSERT. With database auto-increment key, there is no need to fill in primary key field. But database auto-increment key cannot satisfy the requirement of only one primary key being in the distributed situation. So DB Plus Engine provides a distributed auto-increment key generation strategy, enabling users to replace the current auto-increment key invisibly with a distributed one without changing existing codes through column derivation. Distributed auto-increment key generation strategy will be expounded in the following part, here we only explain the content related to SQL rewrite. For example, if the primary key of t_order is order_id, and the original SQL is:

```
INSERT INTO t_order (`field1`, `field2`) VALUES (10, 1);
```

It can be seen that the SQL above does not include an auto-increment key, which will be filled by the database itself. After DB Plus Engine set an auto-increment key, the SQL will be rewritten as:

```
INSERT INTO t_order (`field1`, `field2`, order_id) VALUES (10, 1, xxxxx);
```

Rewritten SQL will add auto-increment key name and its value generated automatically in the last part of INSERT FIELD and INSERT VALUE. xxxxx in the SQL above stands for the latter one.

If INSERT SQL does not contain the column name of the table, DB Plus Engine can also automatically generate auto-increment key by comparing the number of parameter and column in the table meta-information. For example, the original SQL is:

```
INSERT INTO t_order VALUES (10, 1);
```

The rewritten SQL only needs to add an auto-increment key in the column where the primary key is:

```
INSERT INTO t_order VALUES (xxxxx, 10, 1);
```

When auto-increment key derives column, if the user writes SQL with placeholder, he only needs to rewrite parameter list but not SQL itself.

### Pagination Revision

The scenarios of acquiring pagination data from multiple databases is different from that of one single database. If every 10 pieces of data are taken as one page, the user wants to take the second page of data. It is not right to take, acquire LIMIT 10, 10 under sharding situations, and take out the first 10 pieces of data according to sorting conditions after merging. For example, if the SQL is:

```
SELECT score FROM t_score ORDER BY score DESC LIMIT 1, 2;
```

The following picture shows the pagination execution results without SQL rewrite.

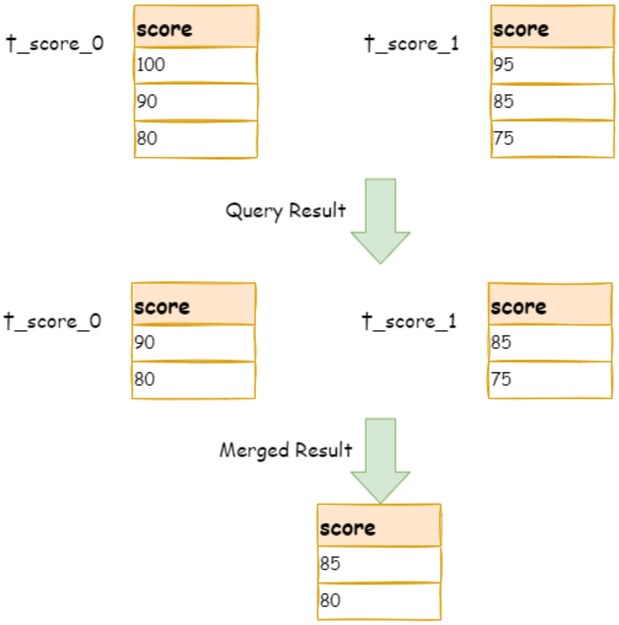


Fig. 4: Pagination without rewrite

As shown in the picture, if you want to acquire the second and the third piece of data ordered by score common in both tables, and they are supposed to be 95 and 90. Since the executed SQL can only acquire the second and the third piece of data from each table, i.e., 90 and 80 from t_score_0, 85 and 75 from t_score_1. When merging results, it can only merge from 90, 80, 85 and 75 already acquired, so the right result cannot be acquired anyway.

The right way is to rewrite pagination conditions as LIMIT 0, 3, take out all the data from the first two pages and combine sorting conditions to calculate the right data. The following picture shows the execution of pagination results after SQL rewrite.

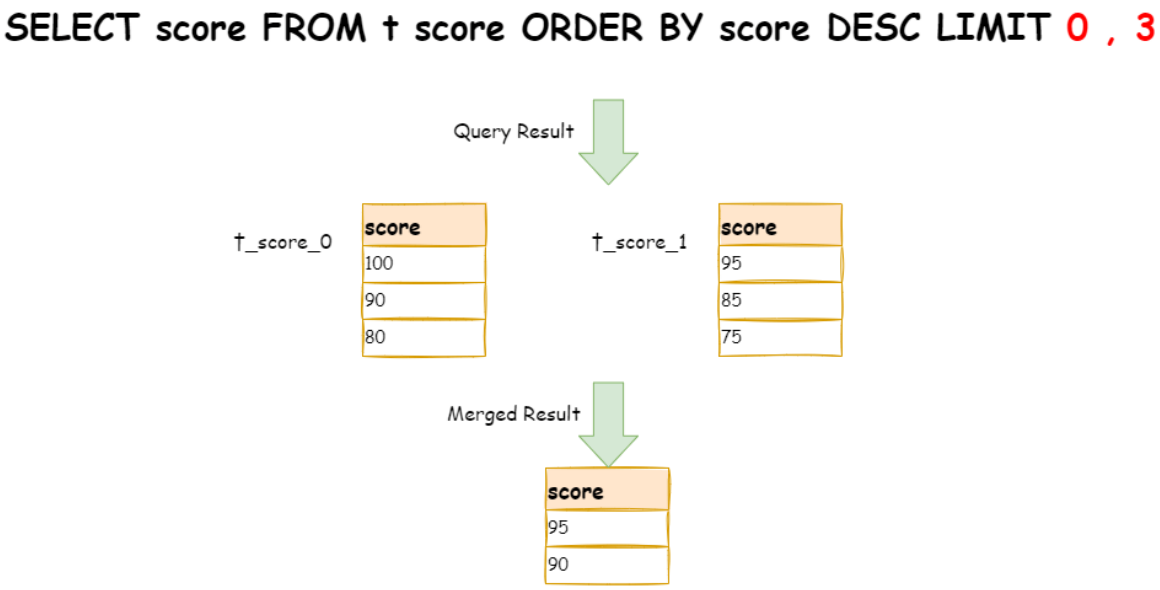


Fig. 5: Pagination with rewrite

The latter the offset position is, the lower the efficiency of using LIMIT pagination will be. There are many ways to avoid using LIMIT as pagination method, such as constructing a secondary index to record line record number and line offset amount, or using the tail ID of last pagination data as the pagination method of conditions of the next query.

When revising pagination information, if the user uses placeholder method to write SQL, he only needs to rewrite parameter list rather than SQL itself.

### Batch Split

When using batch inserted SQL, if the inserted data crosses sharding, the user needs to rewrite SQL to avoid writing excessive data into the database. The differences between insert operation and query operation are: though the query sentence has used sharding keys that do not exist in current sharding, they will not have any influence on data, but insert operation has to delete extra sharding keys. Take the following SQL for example:

```
INSERT INTO t_order (order_id, xxx) VALUES (1, 'xxx'), (2, 'xxx'), (3, 'xxx');
```

If the database is still divided into two parts according to odd and even number of order_id, this SQL will be executed after its table name is revised. Then, both shards will be written with the same record. Though only the data that satisfies sharding conditions can be taken out from query statement, it is not reasonable for the schema to have excessive data. So the SQL should be rewritten as:

```
INSERT INTO t_order_0 (order_id, xxx) VALUES (2, 'xxx');
INSERT INTO t_order_1 (order_id, xxx) VALUES (1, 'xxx'), (3, 'xxx');
```

IN query is similar to batch insertion, but IN operation will not lead to wrong data query result. Through rewriting IN query, the query performance can be further improved. Like the following SQL:

```
SELECT * FROM t_order WHERE order_id IN (1, 2, 3);
```

Is rewritten as:

```
SELECT * FROM t_order_0 WHERE order_id IN (2);
SELECT * FROM t_order_1 WHERE order_id IN (1, 3);
```

The query performance will be further improved. For now, DB Plus Engine has not realized this rewrite strategy, so the current rewrite result is:

```
SELECT * FROM t_order_0 WHERE order_id IN (1, 2, 3);
SELECT * FROM t_order_1 WHERE order_id IN (1, 2, 3);
```

Though the execution result of SQL is right, but it has not achieved the most optimized query efficiency.

### Optimization Rewrite

Its purpose is to effectively improve the performance without influencing the correctness of the query. It can be divided into single node optimization and stream merger optimization.

### Single Node Optimization

It refers to the optimization that stops the SQL rewrite from the route to the single node. After acquiring one route result, if it is routed to a single data node, result merging is unnecessary to be involved, so there is no need for rewrites as derived column, pagination information and others. In particular, there is no need to read from the first piece of information, which reduces the pressure for the database to a large extent and saves meaningless consumption of the network bandwidth.

### Stream Merger Optimization

It only adds sorting items and sorting orders identical with grouping items and ORDER BY to GROUP BY SQL, and they are used to transfer memory merger to stream merger. In the result merger part, stream merger and memory merger will be explained in detail.

The overall structure of rewrite engine is shown in the following picture.

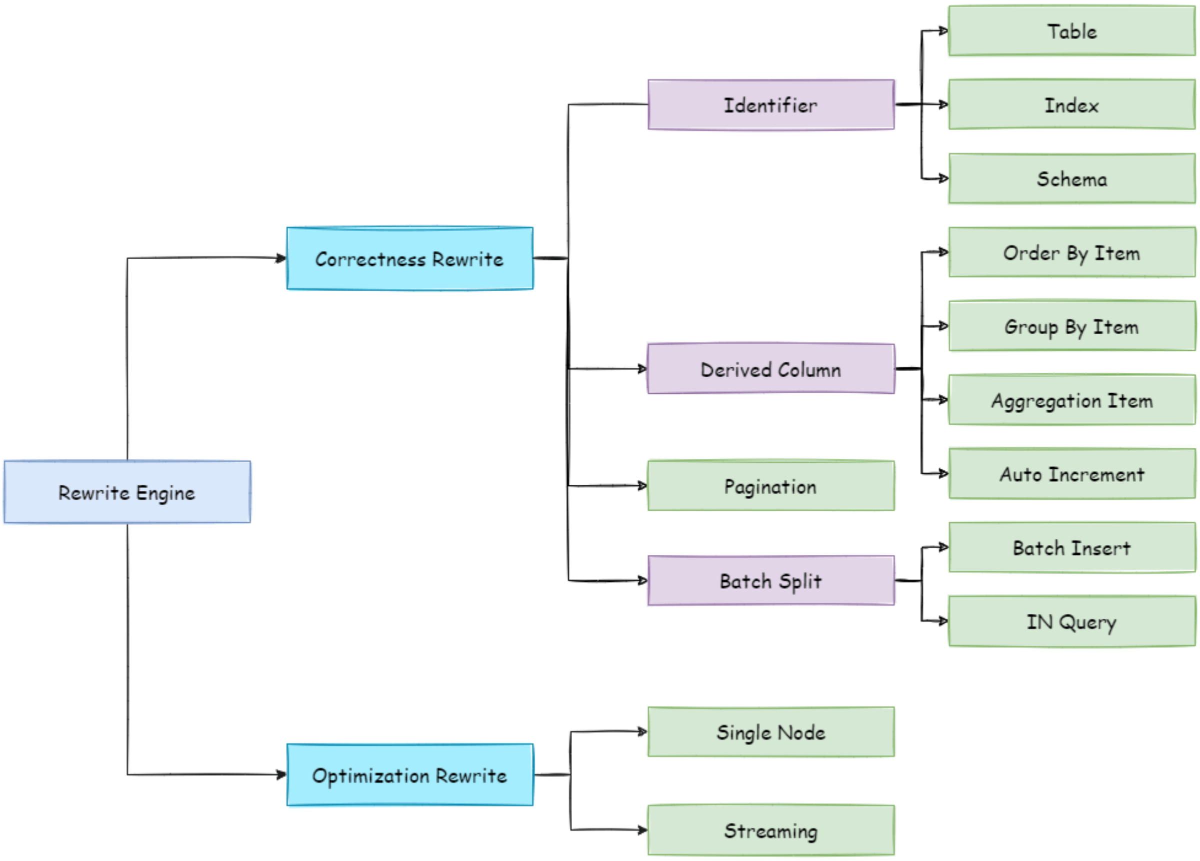


Fig. 6: Rewrite Engine

### 11.2.10 Execute Engine

DB Plus Engine adopts a set of automatic execution engine, responsible for sending the true SQL, which has been routed and rewritten, to execute in the underlying data source safely and effectively. It does not simply send the SQL through JDBC to directly execute in the underlying data source, or put execution requests directly to the thread pool to concurrently execute, but focuses more on the creation of a balanced data source connection, the consumption generated by the memory usage, the maximum utilization of the concurrency and other problems. The objective of the execution engine is to automatically balance between the resource control and the execution efficiency.

#### Connection Mode

From the perspective of resource control, the connection number of the business side' s visit of the database should be limited. It can effectively prevent some certain business from occupying excessive resource, exhausting database connection resources and influencing the normal use of other businesses. Especially when one database contains many tables, a logic SQL that does not contain any sharding key will produce a large amount of physical SQLs that fall into different tables in one database. If each physical SQL takes an independent connection, a query will undoubtedly take up excessive resources.

From the perspective of execution efficiency, holding an independent database connection for each sharding query can make effective use of multi-thread to improve execution efficiency. Opening an independent thread for each database connection can parallelize IO produced consumption. Holding an independent database connection for each sharding query can also avoid loading the query result to the memory too early. It is enough for independent database connections to maintain result set quotation and cursor position, and move the cursor when acquiring corresponding data.

Merging result set by moving down its cursor is called stream merger. It does not require to load all the query results to the memory. Thus, it is able to save memory resource effectively and reduce trash recycle frequency. When it is not able to make sure each sharding query holds an independent database connection, it requires to load all the current query results to the memory before reusing that database connection to acquire the query result from the next sharding table. Therefore, though the stream merger can be used, under this kind of circumstances, it will also degenerate to the memory merger.

The control and protection of database connection resources is one thing, adopting better merging model to save the memory resources of middleware is another thing. How to deal with the relationship between them is a problem that DB Plus Engine execution engine should solve. To be accurate, if a sharding SQL needs to operate 200 tables under some database case, should we choose to create 200 parallel connection executions or a serial connection execution? Or to say, how to choose between efficiency and resource control?

Aiming at the above situation, DB Plus Engine has provided a solution. It has put forward a Connection Mode concept divided into two types, MEMORY_STRICTLY mode and CONNECTION_STRICTLY mode.

### MEMORY_STRICTLY Mode

The prerequisite to use this mode is that DB Plus Engine does not restrict the connection number of one operation. If the actual executed SQL needs to operate 200 tables in some database instance, it will create a new database connection for each table and deal with them concurrently through multi-thread to maximize the execution efficiency. When the SQL is up to standard, it will choose stream merger in priority to avoid memory overflow or frequent garbage recycle.

### CONNECTION_STRICTLY Mode

The prerequisite to use this mode is that DB Plus Engine strictly restricts the connection consumption number of one operation. If the SQL to be executed needs to operate 200 tables in database instance, it will create one database connection and operate them serially. If shards exist in different databases, it will still be multi-thread operations for different databases, but with only one database connection being created for each operation in each database. It can prevent the problem brought by excessive occupation of database connection from one request. The mode chooses memory merger all the time.

The MEMORY_STRICTLY mode is applicable to OLAP operation and can increase the system capacity by removing database connection restrictions. It is also applicable to OLTP operation, which usually has sharding keys and can be routed to a single shard. So it is a wise choice to control database connection strictly to make sure resources of online system databases can be used by more applications.

### Automatic Execution Engine

DB Plus Engine uses which mode at first is up to users’ setting and they can choose to use MEMORY_STRICTLY mode or CONNECTION_STRICTLY mode according to their actual business scenarios.

The solution gives users the right to choose, requiring them to know the advantages and disadvantages of both modes and make decision according to the actual business situations. No doubt, it is not the best solution due to increasing users’ study cost and use cost.

This kind of dichotomy solution lacks flexible coping ability to switch between two modes with static initialization. In practical situations, route results of each time may differ with different SQL and placeholder indexes. It means some operations may need to use memory merger, while others are better to use stream merger. Connection modes should not be set by users before initializing DB Plus Engine, but should be decided dynamically by the situation of SQL and placeholder indexes.

To reduce users’ use cost and solve the dynamic connection mode problem, DB Plus Engine has extracted the thought of automatic execution engine in order to eliminate the connection mode concept inside. Users do not need to know what are so called MEMORY_STRICTLY mode and CONNECTION_STRICTLY mode, but let the execution engine to choose the best solution according to current situations.

Automatic execution engine has narrowed the selection scale of connection mode to each SQL operation. Aiming at each SQL request, automatic execution engine will do real-time calculations and evaluations according to its route result and execute the appropriate connection mode automatically to strike the most optimized balance between resource control and efficiency. For automatic execution engine, users only need to configure maxConnectionSizePerQuery, which represents the maximum connection number allowed by each database for one query.

The execution engine can be divided into two phases: preparation and execution.

### Preparation Phrase

As indicated by its name, this phrase is used to prepare the data to be executed. It can be divided into two steps: result set grouping and unit creation.

Result set grouping is the key to realize the internal connection model concept. According to the configuration option of maxConnectionSizePerQuery, execution engine will choose an appropriate connection mode combined with current route result.

Detailed steps are as follow:

1. Group SQL route results according to data source names.
2. Through the equation in the following picture, users can acquire the SQL route result group to be executed by each database case within the maxConnectionSizePerQuery permission range and calculate the most optimized connection mode of this request.

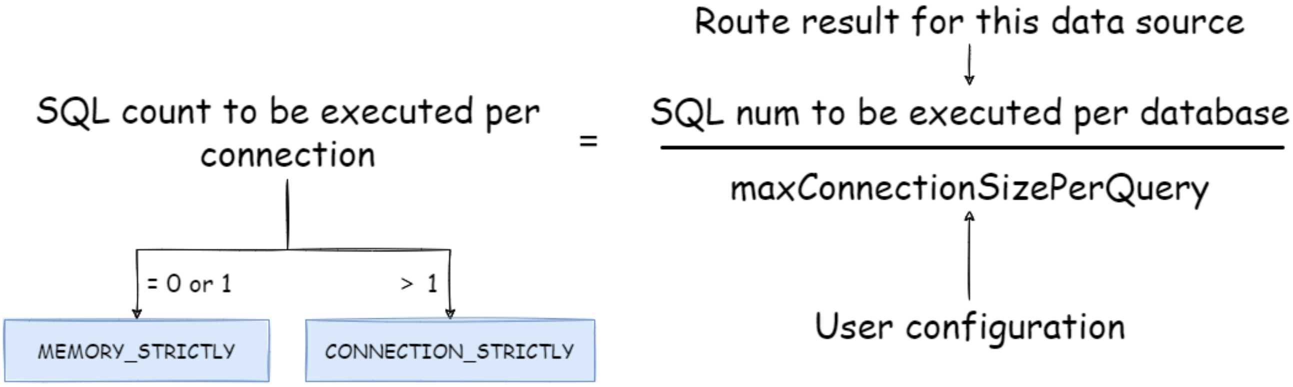

SQL count to be executed per connection
=
Route result for this data source
SQL num to be executed per database
maxConnectionSizePerQuery
User configuration
= 0 or 1
> 1
MEMORY_STRICTLY
CONNECTION_STRICTLY

Fig. 7: Connection mode calculate formula

Within the range that maxConnectionSizePerQuery permits, when the request number that one connection needs to execute is more than 1, meaning current database connection cannot hold the corresponding data result set, it must uses memory merger. On the contrary, when it equals to 1, meaning current database connection can hold the according data result set, it can use stream merger.

Each choice of connection mode aims at each physical database; that is to say, if it is routed to more than one databases, the connection mode of each database may mix with each other and not be the same in one query.

Users can use the route group result acquired from the last step to create the execution unit. When the data source uses technologies, such as database connection pool, to control database connection number, there is some chance for deadlock, if it has not dealt with concurrency properly. As multiple requests waiting for each other to release database connection resources, it will generate hunger wait and cause the crossing deadlock problem.

For example, suppose one query needs to acquire two database connections from a data source and apply them in two table sharding queries routed to one database. It is possible that Query A has already acquired a database connection from that data source and waits to acquire another connection; but in the same time, Query B has also finished it and waits. If the maximum connection number that the connection pool permits is 2, those two query requests will wait forever. The following picture has illustrated the deadlock situation:

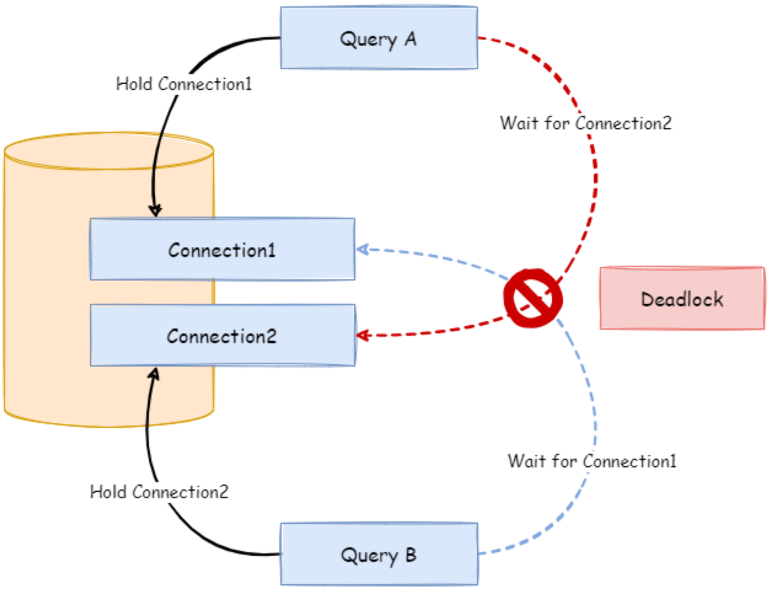


Fig. 8: Dead lock

To avoid the deadlock, DB Plus Engine will go through synchronous processing when acquiring database connection. When creating execution units, it acquires all the database connections that this SQL requires for once with atomic method and reduces the possibility of acquiring only part of the resources. Due to the high operation frequency, locking the connection each time when acquiring it can decrease DB Plus Engine' s concurrency. Therefore, it has improved two aspects here:

1. Avoid the setting that locking only takes one database connection each time. Because under this kind of circumstance, two requests waiting for each other will not happen, so there is no need for locking. Most OLTP operations use sharding keys to route to the only data node, which will make the system in a totally unlocked state, thereby improve the concurrency efficiency further. In addition to routing to a single shard, readwrite-splitting also belongs to this category.
2. Only aim at MEMORY_STRICTLY mode to lock resources. When using CONNECTION_STRICTLY mode, all the query result sets will release database connection resources after loading them to the memory, so deadlock wait will not appear.

### Execution Phrase

Applied in actually SQL execution, this phrase can be divided into two steps: group execution and merger result generation.

Group execution can distribute execution unit groups generated in preparation phrase to the underlying concurrency engine and send events according to each key steps during the execution process, such as starting, successful and failed execution events. Execution engine only focuses on message sending rather than subscribers of the event. Other DB Plus Engine modules, such as distributed transactions, invoked chain tracing and so on, will subscribe focusing events and do corresponding operations. Through the connection mode acquired in preparation phrase, DB Plus Engine will generate memory merger result set or stream merger result set, and transfer it to the result merger engine for the next step.

The overall structure of execution engine is shown as the following picture:

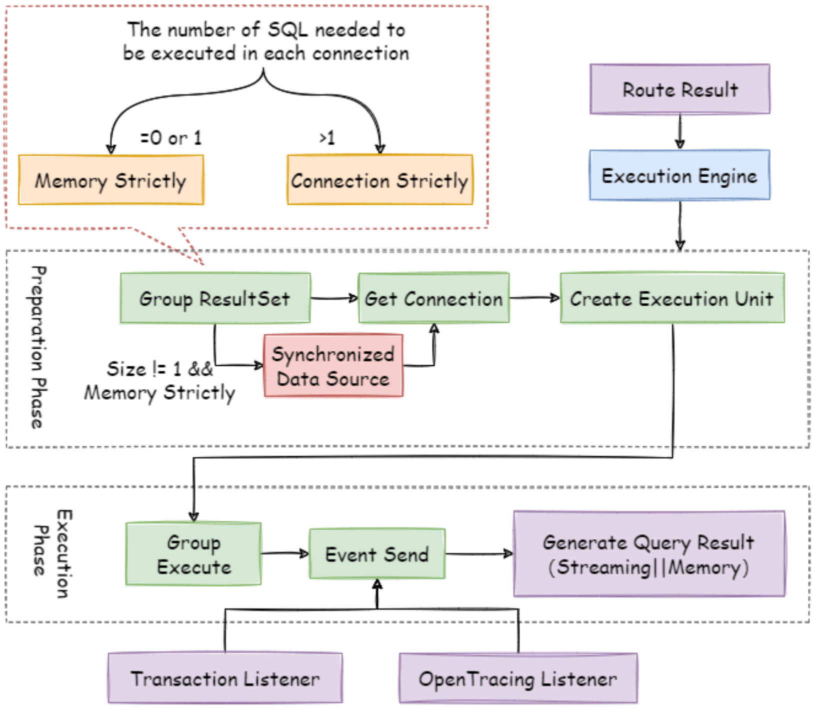


Fig. 9: Execute engine architecture

### 11.2.11 Merger Engine

Result merger refers to merging multi-data result set acquired from all the data nodes as one result set and returning it to the request end rightly.

In function, the result merger supported by DB Plus Engine can be divided into five kinds, iteration, order-by, group-by, pagination and aggregation, which are in composition relation rather than clash relation. In structure, it can be divided into stream merger, memory merger and decorator merger, among which, stream merger and memory merger clash with each other; decorator merger can be further processed based on stream merger and memory merger.

Since the result set is returned from database line by line instead of being loaded to the memory all at once, the most prior choice of merger method is to follow the database returned result set, for it is able to reduce the memory consumption to a large extend.

Stream merger means, each time, the data acquired from the result set is able to return the single piece of right data line by line.

It is the most suitable one for the method that the database returns original result set. Iteration, order-by, and stream group-by belong to stream merger.

Memory merger needs to iterate all the data in the result set and store it in the memory first. after unified grouping, ordering, aggregation and other computations, it will pack it into data result set, which is visited line by line, and return that result set.

Decorator merger merges and reinforces all the result sets function uniformly. Currently, decorator merger has pagination merger and aggregation merger these two kinds.

#### Iteration Merger

As the simplest merger method, iteration merger only requires the combination of multiple data result sets into a single-direction chain table. After iterating current data result sets in the chain table, it only needs to move the element of chain table to the next position and iterate the next data result set.

#### Order-by Merger

Because there is ORDER BY statement in SQL, each data result has its own order. So it is enough only to order data value that the result set cursor currently points to, which is equal to sequencing multiple already ordered arrays, and therefore, order-by merger is the most suitable ordering algorithm in this situation.

When merging order inquiries, DB Plus Engine will compare current data values in each result set (which is realized by Java Comparable interface) and put them into the priority queue. Each time when acquiring the next piece of data, it only needs to move down the result set in the top end of the line, renter the priority order according to the new cursor and relocate its own position.

Here is an instance to explain DB Plus Engine' s order-by merger. The following picture is an illustration of ordering by the score. Data result sets returned by 3 tables are shown in the example and each one of them has already been ordered according to the score, but there is no order between 3 data result sets. Order the data value that the result set cursor currently points to in these 3 result sets. Then put them into the priority queue. the data value of t_score_0 is the biggest, followed by that of t_score_2 and t_score_1 in sequence. Thus, the priority queue is ordered by the sequence of t_score_0, t_score_2 and t_score_1.

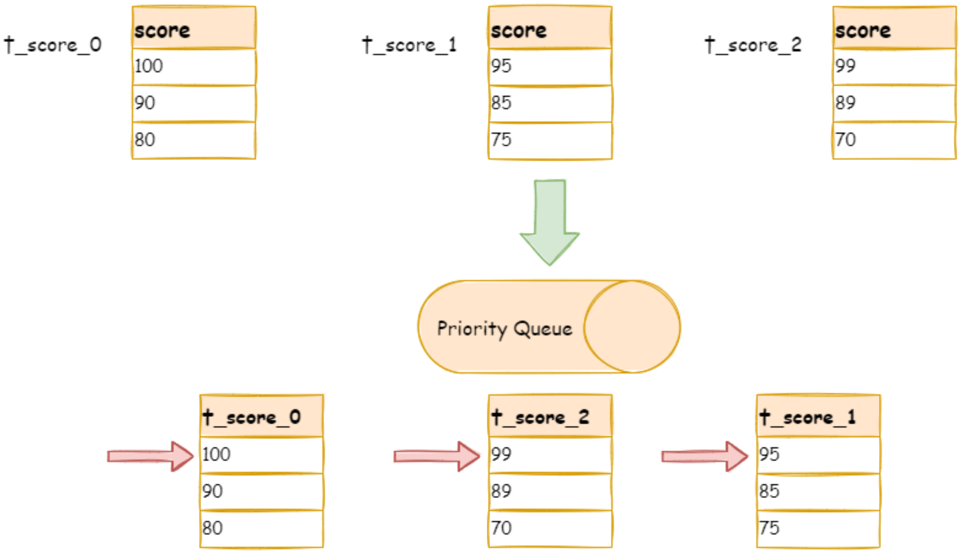


Fig. 10: Order by merger example 1

This diagram illustrates how the order-by merger works when using next invocation. We can see from the diagram that when using next invocation, t_score_0 at the first of the queue will be popped out. After returning the data value currently pointed by the cursor (i.e., 100) to the client end, the cursor will be moved down and t_score_0 will be put back to the queue.

While the priority queue will also be ordered according to the t_score_0 data value (90 here) pointed by the cursor of current data result set. According to the current value, t_score_0 is at the last of the queue, and in the second place of the queue formerly, the data result set of t_score_2, automatically moves to the first place of the queue.

In the second next operation, t_score_2 in the first position is popped out of the queue. Its value pointed by the cursor of the data result set is returned to the client end, with its cursor moved down to rejoin the queue, and the following will be in the same way. If there is no data in the result set, it will not rejoin the queue.

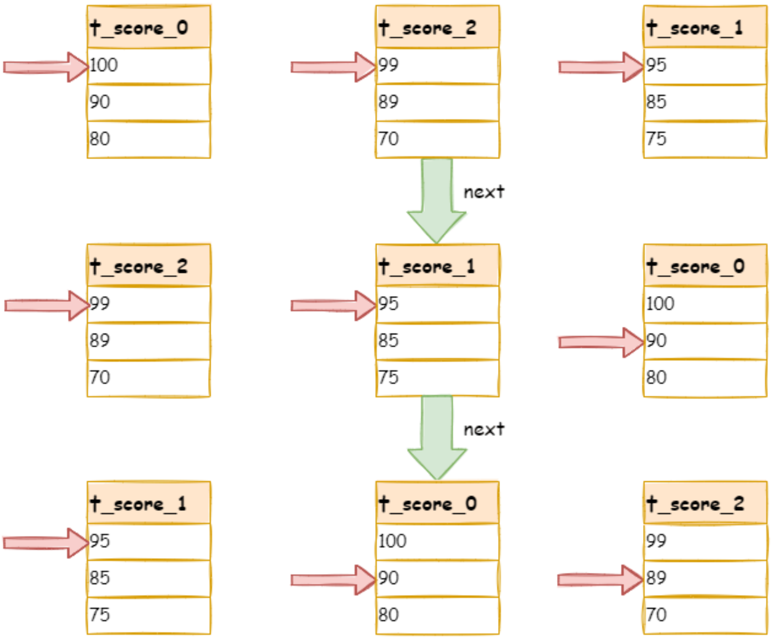


Fig. 11: Order by merger example 2

It can be seen that, under the circumstance that data in each result set is ordered while result sets are disordered, DB Plus Engine does not need to upload all the data to the memory to order. In the order-by merger method, each next operation only acquires the right piece of data each time, which saves the memory consumption to a large extent.

On the other hand, the order-by merger has maintained the orderliness on horizontal axis and vertical axis of the data result set. Naturally ordered, vertical axis refers to each data result set itself, which is acquired by SQL with ORDER BY. Horizontal axis refers to the current value pointed by each data result set, and its order needs to be maintained by the priority queue.Each time when the current cursor moves down, it requires to put the result set in the priority order again, which means only the cursor of the first data result set can be moved down.

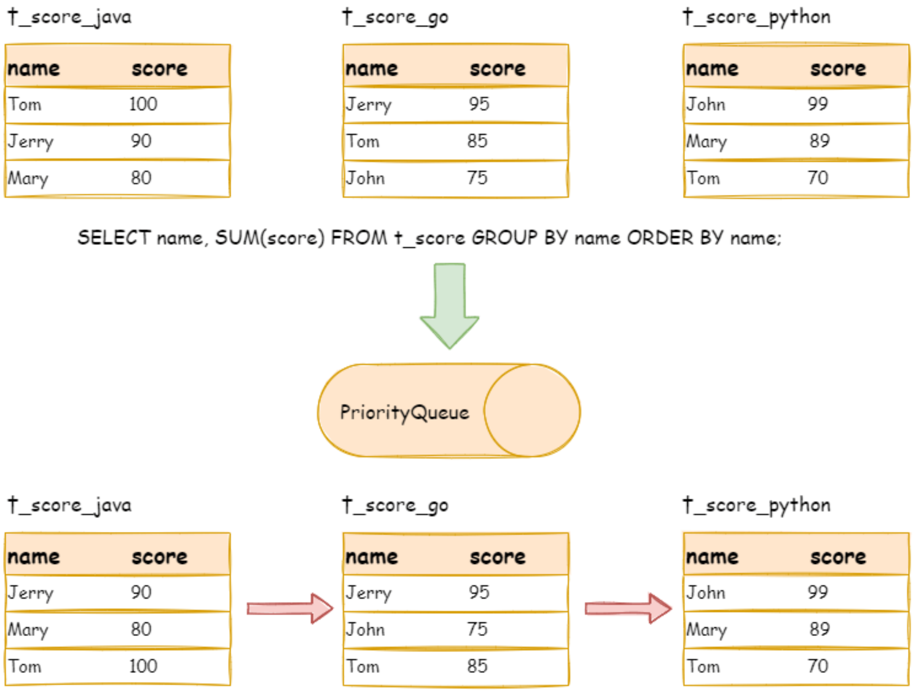
### Group-by Merger

With the most complicated situation, group-by merger can be divided into stream group-by merger and memory group-by merger. Stream group-by merger requires SQL field and order item type (ASC or DESC) to be the same with group-by item. Otherwise, its data accuracy can only be maintained by memory merger.

For instance, if it is sharded by subject, table structure contains examinees' name (to simplify, name repetition is not taken into consideration) and score. The SQL used to acquire each examinee's total score is as follow:

```
SELECT name, SUM(score) FROM t_score GROUP BY name ORDER BY name;
```

When order-by item and group-by item are totally consistent, the data obtained is continuous. The data to group are all stored in the data value that data result set cursor currently points to, stream group-by merger can be used, as illustrated by the diagram:



t_score_java

| name | score |
|---|---|
| Tom | 100 |
| Jerry | 90 |
| Mary | 80 |

t_score_go

| name | score |
|---|---|
| Jerry | 95 |
| Tom | 85 |
| John | 75 |

t_score_python

| name | score |
|---|---|
| John | 99 |
| Mary | 89 |
| Tom | 70 |

SELECT name, SUM(score) FROM t_score GROUP BY name ORDER BY name;

t_score_java

| name | score |
|---|---|
| Jerry | 90 |
| Mary | 80 |
| Tom | 100 |

t_score_go

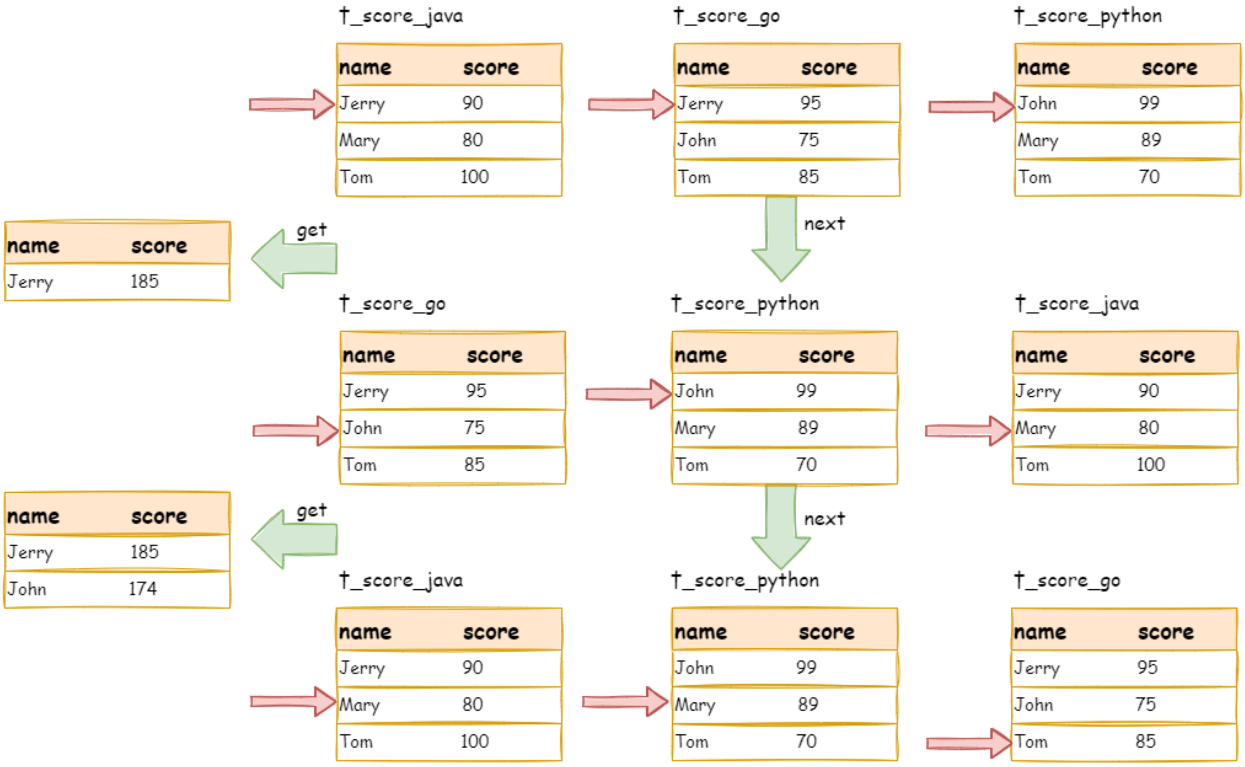
| name | score |
|---|---|
| Jerry | 95 |
| John | 75 |
| Tom | 85 |

t_score_python

| name | score |
|---|---|
| John | 99 |
| Mary | 89 |
| Tom | 70 |

Fig. 12: Group by merger example 1

The merging logic is similar to that of order-by merger. The following picture shows how stream group-by merger works in next invocation.



t_score_java

| name | score |
|---|---|
| Jerry | 90 |
| Mary | 80 |
| Tom | 100 |

t_score_go

| name | score |
|---|---|
| Jerry | 95 |
| John | 75 |
| Tom | 85 |

t_score_python

| name | score |
|---|---|
| John | 99 |
| Mary | 89 |
| Tom | 70 |

| name | score |
|---|---|
| Jerry | 185 |

t_score_go

| name | score |
|---|---|
| Jerry | 95 |
| John | 75 |
| Tom | 85 |

t_score_python

| name | score |
|---|---|
| John | 99 |
| Mary | 89 |
| Tom | 70 |

t_score_java

| name | score |
|---|---|
| Jerry | 90 |
| Mary | 80 |
| Tom | 100 |

| name | score |
|---|---|
| Jerry | 185 |
| John | 174 |

t_score_java

| name | score |
|---|---|
| Jerry | 90 |
| Mary | 80 |
| Tom | 100 |

t_score_python

| name | score |
|---|---|
| John | 99 |
| Mary | 89 |
| Tom | 70 |

t_score_go

| name | score |
|---|---|
| Jerry | 95 |
| John | 75 |
| Tom | 85 |

Fig. 13: Group by merger example 2

We can see from the picture, in the first next invocation, t_score_java in the first position, along with other result set data also having the grouping value of "Jerry" , will be popped out of the queue. After acquiring all the students' scores with the name of "Jerry" , the accumulation operation will be proceeded. Hence, after the first next invocation is finished, the result set acquired is the sum of Jerry' s scores. In the same time, all the cursors in data result sets will be moved down to a different data value next to "Jerry" and rearranged according to current result set value. Thus, the data that contains the second name "John" will be put at the beginning of the queue.

Stream group-by merger is different from order-by merger only in two points:

1. It will take out all the data with the same group item from multiple data result sets for once.
2. It does the aggregation calculation according to aggregation function type.

For the inconsistency between the group item and the order item, it requires to upload all the data to the memory to group and aggregate, since the relevant data value needed to acquire group information is not continuous, and stream merger is not able to use. For example, acquire each examinee' s total score through the following SQL and order them from the highest to the lowest:

```
SELECT name, SUM(score) FROM t_score GROUP BY name ORDER BY score DESC;
```

Then, stream merger is not able to use, for the data taken out from each result set is the same as the original data of the diagram ordered by score in the upper half part structure.

When SQL only contains group-by statement, according to different database implementation, its sequencing order may not be the same as the group order. The lack of ordering statement indicates the order is not important in this SQL. Therefore, through SQL optimization re-write, DB Plus Engine can automatically add the ordering item same as grouping item, converting it from the memory merger that consumes memory to stream merger.

### Aggregation Merger

Whether stream group-by merger or memory group-by merger processes the aggregation function in the same way. Therefore, aggregation merger is an additional merging ability based on what have been introduced above, i.e., the decorator mode. The aggregation function can be categorized into three types, comparison, sum and average.

Comparison aggregation function refers to MAX and MIN. They need to compare all the result set data and return its maximum or minimum value directly.

Sum aggregation function refers to SUM and COUNT. They need to sum up all the result set data.

Average aggregation function refers only to AVG. It must be calculated through SUM and COUNT of SQL re-write, which has been mentioned in SQL re-write, so we will state no more here.

### Pagination Merger

All the merger types above can be paginated. Pagination is the decorator added on other kinds of mergers. DB Plus Engine augments its ability to paginate the data result set through the decorator mode. Pagination merger is responsible for filtering the data unnecessary to acquire.

DB Plus Engine’s pagination function can be misleading to users in that they may think it will take a large amount of memory. In distributed scenarios, it can only guarantee the data accuracy by rewriting LIMIT 10000000, 10 to LIMIT 0, 10000010. Users can easily have the misconception that DB Plus Engine uploads a large amount of meaningless data to the memory and has the risk of memory overflow. Actually, it can be known from the principle of stream merger, only memory group-by merger will upload all the data to the memory. Generally speaking, however, SQL used for OLAP grouping, is applied more frequently to massive calculation or small result generation rather than vast result data generation. Except for memory group-by merger, other cases use stream merger to acquire data result set. So DB Plus Engine would skip unnecessary data through next method in result set, rather than storing them in the memory.

What’s to be noticed, pagination with LIMIT is not the best practice actually, because a large amount of data still needs to be transmitted to DB Plus Engine’s memory space for ordering. LIMIT cannot search for data by index, so paginating with ID is a better solution on the premise that the ID continuity can be guaranteed. For example:

```sql
SELECT * FROM t_order WHERE id > 100000 AND id <= 100010 ORDER BY id;
```

Or search the next page through the ID of the last query result, for example:

```sql
SELECT * FROM t_order WHERE id > 10000000 LIMIT 10;
```

The overall structure of merger engine is shown in the following diagram:

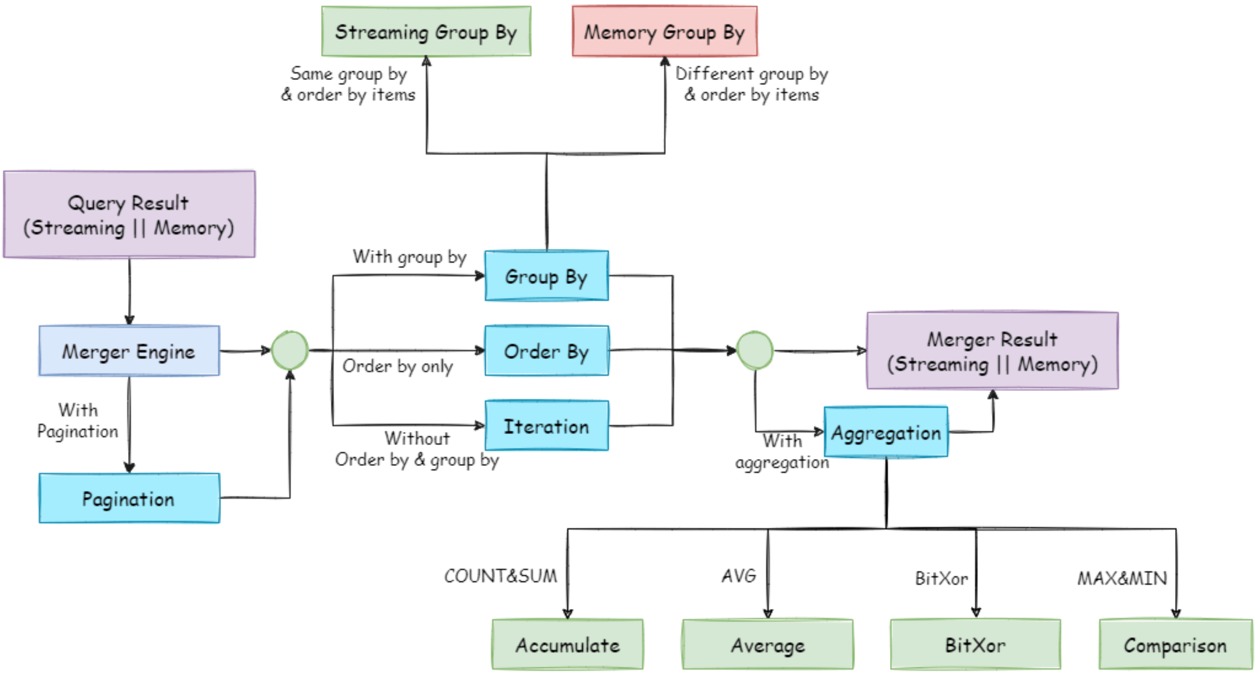


Fig. 14: Merge Architecture

## 11.3 Transaction

### 11.3.1 Navigation

This chapter introduces the principles of the distributed transactions:

- 2PC transaction with XA
- BASE transaction with Seata

### 11.3.2 XA Transaction

XAShardingSphereTransactionManager is XA transaction manager of DBPlusEngine. Its main responsibility is to manage and adapt multiple data sources, and send the corresponding transactions to concrete XA transaction manager.

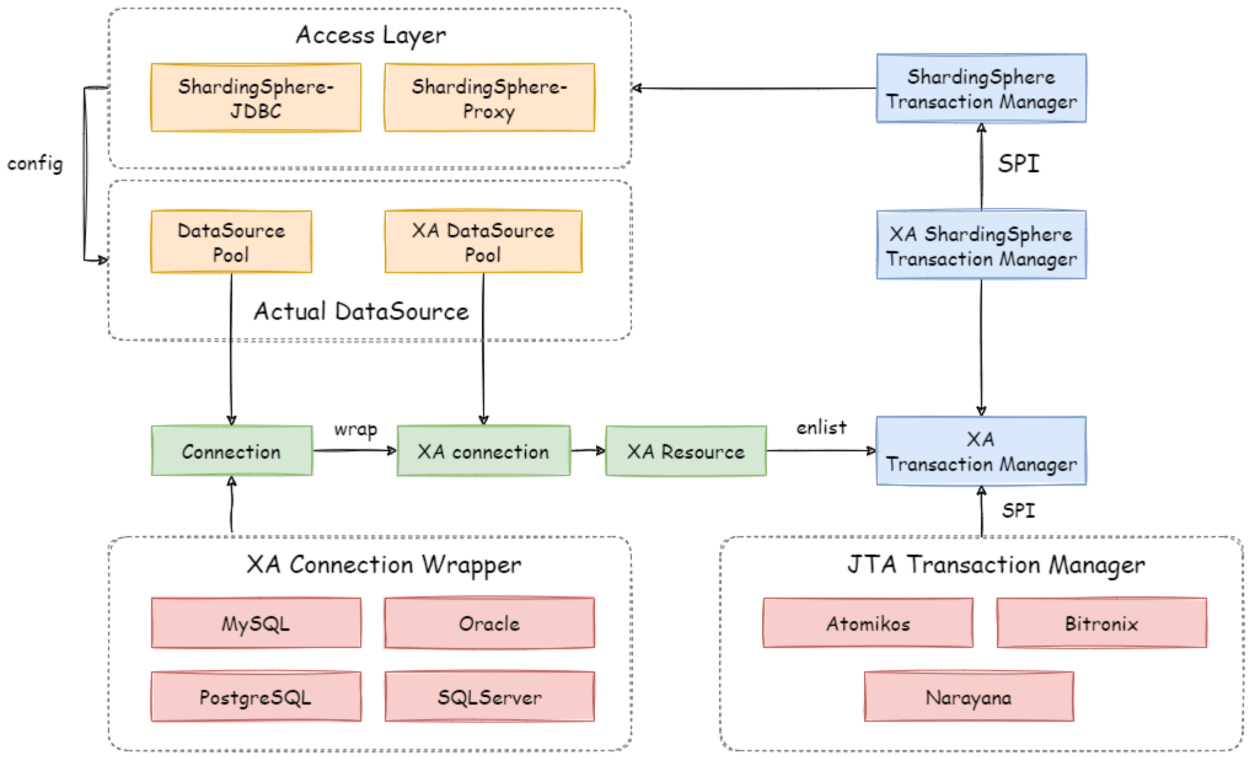


Fig. 15: Principle of ShardingSphere transaction XA

### Transaction Begin

When receiving set autoCommit=0 from client, XAShardingSphereTransactionManager will use XA transaction managers to start overall XA transactions, which is marked by XID.

### Execute actual sharding SQL

After XAShardingSphereTransactionManager register the corresponding XAResource to the current XA transaction, transaction manager will send XAResource.start command to databases. After databases received XAResource.end command, all SQL operator will mark as XA transaction.

For example:

```
XAResource1.start        ## execute in the enlist phase
statement.execute("sql1");
statement.execute("sql2");
XAResource1.end          ## execute in the commit phase
```

sql1 and sql2 in example will be marked as XA transaction.

### Commit or Rollback

After XAShardingSphereTransactionManager receives the commit command in the access, it will delegate it to the actual XA manager. It will collect all the registered XAResource in the thread, before sending XAResource.end to mark the boundary for the XA transaction. Then it will send prepare command one by one to collect votes from XAResource. If all the XAResource feedback is OK, it will send commit command to finally finish it; If there is any No XAResource feedback, it will send rollback command to roll back. After sending the commit command, all XAResource exceptions will be submitted again according to the recovery log to ensure the atomicity and high consistency.

For example:

```
XAResource1.prepare     ## ack: yes
XAResource2.prepare     ## ack: yes
XAResource1.commit
XAResource2.commit

XAResource1.prepare     ## ack: yes
XAResource2.prepare     ## ack: no
XAResource1.rollback
XAResource2.rollback
```

## 11.3.3 Seata BASE transaction

When integrating Seata AT transaction, we need to integrate TM, RM and TC components into DBPlusEngine transaction manager. Seata have proxied DataSource interface in order to RPC with TC. Similarly, DBPlusEngine faced to DataSource interface to aggregate data sources too. After Seata DataSource encapsulation, it is easy to put Seata AT transaction into DBPlusEngine sharding ecosystem.

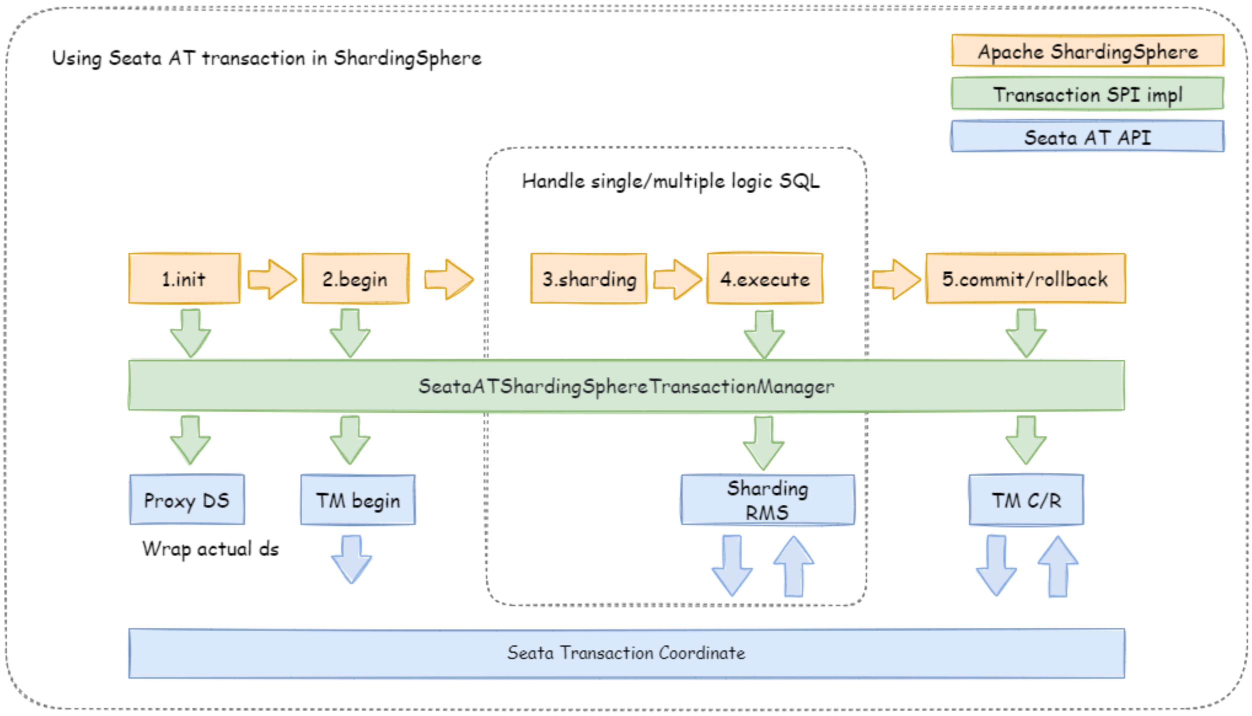


Fig. 16: Seata BASE transaction

### Init Seata Engine

When an application containing ShardingSphereTransactionBaseSeataAT startup, the user-configured DataSource will be wrapped into Seata DataSourceProxy through seata.conf, then registered into RM.

### Transaction Begin

TM controls the boundaries of global transactions. TM obtains the global transaction ID by sending Begin instructions to TC. All branch transactions participate in the global transaction through this global transaction ID. The context of the global transaction ID will be stored in the thread local variable.

### Execute actual sharding SQL

Actual SQL in Seata global transaction will be intercepted to generate undo snapshots by RM and sends participate instructions to TC to join global transaction. Since actual sharding SQLs executed in multi-threads, global transaction context should transfer from main thread to child thread, which is exactly the same as context transfer between services.

### Commit or Rollback

When submitting a Seata transaction, TM sends TC the commit and rollback instructions of the global transaction. TC coordinates all branch transactions for commit and rollback according to the global transaction ID.

# 11.4 Scaling

## 11.4.1 Principle Description

Consider about these challenges of DBPlusEngine-Scaling, the solution is: Use two database clusters temporarily, and switch after the scaling is completed.

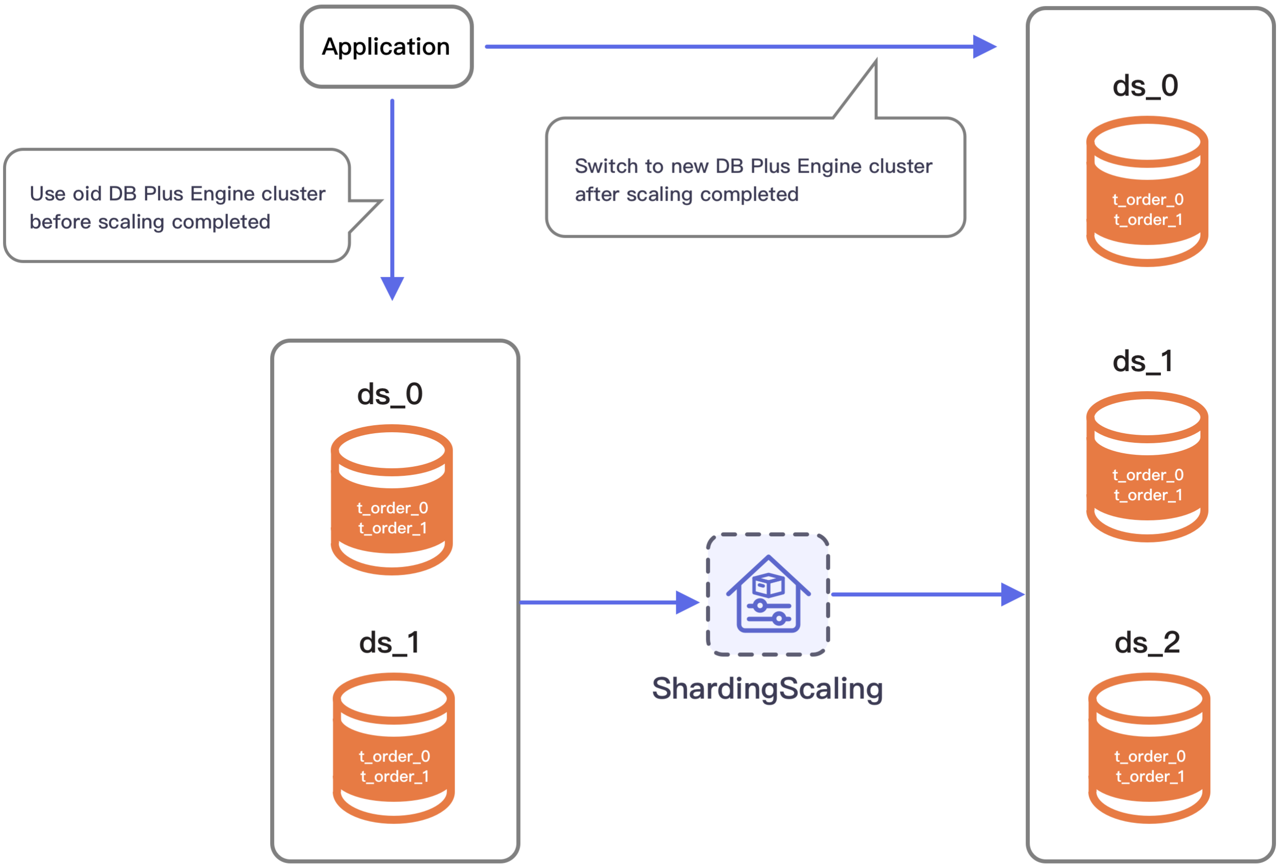


Fig. 17: Scaling Principle Overview

Advantages:

1. No effect for origin data during scaling.
2. No risk for scaling failure.
3. No limited by sharding strategies.

Disadvantages：

1. Redundant servers during scaling.
2. All data needs to be moved.

DBPlusEngine-Scaling will analyze the sharding rules and extract information like datasource and data nodes. According the sharding rules, DBPlusEngine-Scaling create a scaling job with 4 main phases.

1. Preparing Phase.
2. Inventory Phase.
3. Incremental Phase.
4. Switching Phase.

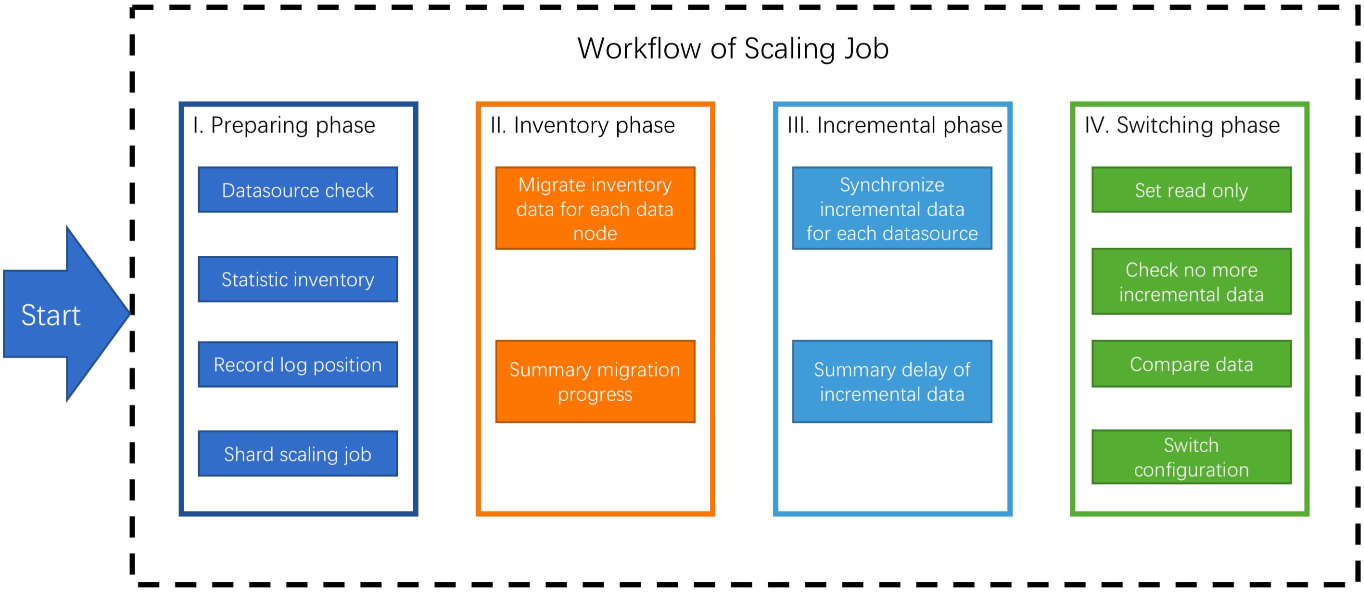


Fig. 18: Workflow

## 11.4.2 Phase Description

### Preparing Phase

DBPlusEngine-Scaling will check the datasource connectivity and permissions, statistic the amount of inventory data, record position of log, shard tasks based on amount of inventory data and the parallelism set by the user.

#### Inventory Phase

Executing the Inventory data migration tasks sharded in preparing phase. DBPlusEngine-Scaling uses JDBC to query inventory data directly from data nodes and write to the new cluster using new rules.

#### Incremental Phase

The data in data nodes is still changing during the inventory phase, so DBPlusEngine-Scaling need to synchronize these incremental data to new data nodes. Different databases have different implementations, but generally implemented by change data capture function based on replication protocols or WAL logs.

- MySQL： subscribe and parse binlog.
- PostgreSQL： official logic replication test_decoding.

These captured incremental data, DBPlusEngine also write to the new cluster using new rules.

#### Switching Phase

In this phase, there may be a temporary read only time, make the data in old data nodes static so that the incremental phase complete fully. The read only time is range seconds to minutes, it depends on the amount of data and the checking data. After finished, DBPlusEngine can switch the configuration by register-center and config-center, make application use new sharding rule and new data nodes.

## 11.5 Encryption

### 11.5.1 Process Details

DBPlusEngine can encrypt the plaintext by parsing and rewriting SQL according to the encryption rule, and store the plaintext (optional) and ciphertext data to the database at the same time. Queries data only extracts the ciphertext data from database and decrypts it, and finally returns the plaintext to user. DBPlusEngine transparently process of data encryption, so that users do not need to know to the implementation details of it, use encrypted data just like as regular data. In addition, DBPlusEngine can provide a relatively complete set of solutions whether the online business system has been encrypted or the new online business system uses the encryption function.

### Overall Architecture

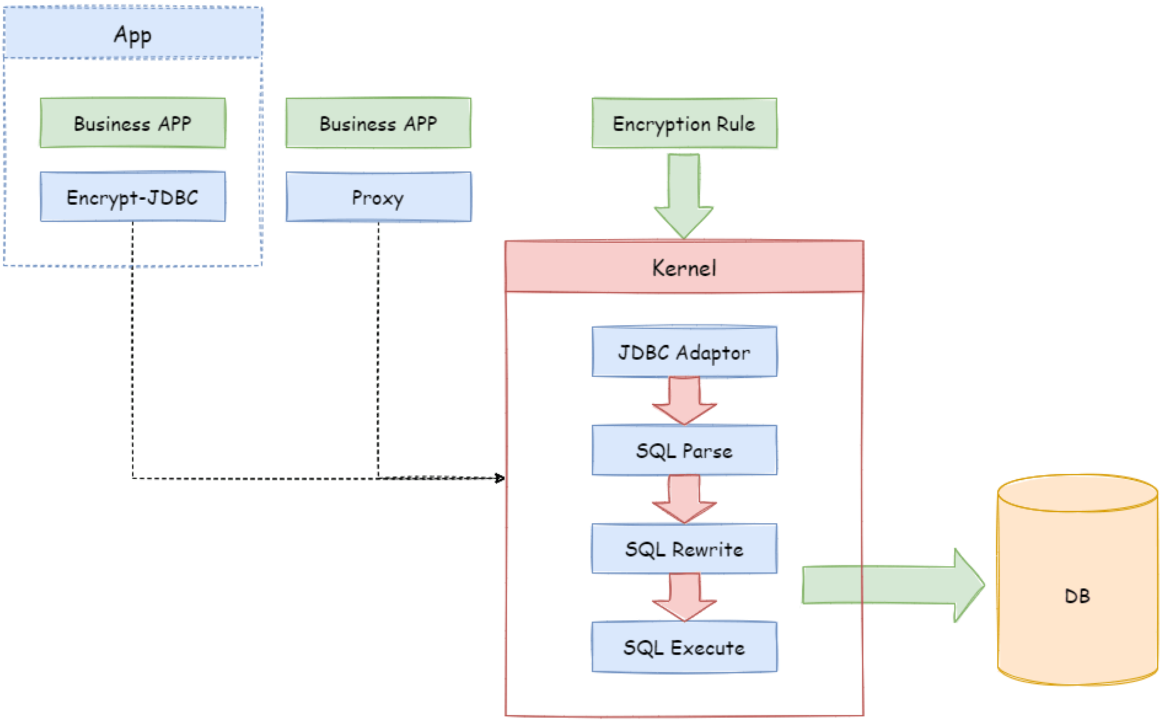


Fig. 19: 1

Encrypt module intercepts SQL initiated by user, analyzes and understands SQL behavior through the SQL syntax parser. According to the encryption rules passed by the user, find out the fields that need to be encrypted/decrypted and the encryptor/decryptor used to encrypt/decrypt the target fields, and then interact with the underlying database. DBPlusEngine will encrypt the plaintext requested by the user and store it in the underlying database; and when the user queries, the ciphertext will be taken out of the database for decryption and returned to the end user. DBPlusEngine shields the encryption of data, so that users do not need to perceive the process of parsing SQL, data encryption, and data decryption, just like using ordinary data.

### Encryption Rule

Before explaining the whole process in detail, we need to understand the encryption rules and configuration, which is the basis of understanding the whole process. The encryption configuration is mainly divided into four parts: data source configuration, encrypt algorithm configuration, encryption table rule configuration, and query attribute configuration. The details are shown in the following figure:

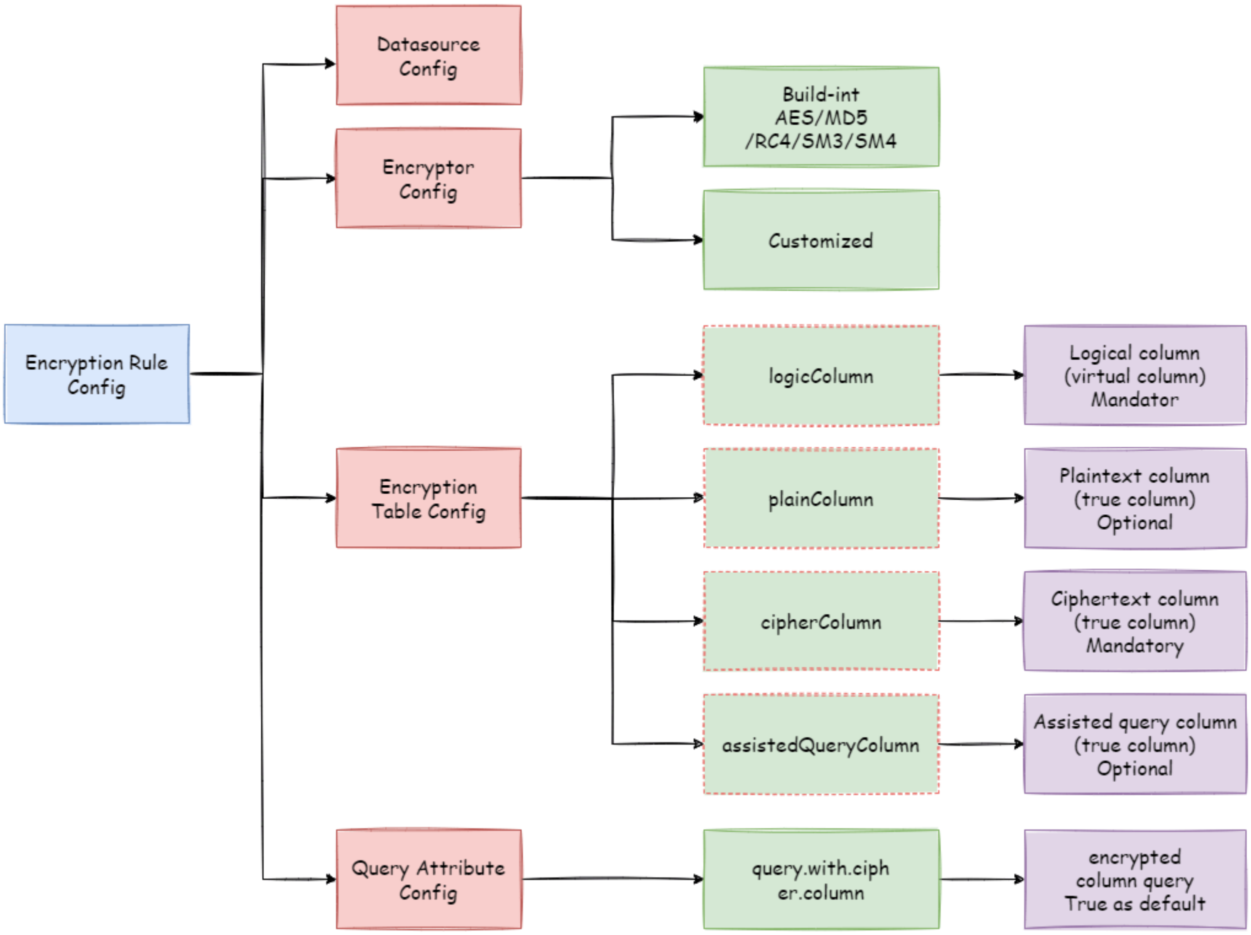

Datasource Config
Encryptor Config
Build-int AES/MD5 /RC4/SM3/SM4
Customized
Encryption Rule Config
logicColumn
Logical column (virtual column) Mandator
Encryption Table Config
plainColumn
Plaintext column (true column) Optional
cipherColumn
Ciphertext column (true column) Mandatory
assistedQueryColumn
Assisted query column (true column) Optional
Query Attribute Config
query.with.cipher.column
encrypted column query True as default

Fig. 20: 2

**Datasource Configuration**: The configuration of DataSource.

**Encrypt Algorithm Configuration**: What kind of encryption strategy to use for encryption and decryption. Currently DBPlusEngine has five built-in encryption/decryption strategies: AES, MD5, RC4, SM3, and SM4. Users can also implement a set of encryption/decryption algorithms by implementing the interface provided by DBPlusEngine.

**Encryption Table Configuration**: Show the DBPlusEngine data table which column is used to store cipher column data (cipherColumn), which column is used to store plain text data (plainColumn), and which column users want to use for SQL writing (logicColumn)

> How to understand Which column do users want to use to write SQL (logicColumn)?
>
> We can understand according to the meaning of DBPlusEngine. The ultimate goal of DBPlusEngine is to shield the encryption of the underlying data, that is, we do not want users to know how the data is encrypted/decrypted, how to store plaintext data in plainColumn, and ciphertext data in cipherColumn. In other words, we do not even want users to know the existence and use of plainColumn and cipherColumn. Therefore, we need to provide users with a column in conceptual. This column can be separated from the real column of the underlying database. It can be a real column in the database table or not, so that the user can freely change the plainColumn and The column name of cipherColumn. Or delete plainColumn and choose to never store plain text and only store cipher text. As long as the user' s SQL is written according to this logical column, and the correct mapping relationship between logicColumn and plainColumn, cipherColumn is given in the encryption rule.
>
> Why do you do this? The answer is at the end of the article, that is, to enable the online services to seamlessly, transparently, and safely carry out data encryption migration.

**Query Attribute configuration**: When the plaintext data and ciphertext data are stored in the underlying database table at the same time, this attribute switch is used to decide whether to directly query the plaintext data in the database table to return, or to query the ciphertext data and decrypt it through DBPlusEngine to return.

### Encryption Process

For example, if there is a table in the database called t_user, there are actually two fields pwd_plain in this table, used to store plain text data, pwd_cipher, used to store cipher text data, and define logicColumn as pwd. Then, when writing SQL, users should write to logicColumn, that is, INSERT INTO t_user SET pwd = '123'. DBPlusEngine receives the SQL, and through the encryption configuration provided by the user, finds that pwd is a logicColumn, so it decrypt the logical column and its corresponding plaintext data. As can be seen that DBPlusEngine has carried out the column-sensitive and data-sensitive mapping conversion of the logical column facing the user and the plaintext and ciphertext columns facing the underlying database. As shown below:

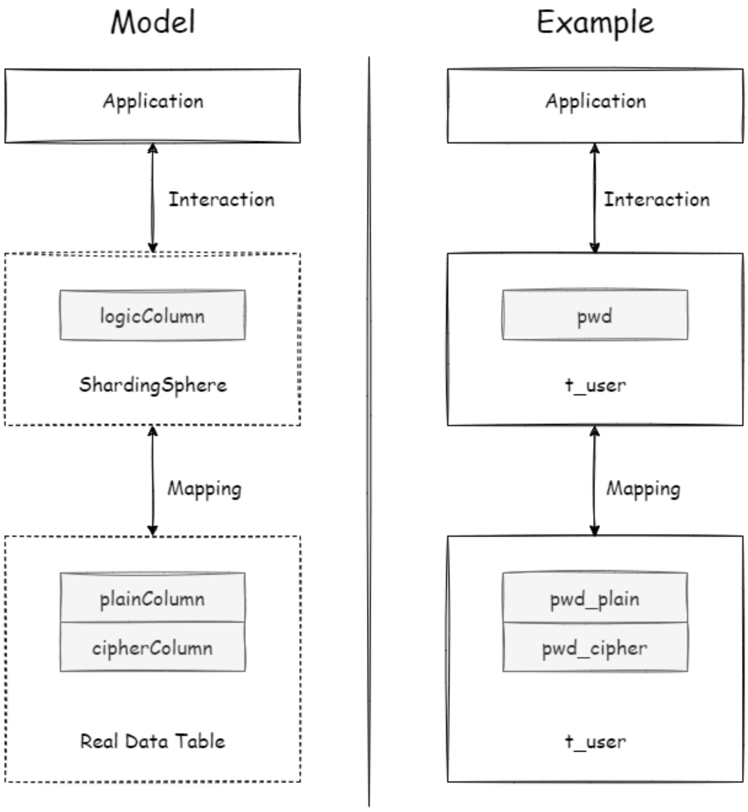


Fig. 21: 3

This is also the core meaning of DBPlusEngine, which is to separate user SQL from the underlying data table structure according to the encryption rules provided by the user, so that the SQL writer by user no longer depends on the actual database table structure. The connection, mapping, and conversion between the user and the underlying database are handled by DBPlusEngine. Why should we do this? It is still the same : in order to enable the online business to seamlessly, transparently and safely perform data encryption migration.

In order to make the reader more clearly understand the core processing flow of DBPlusEngine, the following picture shows the processing flow and conversion logic when using DBPlusEngine to add, delete, modify and check, as shown in the following figure.

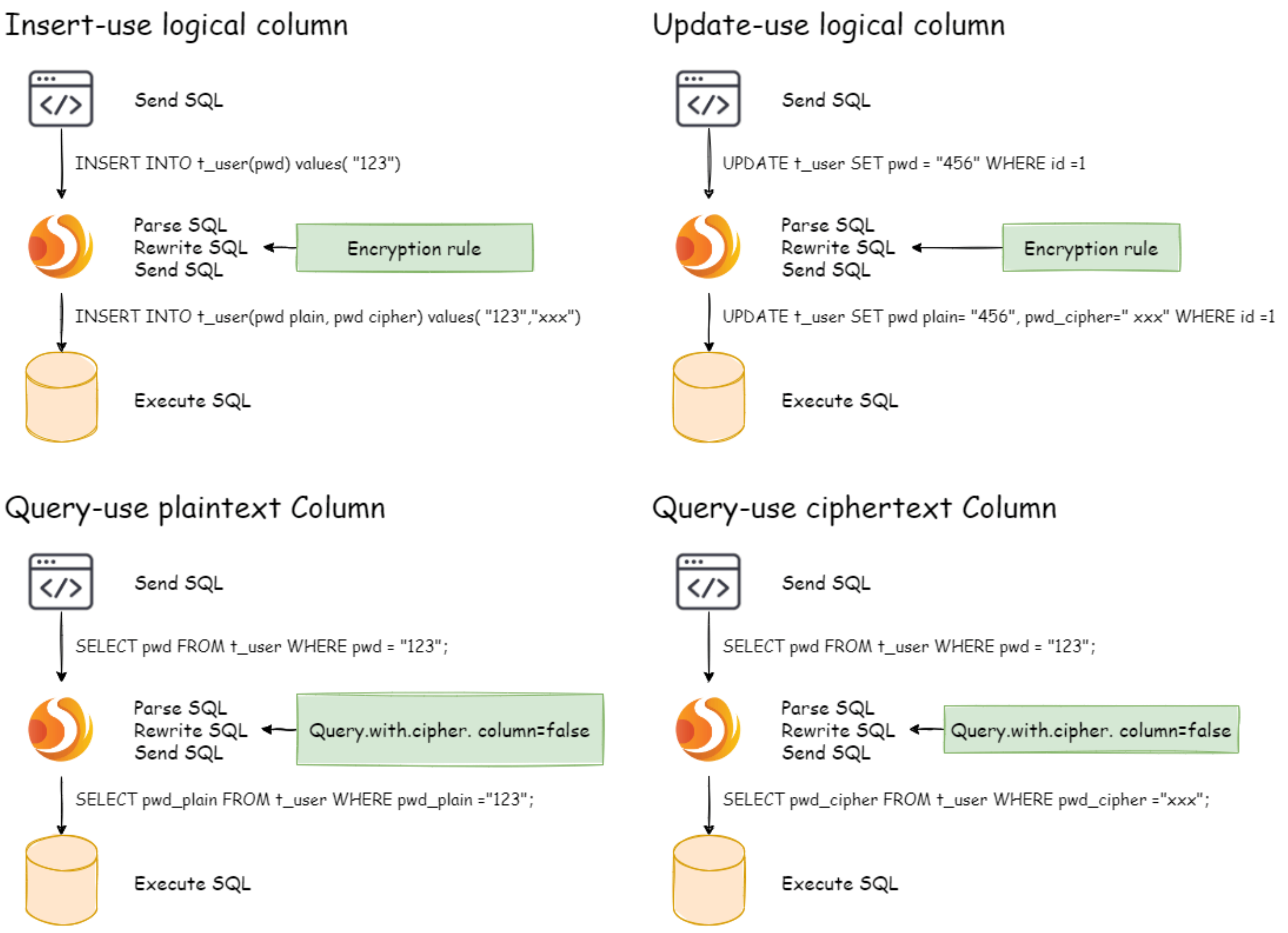

Insert-use logical column
Send SQL
INSERT INTO t_user(pwd) values( "123")
Parse SQL
Rewrite SQL
Send SQL
Encryption rule
INSERT INTO t_user(pwd plain, pwd cipher) values( "123","xxx")
Execute SQL
Update-use logical column
Send SQL
UPDATE t_user SET pwd = "456" WHERE id =1
Parse SQL
Rewrite SQL
Send SQL
Encryption rule
UPDATE t_user SET pwd plain= "456", pwd_cipher=" xxx" WHERE id =1
Execute SQL
Query-use plaintext Column
Send SQL
SELECT pwd FROM t_user WHERE pwd = "123";
Parse SQL
Rewrite SQL
Send SQL
Query.with.cipher. column=false
SELECT pwd_plain FROM t_user WHERE pwd_plain ="123";
Execute SQL
Query-use ciphertext Column
Send SQL
SELECT pwd FROM t_user WHERE pwd = "123";
Parse SQL
Rewrite SQL
Send SQL
Query.with.cipher. column=false
SELECT pwd_cipher FROM t_user WHERE pwd_cipher ="xxx";
Execute SQL

Fig. 22: 4

### 11.5.2 Detailed Solution

After understanding the DBPlusEngine encryption process, you can combine the encryption configuration and encryption process with the actual scenario. All design and development are to solve the problems encountered in business scenarios. So for the business scenario requirements mentioned earlier, how should DBPlusEngine be used to achieve business requirements?

#### New Business

Business scenario analysis: The newly launched business is relatively simple because everything starts from scratch and there is no historical data cleaning problem.

Solution description: After selecting the appropriate encrypt algorithm, such as AES, you only need to configure the logical column (write SQL for users) and the ciphertext column (the data table stores the ciphertext data). It can also be different **. The recommended configuration is as follows (shown in Yaml format):

```
-!ENCRYPT
 encryptors:
  aes_encryptor:
   type: AES
   props:
    aes-key-value: 123456abc
 tables:
```

```
t_user:
  columns:
    pwd:
      cipherColumn: pwd
      encryptorName: aes_encryptor
```

With this configuration, DBPlusEngine only needs to convert logicColumn and cipherColumn. The underlying data table does not store plain text, only cipher text. This is also a requirement of the security audit part. If users want to store plain text and cipher text together in the database, they just need to add plainColumn configuration. The overall processing flow is shown below:

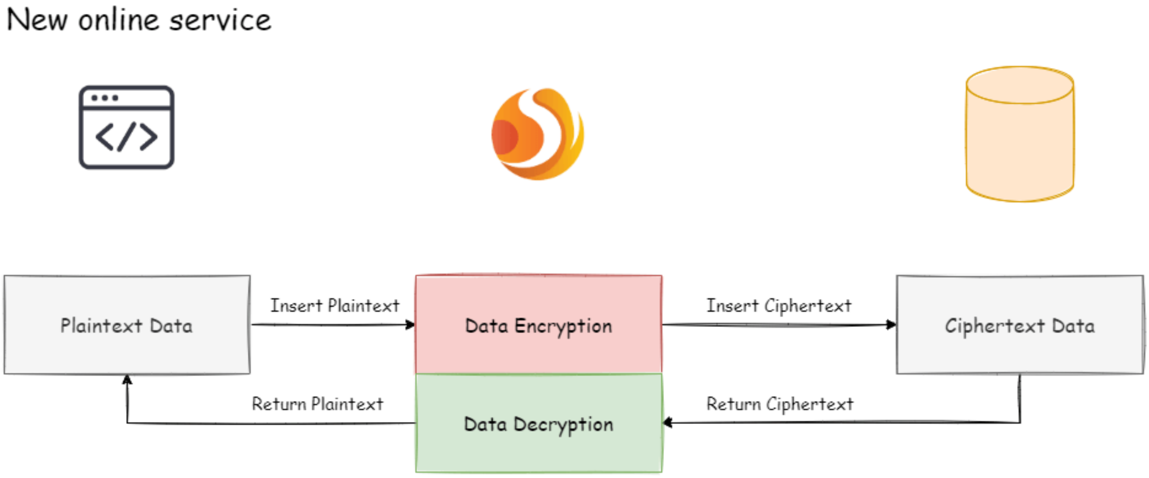


Fig. 23: 5

### Online Business Transformation

Business scenario analysis: As the business is already running online, there must be a large amount of plain text historical data stored in the database. The current challenges are how to enable historical data to be encrypted and cleaned, how to enable incremental data to be encrypted, and how to allow businesses to seamlessly and transparently migrate between the old and new data systems.

Solution description: Before providing a solution, let 's brainstorm: First, if the old business needs to be desensitized, it must have stored very important and sensitive information. This information has a high gold content and the business is relatively important. If it is broken, the whole team KPI is over. Therefore, it is impossible to suspend business immediately, prohibit writing of new data, encrypt and clean all historical data with an encrypt algorithm, and then deploy the previously reconstructed code online, so that it can encrypt and decrypt online and incremental data. Such a simple and rough way, based on historical experience, will definitely not work.

Then another relatively safe approach is to rebuild a pre-release environment exactly like the production environment, and then encrypt the **Inventory plaintext data** of the production environment through the relevant migration and washing tools and store it in the pre-release environment. The **Increment data** is encrypted by tools such as MySQL replica query and the business party ’ s own development, encrypted and stored in the database of the pre-release environment, and then the refactored code can be deployed to the pre-release environment. In this way, the production environment is a set of environment for **modified/queries with plain text as the core**; the pre-release environment is a set of **encrypt/decrypt queries modified with ciphertext as the core**. After comparing for a period of time, the production flow can be cut into the pre-release environment at night. This solution is relatively safe and reliable, but it takes more time, manpower, capital, and costs. It mainly includes: pre-release environment construction, production code rectification, and related auxiliary tool development. Unless there is no way to go, business developers generally go from getting started to giving up.

Business developers must hope: reduce the burden of capital costs, do not modify the business code, and be able to safely and smoothly migrate the system. So, the encryption function module of DBPlusEngine was born. It can be divided into three steps:

1. Before system migration

Assuming that the system needs to encrypt the pwd field of t_user, the business side uses DBPlusEngine to replace the standardized JDBC interface, which basically requires no additional modification (we also provide Spring Boot Starter, Spring Namespace, YAML and other access methods to achieve different services demand). In addition, demonstrate a set of encryption configuration rules, as follows:

```yaml
-!ENCRYPT
  encryptors:
    aes_encryptor:
      type: AES
      props:
        aes-key-value: 123456abc
  tables:
    t_user:
      columns:
        pwd:
          plainColumn: pwd
          cipherColumn: pwd_cipher
          encryptorName: aes_encryptor
  queryWithCipherColumn: false
```

According to the above encryption rules, we need to add a column called pwd_cipher in the t_user table, that is, cipherColumn, which is used to store ciphertext data. At the same time, we set plainColumn to pwd, which is used to store plaintext data, and logicColumn is also set to pwd. Because the previous SQL was written using pwd, that is, the SQL was written for logical columns, so the business code did not need to be changed. Through DBPlusEngine, for the incremental data, the plain text will be written to the pwd column, and the plain text will be encrypted and stored in the pwd_cipher column. At this time, because queryWithCipherColumn is set to false, for business applications, the plain text column of pwd is still used for query storage, but the cipher text data of the new data is additionally stored on the underlying database table pwd_cipher. The processing flow is shown below:

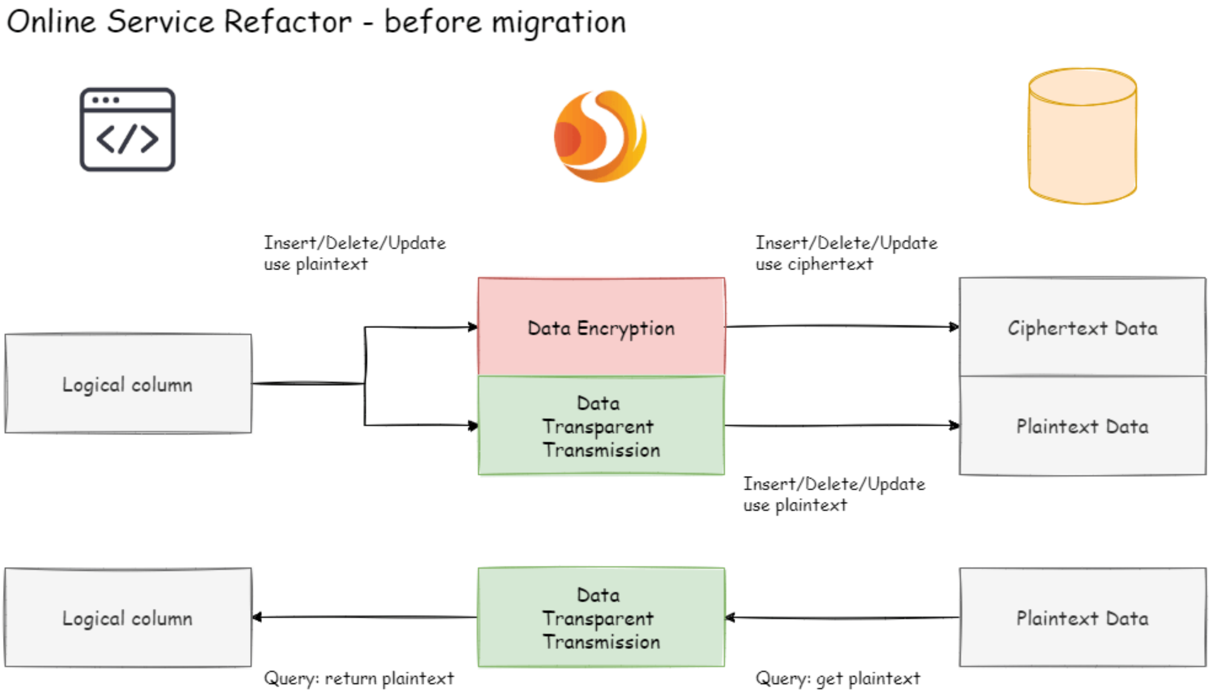


Fig. 24: 6

When the newly added data is inserted, it is encrypted as ciphertext data through DBPlusEngine and stored in the cipherColumn. Now it is necessary to process historical plaintext inventory data. **As DBPlusEngine currently does not provide the corresponding migration and washing tools, the business party needs to encrypt and store the plain text data in pwd to pwd_cipher.**

2. During system migration

The incremental data has been stored by DBPlusEngine in the ciphertext column and the plaintext is stored in the plaintext column; after the historical data is encrypted and cleaned by the business party itself, the ciphertext is also stored in the ciphertext column. That is to say, the plaintext and the ciphertext are stored in the current database. Since the queryWithCipherColumn = false in the configuration item, the ciphertext has never been used. Now we need to set the queryWithCipherColumn in the encryption configuration to true in order for the system to cut the ciphertext data for query. After restarting the system, we found that the system business is normal, but DBPlusEngine has started to extract the ciphertext data from the database, decrypt it and return it to the user; and for the user' s insert, delete and update requirements, the original data will still be stored The plaintext column, the encrypted ciphertext data is stored in the ciphertext column.

Although the business system extracts the data in the ciphertext column and returns it after decryption; however, it will still save a copy of the original data to the plaintext column during storage. Why? The answer is: in order to be able to roll back the system. **Because as long as the ciphertext and plaintext always exist at the same time, we can freely switch the business query to cipherColumn or plainColumn through the configuration of the switch item.** In other words, if the system is switched to the ciphertext column for query, the system reports an error and needs to be rolled back. Then just set queryWithCipherColumn = false, Apache ShardingSphere will restore, that is, start using plainColumn to query again. The processing flow is shown in the following figure:

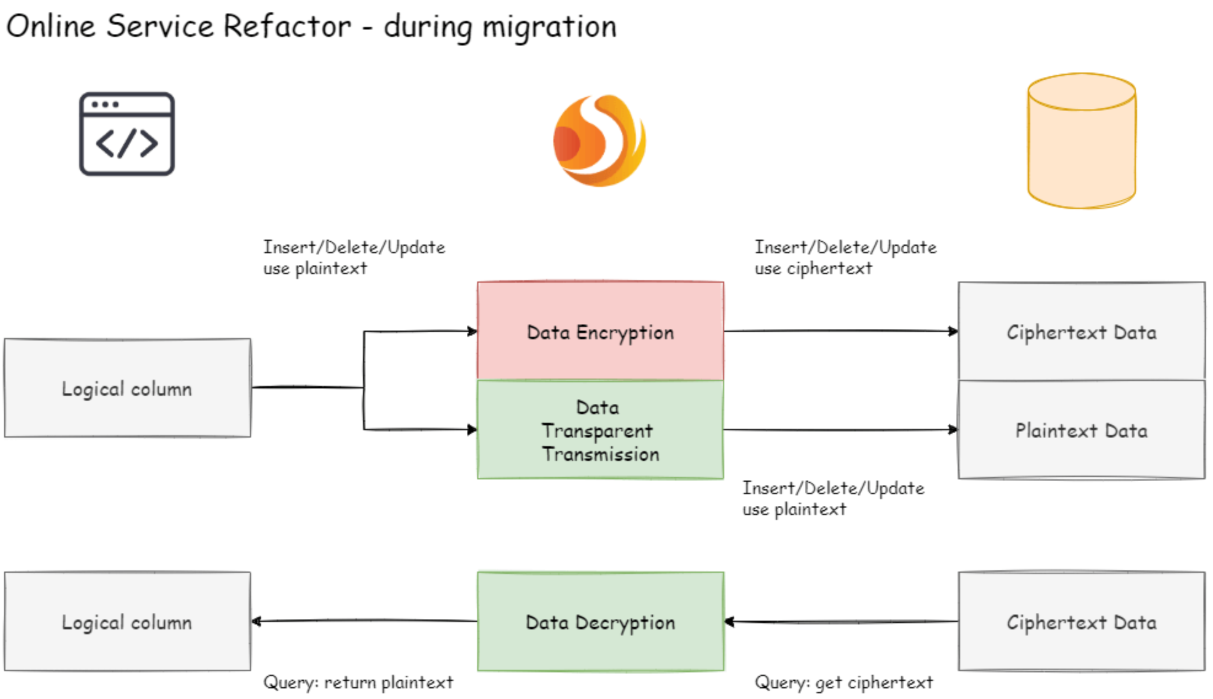


Fig. 25: 7

3. After system migration

Due to the requirements of the security audit department, it is generally impossible for the business system to keep the plaintext and ciphertext columns of the database permanently synchronized. We need to delete the plaintext data after the system is stable. That is, we need to delete plainColumn (ie pwd) after system migration. The problem is that now the business code is written for pwd SQL, delete the pwd in the underlying data table stored in plain text, and use pwd_cipher to decrypt to get the original data, does that mean that the business side needs to rectify all SQL, thus Do not use the pwd column that is about to be deleted? Remember the core meaning of our encrypt module?

> This is also the core meaning of encrypt module. According to the encryption rules provided by the user, the user SQL is separated from the underlying database table structure, so that the user' s SQL writing no longer depends on the actual database table structure. The connection, mapping, and conversion between the user and the underlying database are handled by DBPlusEngine.

Yes, because of the existence of logicColumn, users write SQL for this virtual column. DBPlusEngine can map this logical column and the ciphertext column in the underlying data table. So the encryption configuration after migration is:

```
-!ENCRYPT
 encryptors:
  aes_encryptor:
   type: AES
   props:
    aes-key-value: 123456abc
 tables:
  t_user:
```

```
columns:
 pwd: # pwd and pwd_cipher transformation mapping
  cipherColumn: pwd_cipher
  encryptorName: aes_encryptor
```

The processing flow is as follows:

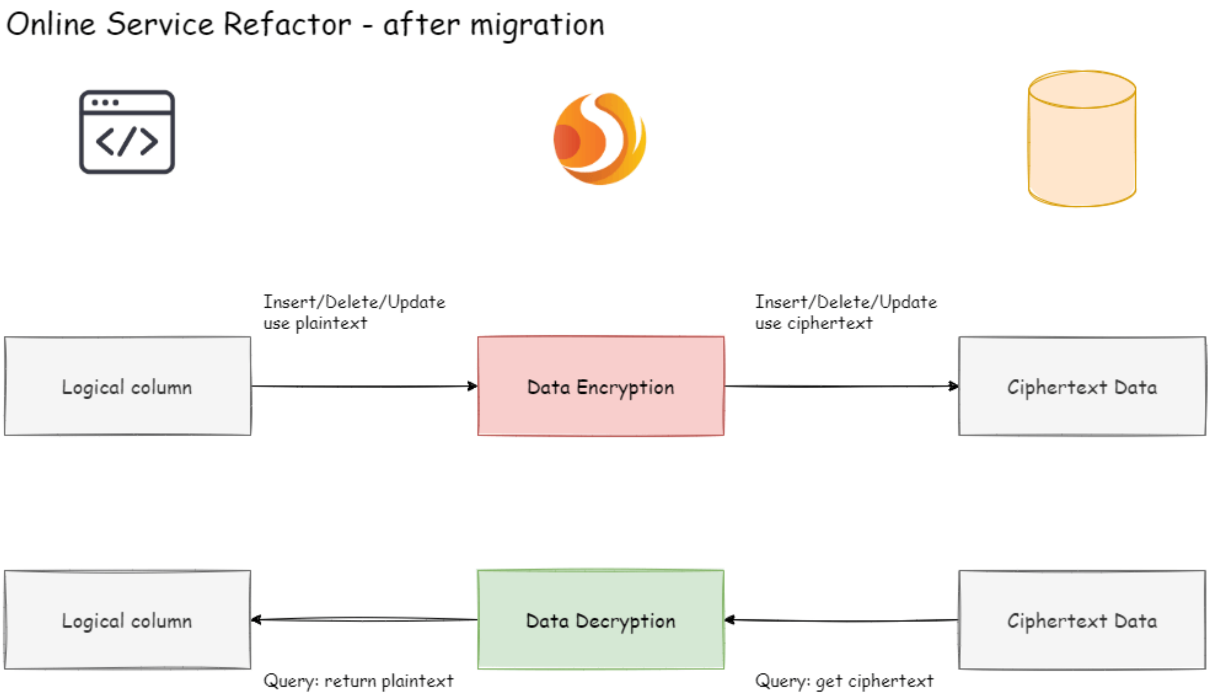


Fig. 26: 8

So far, the online service encryption and rectification solutions have all been demonstrated. We provide Java, YAML, Spring Boot Starter, Spring Namespace multiple ways for users to choose to use, and strive to fulfill business requirements. The solution has been continuously launched on JD Digits, providing internal basic service support.

### 11.5.3 The advantages of Middleware encryption service

1. Transparent data encryption process, users do not need to pay attention to the implementation details of encryption.
2. Provide a variety of built-in, third-party (AKS) encryption strategies, users only need to modify the configuration to use.
3. Provides a encryption strategy API interface, users can implement the interface to use a custom encryption strategy for data encryption.
4. Support switching different encryption strategies.

5. For online services, it is possible to store plaintext data and ciphertext data synchronously, and decide whether to use plaintext or ciphertext columns for query through configuration. Without changing the business query SQL, the on-line system can safely and transparently migrate data before and after encryption.

## 11.5.4 Solution

DBPlusEngine has provided two data encryption solutions, corresponding to two DBPlusEngine encryption and decryption interfaces, i.e., EncryptAlgorithm and QueryAssistedEncryptAlgorithm.

On the one hand, DBPlusEngine has provided internal encryption and decryption implementations for users, which can be used by them only after configuration. On the other hand, to satisfy users' requirements for different scenarios, we have also opened relevant encryption and decryption interfaces, according to which, users can provide specific implementation types. Then, after simple configurations, DBPlusEngine can use encryption and decryption solutions defined by users themselves to desensitize data.

### EncryptAlgorithm

The solution has provided two methods encrypt() and decrypt() to encrypt/decrypt data for encryption.

When users INSERT, DELETE and UPDATE, DBPlusEngine will parse, rewrite and route SQL according to the configuration. It will also use encrypt() to encrypt data and store them in the database. When using SELECT, they will decrypt sensitive data from the database with decrypt() reversely and return them to users at last.

Currently, DBPlusEngine has provided three types of implementations for this kind of encrypt solution, MD5 (irreversible), AES (reversible) and RC4 (reversible), which can be used after configuration.

### QueryAssistedEncryptAlgorithm

Compared with the first encrypt scheme, this one is more secure and complex. Its concept is: even the same data, two same user passwords for example, should not be stored as the same desensitized form in the database. It can help to protect user information and avoid credential stuffing.

This scheme provides three functions to implement, encrypt(), decrypt() and queryAssistedEncrypt(). In encrypt() phase, users can set some variable, timestamp for example, and encrypt a combination of original data + variable. This method can make sure the encrypted data of the same original data are different, due to the existence of variables. In decrypt() phase, users can use variable data to decrypt according to the encryption algorithms set formerly.

Though this method can indeed increase data security, another problem can appear with it: as the same data is stored in the database in different content, users may not be able to find out all the same original data with equivalent query (SELECT FROM table WHERE encryptedColumnn = ?) according to this encryption column. Because of it, we have brought out assistant query column, which is generated by queryAssistedEncrypt(). Different from decrypt(), this method uses another way to encrypt the original data; but for the same original data, it can generate consistent encryption data. Users can store data processed by queryAssistedEncrypt() to assist the query of original data. So there may be one more assistant query column in the table.

queryAssistedEncrypt() and encrypt() can generate and store different encryption data; decrypt() is reversible and queryAssistedEncrypt() is irreversible. So when querying the original data, we will parse, rewrite and route SQL automatically. We will also use assistant query column to do WHERE queries and use decrypt() to decrypt encrypt() data and return them to users. All these can not be felt by users.

For now, DBPlusEngine has abstracted the concept to be an interface for users to develop rather than providing accurate implementation for this kind of encrypt solution. DBPlusEngine will use the accurate implementation of this solution provided by users to desensitize data.

# 11.6 Shadow

## 11.6.1 Overall Architecture

DBPlusEngine makes shadow judgments on incoming SQL by parsing SQL, according to the shadow rules set by the user in the configuration file, route to production DB or shadow DB.

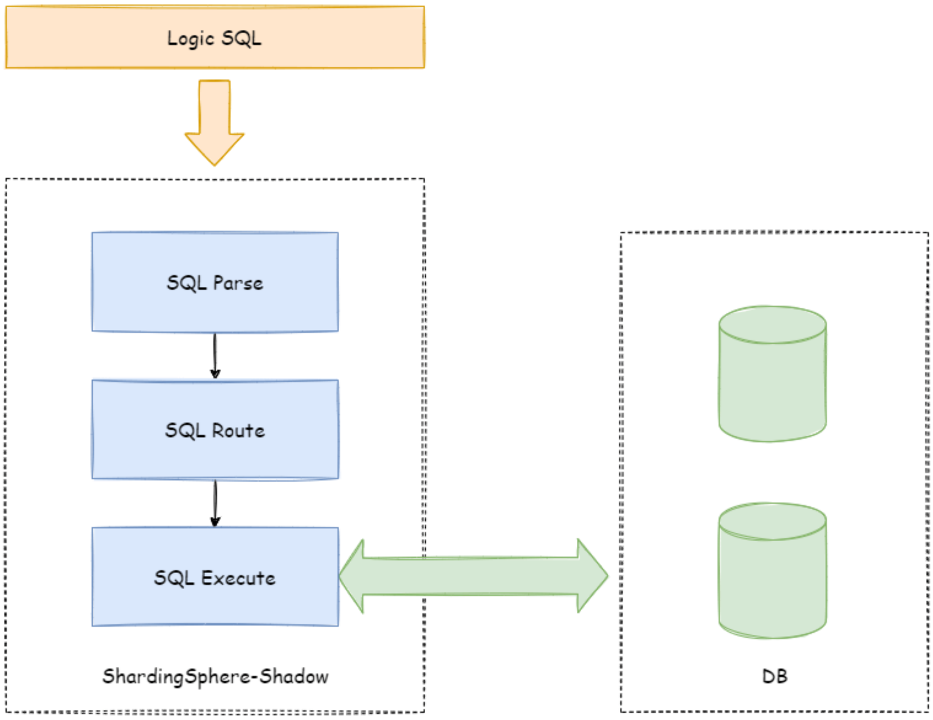


Fig. 27: Execute Process

### 11.6.2 Shadow Rule

Shadow rules include shadow data source mapping, shadow tables, and shadow algorithms.

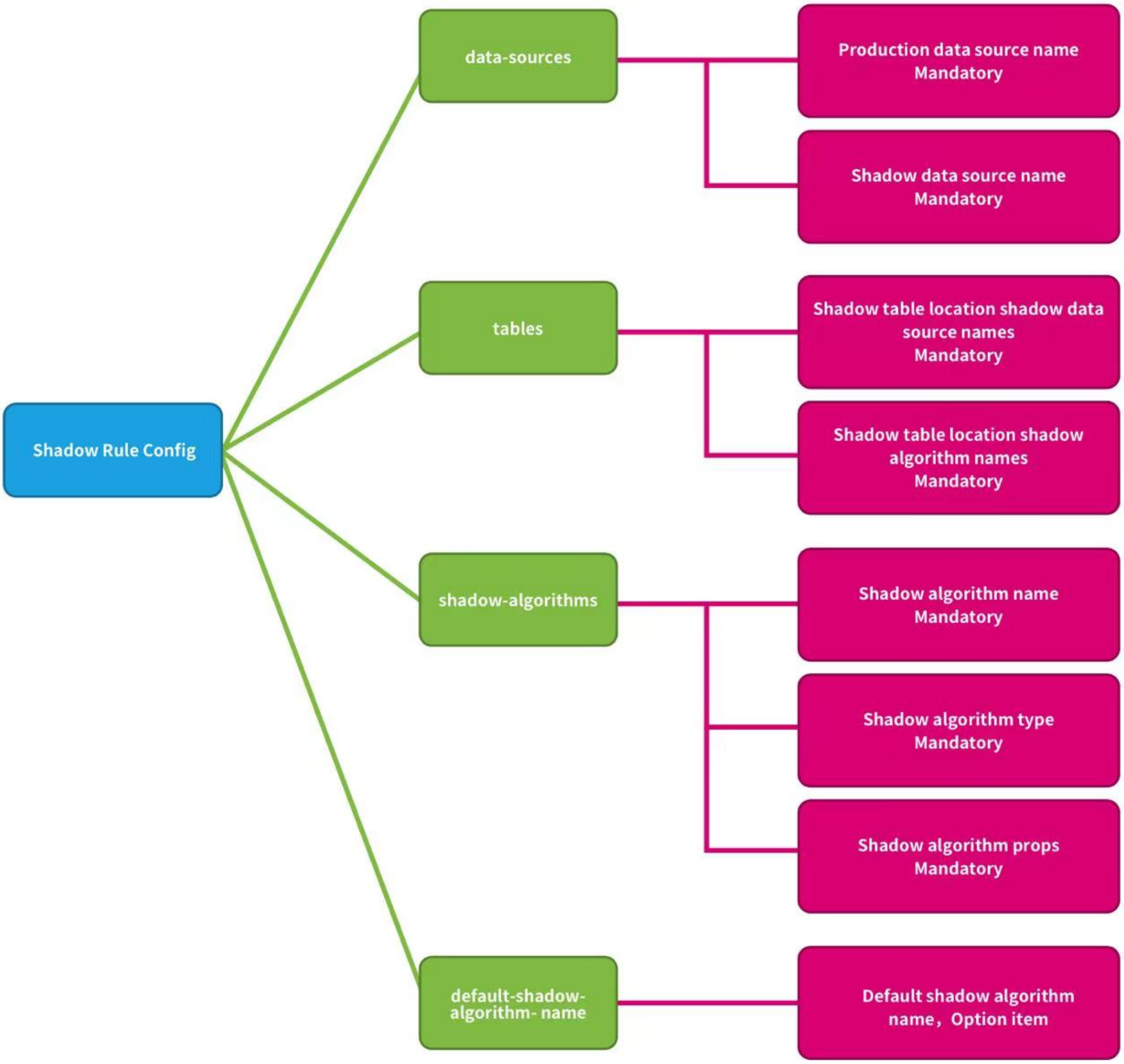

Shadow Rule Config
data-sources
tables
shadow-algorithms
default-shadow-algorithm- name
Production data source name Mandatory
Shadow data source name Mandatory
Shadow table location shadow data source names Mandatory
Shadow table location shadow algorithm names Mandatory
Shadow algorithm name Mandatory
Shadow algorithm type Mandatory
Shadow algorithm props Mandatory
Default shadow algorithm name，Option item

Fig. 28: Shadow Rule

**data-sources**: Production data source name and shadow data source name mappings.

**tables**: Shadow tables related to stress testing. Shadow tables must exist in the specified shadow DB, and the shadow algorithm needs to be specified.

**shadow-algorithms**: SQL routing shadow algorithm.

**default-shadow-algorithm-name**: Default shadow algorithm. Optional item, the default matching algorithm for tables that not configured with the shadow algorithm.

## 11.6.3 Routing Process

Take the INSERT statement as an example. When writing data DBPlusEngine will parse the SQL, and then construct a routing chain according to the rules in the configuration file.

In the current version of the function, the shadow function is the last execution unit in the routing chain, that is, if there are other rules that require routing, such as sharding, DBPlusEngine will first route to a certain database according to the sharding rules, and then perform the shadow routing decision process.

It determined that the execution of SQL satisfies the configuration of the shadow rule, the data routed to the corresponding shadow database, and the production data remains unchanged.

## 11.6.4 Shadow Judgment Process

The Shadow DB performs shadow judgment on the executed SQL statements.

Shadow judgment supports two types of algorithms, users can choose one or combine them according to actual business needs.

### DML Statement

Support two type shadow algorithms.

The shadow judgment first judges whether there is an intersection between SQL related tables and configured shadow tables.

If there is an intersection, determine the shadow algorithm associated with the shadow table of the intersection in turn， and any one of them was successful. SQL statement executed shadow DB.

If shadow tables have no intersection, or shadow algorithms are unsuccessful, SQL statement executed production DB.

### DDL Statement

Only support note shadow algorithm.

In the pressure testing scenarios, DDL statements are not need tested generally. It is mainly used when initializing or modifying the shadow table in the shadow DB.

The shadow judgment first judges whether the executed SQL contains notes.

If contains notes, determine the note shadow algorithms in the shadow rule in turn, and any one of them was successful. SQL statement executed shadow DB.

The executed SQL does not contain notes, or shadow algorithms are unsuccessful, SQL statement executed production DB.

## 11.6.5 Shadow Algorithm

**Shadow algorithm details, please refer to** List

### 11.6.6 Use Example

#### Scenario

Assume that the e-commerce website wants to perform pressure testing on the order business,

the pressure testing related table t_order is a shadow table，the production data executed to the ds production DB, and the pressure testing data executed to the database ds_shadow shadow DB.

#### Shadow DB configuration

The shadow configuration for example(YAML)：

```yaml
data-sources:
  shadow-data-source:
    source-data-source-name: ds
    shadow-data-source-name: ds-shadow
tables:
  t_order:
    data-source-names: shadow-data-source
    shadow-algorithm-names:
      - simple-hint-algorithm
      - user-id-value-match-algorithm
shadow-algorithms:
  simple-hint-algorithm:
    type: SIMPLE_HINT
    props:
      foo: bar
  user-id-value-match-algorithm:
    type: VALUE_MATCH
    props:
      operation: insert
      column: user_id
      value: 0

sql-parser:
  sql-comment-parse-enabled: true
```

**Note**: If you use the Hint shadow algorithm, the parse SQL comment configuration item sql-comment-parse-enabled: true need to be turned on. turned off by default. please refer to SQL-PARSER Configuration

#### Shadow DB environment

- Create the shadow DB ds_shadow.
- Create shadow tables, tables structure must be consistent with the production environment. Assume that the t_order table created in the shadow DB. Create table statement need to add SQL comment /*foo:bar,.. .*/.

```sql
CREATE TABLE t_order (order_id INT(11) primary key, user_id int(11) not null, ...) /*foo:bar,...*/
```

Execute to the shadow DB.

**Note**: If use the MySQL client for testing, the link needs to use the parameter -c, for example:

```
mysql> mysql -u root -h127.0.0.1 -P3306 -proot -c
```

Parameter description: keep the comment, send the comment to the server

Execute SQL containing annotations, for example:

```
SELECT * FROM table_name /*shadow:true,foo:bar*/;
```

Comment statement will be intercepted by the MySQL client if parameter -c not be used, for example:

```
SELECT * FROM table_name;
```

Affect test results.

### Shadow algorithm example

1. Column shadow algorithm example

Assume that the t_order table contains a list of user_id to store the order user ID. The data of the order created by the user whose user ID is 0 executed to shadow DB, other data executed to production DB.

```
INSERT INTO t_order (order_id, user_id, ...) VALUES (xxx..., 0, ...)
```

No need to modify any SQL or code, only need to control the data of the testing to realize the pressure testing.

Column Shadow algorithm configuration (YAML):

```
shadow-algorithms:
  user-id-value-match-algorithm:
    type: VALUE_MATCH
    props:
      operation: insert
      column: user_id
      value: 0
```

**Note**: When the shadow table uses the column shadow algorithm, the same type of shadow operation (INSERT, UPDATE, DELETE, SELECT) currently only supports a single column.

2. Hint shadow algorithm example

Assume that the t_order table does not contain columns that can matching. Executed SQL statement need to add SQL note /*foo:bar,.. .*/

```
SELECT * FROM t_order WHERE order_id = xxx /*foo:bar,...*/
```

SQL executed to shadow DB, other data executed to production DB.

Note Shadow algorithm configuration (YAML):

```
shadow-algorithms:
  simple-hint-algorithm:
    type: SIMPLE_HINT
    props:
      foo: bar
```

3. Hybrid two shadow algorithm example

Assume that the pressure testing of the t_order gauge needs to cover the above two scenarios.

```
INSERT INTO t_order (order_id, user_id, ...) VALUES (xxx..., 0, ...);

SELECT * FROM t_order WHERE order_id = xxx /*foo:bar,...*/;
```

Both will be executed to shadow DB, other data executed to production DB.

2 type of shadow algorithm example (YAML):

```
shadow-algorithms:
  user-id-value-match-algorithm:
    type: VALUE_MATCH
    props:
```

```
   operation: insert
   column: user_id
   value: 0
 simple-hint-algorithm:
  type: SIMPLE_HINT
  props:
   foo: bar
```

4. Default shadow algorithm example

Assume that the column shadow algorithm used for the t_order, all other shadow tables need to use the note shadow algorithm.

```
INSERT INTO t_order (order_id, user_id, ...) VALUES (xxx..., 0, ...);

INSERT INTO t_xxx_1 (order_item_id, order_id, ...) VALUES (xxx..., xxx..., ...) /*foo:bar,...*/;

SELECT * FROM t_xxx_2 WHERE order_id = xxx /*foo:bar,...*/;

SELECT * FROM t_xxx_3 WHERE order_id = xxx /*foo:bar,...*/;
```

Both will be executed to shadow DB, other data executed to production DB.

Default shadow algorithm configuration (YAML):

```
data-sources:
 shadow-data-source:
  source-data-source-name: ds
  shadow-data-source-name: ds-shadow
tables:
 t_order:
  data-source-names: shadow-data-source
  shadow-algorithm-names:
   - simple-hint-algorithm
   - user-id-value-match-algorithm
default-shadow-algorithm-name: simple-note-algorithm
shadow-algorithms:
 simple-hint-algorithm:
  type: SIMPLE_HINT
  props:
   foo: bar
 user-id-value-match-algorithm:
  type: VALUE_MATCH
  props:
   operation: insert
   column: user_id
   value: 0

sql-parser:
 sql-comment-parse-enabled: true
```

**Note**: The default shadow algorithm only supports Hint shadow algorithm. When using ensure that the configuration items of props in the configuration file are less than or equal to those in the SQL comment, And the specific configuration of the configuration file should same as the configuration written in the SQL comment. The fewer configuration items in the configuration file, the looser the matching conditions

```
simple-note-algorithm:
 type: SIMPLE_HINT
 props:
  foo: bar
  foo1: bar1
```

For example, the ‘props’ item have 2 configure, the following syntax can be used in SQL:

```
SELECT * FROM t_xxx_2 WHERE order_id = xxx /*foo:bar, foo1:bar1*/
```

```
SELECT * FROM t_xxx_2 WHERE order_id = xxx /*foo:bar, foo1:bar1, foo2:bar2, ...*/
```

```
simple-note-algorithm:
  type: SIMPLE_HINT
  props:
    foo: bar
```

For example, the ‘props’ item have 1 configure, the following syntax can be used in SQL:

```
SELECT * FROM t_xxx_2 WHERE order_id = xxx /*foo:foo*/
```

```
SELECT * FROM t_xxx_2 WHERE order_id = xxx /*foo:foo, foo1:bar1, ...*/
```

## 11.7 Login Authentication

### 11.7.1 MySQL Authentication Protocol

https://dev.mysql.com/doc/refman/8.0/en/authentication-plugins.html

### 11.7.2 PostgreSQL Authentication Protocol

https://www.postgresql.org/docs/14/auth-methods.html

### 11.7.3 OpenGauss Authentication Protocol

https://opengauss.org/en/docs/2.1.0/docs/Developerguide/configuration-file-reference.html

### 11.7.4 LDAP

https://en.wikipedia.org/wiki/Lightweight_Directory_Access_Protocol

## 11.8 Data Migration

### 11.8.1 Explanation

The current data migration solution uses a completely new database cluster as the migration target.

This implementation has the following advantages:

1. No impact on the original data during migration.
2. No risk in case of migration failure.
3. Free from sharding policy limitations.

The implementation has the following disadvantages:

1. Redundant servers can exist for a certain period of time.
2. All data needs to be moved.

A single data migration mainly consists of the following phases:

1. Preparation.
2. Stock data migration.
3. The synchronization of incremental data.
4. Traffic switching .

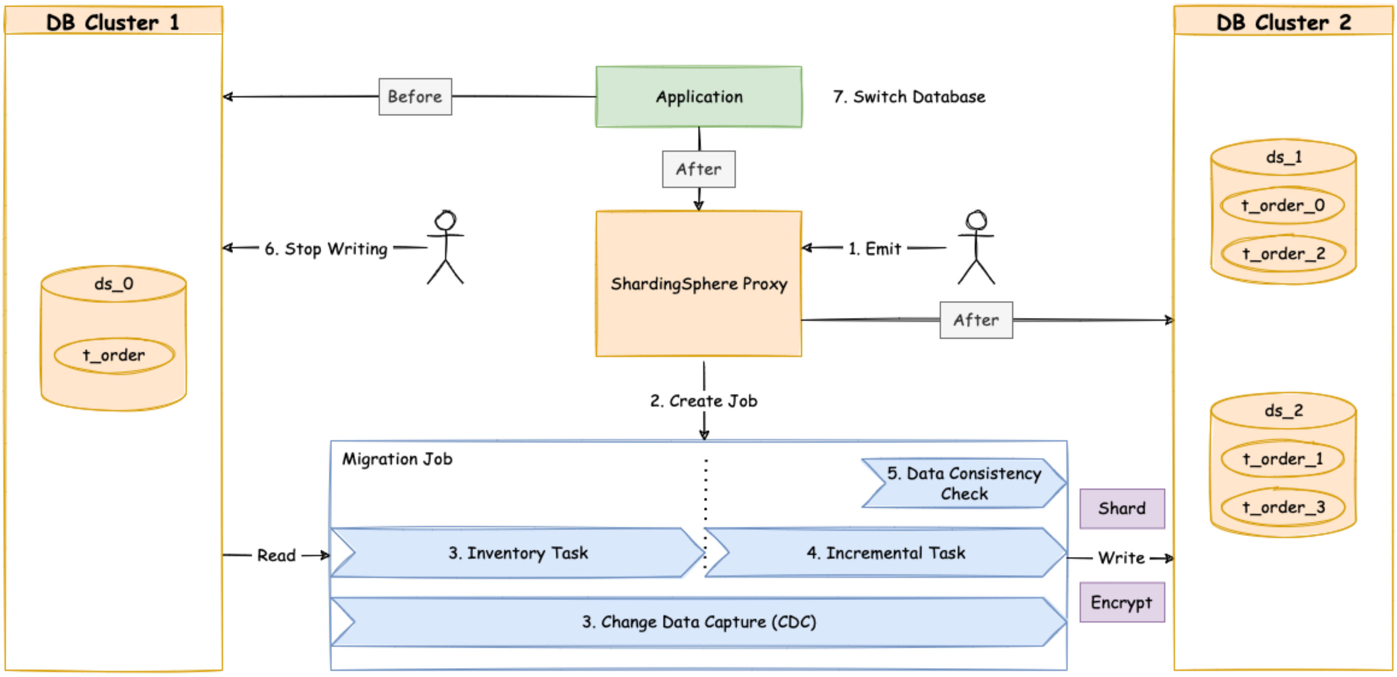


Fig. 29: Illustration

## 11.8.2 Execution Stage Explained

### Preparation

In the preparation stage, the data migration module verifies data source connectivity and permissions, counts stock data statistics, records the log and finally shards the tasks according to data volume and parallelism set by the users.

### Stock data migration

Execute the stock data migration tasks that have been sharded during preparation stage. The stock migration stage uses JDBC queries to read data directly from the source and write into the target based on the sharding rules and other configurations.

#### The Synchronization of incremental data

Since the duration of stock data migration depends on factors such as data volume and parallelism, it is necessary to synchronize the data added to the business operations during this period. Different databases differ in technical details, but in general they are all based on replication protocols or WAL logs to achieve the capture of changed data.

- MySQL: subscribe and parse binlog
- PostgreSQL: uses official logical replication test_decoding.

These incremental data captured are also written into the new data nodes by the data migration modules. When synchronization of incremental data is basically completed (the incremental data flow is not interrupted since the business system is still in function), you can then move to the traffic switching stage.

#### Traffic Switching

During this stage, there may be a read-only period of time, where data in the source data nodes is allowed to be in static mode for a short period of time to ensure that the incremental synchronization can be fully completed. Users can set this by shifting the database to read-only status or by controlling the traffic flow generated from the source.

The length of this read-only window depends on whether users need to perform consistency checks on the data and the exact amount of data in this scenario. Once confirmed, the data migration is complete.

Users can then switch the read traffic or write traffic to Apache ShardingSphere.

### 11.8.3 References

Configurations of data migration

## 11.9 FAQ

### 11.9.1 [Driver] Why there may be an error when configuring both shardingsphere-jdbc-spring-boot-starter and a spring-boot-starter of certain datasource pool(such as druid)?

Answer:

1. Because the spring-boot-starter of certain datasource pool (such as druid) will be configured before shardingsphere-jdbc-spring-boot-starter and create a default datasource, then conflict occur when DBPlusEngine-Driver create datasources.
2. A simple way to solve this issue is removing the spring-boot-starter of certain datasource pool, shardingsphere-jdbc create datasources with suitable pools.

### 11.9.2 [Driver] Why is xsd unable to be found when Spring Namespace is used?

Answer:

The use norm of Spring Namespace does not require to deploy xsd files to the official website. But considering some users' needs, we will deploy them to DBPlusEngine's official website.

Actually, META-INF:raw-latex:spring.schemas in the jar package of shardingsphere-jdbc-spring-namespace has been configured with the position of xsd files: META-INF\namespace\sharding.xsd and META-INF\namespace\replica-query.xsd, so you only need to make sure that the file is in the jar package.

### 11.9.3 [Driver] Found a JtaTransactionManager in spring boot project when integrating with transaction of XA

Answer:

1. shardingsphere-transaction-xa-core include atomikos, it will trigger auto-configuration mechanism in spring-boot, add @SpringBootApplication(exclude = JtaAutoConfiguration.class) will solve it.

### 11.9.4 [Proxy] In a Windows environment, I could not find or load main class org.apache.shardingsphere.proxy.Bootstrap, how to solve it?

Answer:

Some decompression tools may truncate the file name when decompressing the DBPlusEngine-Proxy binary package, resulting in some classes not being found.

The solutions:

Open cmd.exe and execute the following command:

```
tar zxvf apache-shardingsphere-${RELEASE.VERSION}-shardingsphere-proxy-bin.tar.gz
```

### 11.9.5 [Proxy] How to add a new logic schema dynamically when using DBPlusEngine-Proxy?

Answer:

When using DBPlusEngine-Proxy, users can dynamically create or drop logic schema through DistSQL, the syntax is as follows:

```
CREATE (DATABASE | SCHEMA) [IF NOT EXISTS] schemaName;

DROP (DATABASE | SCHEMA) [IF EXISTS] schemaName;
```

Example:

```
CREATE DATABASE sharding_db;

DROP SCHEMA sharding_db;
```

### 11.9.6 [Proxy] How to use suitable database tools when connecting DBPlusEngine-Proxy?

Answer:

1. DBPlusEngine-Proxy could be considered as a mysql sever, so we recommend using mysql command line tool to connect to and operate it.
2. If users would like use a third-party database tool, there may be some errors cause of the certain implementation/options.
3. The currently tested third-party database tools are as follows:
   - Navicat：11.1.13、15.0.20.
   - DataGrip：2020.1、2021.1 (turn on “introspect using jdbc metadata” in idea or datagrip).
   - WorkBench：8.0.25.

### 11.9.7 [Proxy] When using a client such as Navicat to connect to DBPlusEngine-Proxy, if DBPlusEngine-Proxy does not create a Schema or does not add a Resource, the client connection will fail?

Answer:

1. Third-party database tools will send some SQL query metadata when connecting to DBPlusEngine-Proxy. When DBPlusEngine-Proxy does not create a schema or does not add a resource, DBPlusEngine-Proxy cannot execute SQL.
2. It is recommended to create schema and resource first, and then use third-party database tools to connect.
3. **Please refer to** Related

   the details about resource.

### 11.9.8 [Sharding] How to solve Cloud not resolve placeholder …in string value … error?

Answer:

${...} or $->{...} can be used in inline expression identifiers, but the former one clashes with place holders in Spring property files, so $->{...} is recommended to be used in Spring as inline expression identifiers.

### 11.9.9 [Sharding] Why does float number appear in the return result of inline expression?

Answer:

The division result of Java integers is also integer, but in Groovy syntax of inline expression, the division result of integers is float number. To obtain integer division result, A/B needs to be modified as A.intdiv(B).

### 11.9.10 [Sharding] If sharding database is partial, should tables without sharding database & table configured in sharding rules?

Answer:

No, DBPlusEngine will recognize it automatically.

### 11.9.11 [Sharding] When generic Long type SingleKeyTableShardingAlgorithm is used, why doesClassCastException: Integer can not cast to Long exception appear?

Answer:

You must make sure the field in database table consistent with that in sharding algorithms. For example, the field type in database is int(11) and the sharding type corresponds to genetic type is Integer, if you want to configure Long type, please make sure the field type in the database is bigint.

### 11.9.12 [Sharding:raw-latex:PROXY] When implementing the StandardShardingAlgorithm custom algorithm, the specific type of Comparable is specified as Long, and the field type in the database table is bigint, a ClassCastException: Integer can not cast to Long exception occurs.

Answer：

When implementing the doSharding method, it is not recommended to specify the specific type of Comparable in the method declaration, but to convert the type in the implementation of the doSharding method. You can refer to the ModShardingAlgorithm#doSharding method.

### 11.9.13 [Sharding] Why are the default distributed auto-augment key strategy provided by DBPlusEngine not continuous and most of them end with even numbers?

Answer:

ShardingSphere uses snowflake algorithms as the default distributed auto-augment key strategy to make sure unrepeated and decentralized auto-augment sequence is generated under the distributed situations. Therefore, auto-augment keys can be incremental but not continuous.

But the last four numbers of snowflake algorithm are incremental value within one millisecond. Thus, if concurrency degree in one millisecond is not high, the last four numbers are likely to be zero, which explains why the rate of even end number is higher.

In 3.1.0 version, the problem of ending with even numbers has been totally solved, please refer to: https://github.com/apache/shardingsphere/issues/1617

### 11.9.14 [Sharding] How to allow range query with using inline sharding strategy(BETWEEN AND, >, <, >=, <=)?

Answer:

1. Update to 4.1.0 above.
2. Configure(A tip here: then each range query will be broadcast to every sharding table):

- Version 4.x: allow.range.query.with.inline.sharding to true (Default value is false).
- Version 5.x: allow-range-query-with-inline-sharding to true in InlineShardingStrategy (Default value is false).

### 11.9.15 [Sharding] Why does my custom distributed primary key do not work after implementing KeyGenerateAlgorithm interface and configuring type property?

Answer:

Service Provider Interface (SPI) is a kind of API for the third party to implement or expand. Except implementing interface, you also need to create a corresponding file in META-INF/services to make the JVM load these SPI implementations.

More detail for SPI usage, please search by yourself.

Other DBPlusEngine functionality implementation will take effect in the same way.

### 11.9.16 [Sharding] In addition to internal distributed primary key, does ShardingSphere support other native auto-increment keys?

Answer:

Yes. But there is restriction to the use of native auto-increment keys, which means they cannot be used as sharding keys at the same time.

Since DBPlusEngine does not have the database table structure and native auto-increment key is not included in original SQL, it cannot parse that field to the sharding field. If the auto-increment key is not sharding key, it can be returned normally and is needless to be cared. But if the auto-increment key is also used as sharding key, DBPlusEngine cannot parse its sharding value, which will make SQL routed to multiple tables and influence the rightness of the application.

The premise for returning native auto-increment key is that INSERT SQL is eventually routed to one table. Therefore, auto-increment key will return zero when INSERT SQL returns multiple tables.

### 11.9.17 [Encryption] How to solve that data encryption can' t work with JPA?

Answer:

Because DDL for data encryption has not yet finished, JPA Entity cannot meet the DDL and DML at the same time, when JPA that automatically generates DDL is used with data encryption.

The solutions are as follows:

1. Create JPA Entity with logicColumn which needs to encrypt.
2. Disable JPA auto-ddl, For example setting auto-ddl=none.
3. Create table manually. Table structure should use cipherColumn,plainColumn and assistedQueryColumn to replace the logicColumn.

### 11.9.18 [DistSQL] How to set custom JDBC connection properties or connection pool properties when adding a data source using DistSQL?

Answer:

1. If you need to customize JDBC connection properties, please take the urlSource way to define dataSource.
2. DBPlusEngine presets necessary connection pool properties, such as maxPoolSize, idleTimeout, etc. If you need to add or overwrite the properties, please specify it with PROPERTIES in the dataSource.
3. Please refer to Related introduction for above rules.

### 11.9.19 [DistSQL] How to solve Resource [xxx] is still used by [SingleTableRule]. exception when dropping a data source using DistSQL?

Answer：

1. Resources referenced by rules cannot be deleted
2. If the resource is only referenced by single table rule, and the user confirms that the restriction can be ignored, the optional parameter ignore single tables can be added to perform forced deletion

```
UNREGISTER STORAGE UNIT dataSourceName [, dataSourceName] ... [ignore single tables]
```

### 11.9.20 [DistSQL] How to solve Failed to get driver instance for jdbcURL=xxx. exception when adding a data source using DistSQL?

Answer：

DBPlusEngine Proxy do not have jdbc driver during deployment. Some example of this include mysql-connector. To use it otherwise following syntax can be used:

```
REGISTER STORAGE UNIT dataSourceName [..., dataSourceName]
```

### 11.9.21 [Other] How to debug when SQL can not be executed rightly in DBPlusEngine?

Answer:

sql.show configuration is provided in DBPlusEngine-Proxy and post-1.5.0 version of DBPlusEngine-Driver, enabling the context parsing, rewritten SQL and the routed data source printed to info log. sql.show configuration is off in default, and users can turn it on in configurations.

A Tip: Property sql.show has changed to sql-show in version 5.x.

### 11.9.22 [Other] Why do some compiling errors appear? Why did not the IDEA index the generated codes?

Answer:

DBPlusEngine uses lombok to enable minimal coding. For more details about using and installment, please refer to the official website of lombok.

The codes under the package org.apache.shardingsphere.sql.parser.autogen are generated by ANTLR. You may execute the following command to generate codes:

```
./mvnw -Dcheckstyle.skip=true -Drat.skip=true -Dmaven.javadoc.skip=true -Djacoco.skip=true -DskipITs -DskipTests install -T1C
```

The generated codes such as org.apache.shardingsphere.sql.parser.autogen.PostgreSQLStatementParser may be too large to be indexed by the IDEA. You may configure the IDEA’s property idea.max.intellisense.filesize=10000.

### 11.9.23 [Other] In SQLSever and PostgreSQL, why does the aggregation column without alias throw exception?

Answer:

SQLServer and PostgreSQL will rename aggregation columns acquired without alias, such as the following SQL:

```
SELECT SUM(num), SUM(num2) FROM tablexxx;
```

Columns acquired by SQLServer are empty string and (2); columns acquired by PostgreSQL are empty sum and sum(2). It will cause error because DBPlusEngine is unable to find the corresponding column.

The right SQL should be written as:

```
SELECT SUM(num) AS sum_num, SUM(num2) AS sum_num2 FROM tablexxx;
```

### 11.9.24 [Other] Why does Oracle database throw “Order by value must implements Comparable” exception when using Timestamp Order By?

Answer:

There are two solutions for the above problem: 1. Configure JVM parameter “-oracle.jdbc.J2EE13Compliant=true” 2. Set System.getProperties().setProperty( “oracle.jdbc.J2EE13Compliant” , “true” ) codes in the initialization of the project.

Reasons:

org.apache.shardingsphere.sharding.merge.dql.orderby.OrderByValue#getOrderValues():

```
private List<Comparable<?>> getOrderValues() throws SQLException {
  List<Comparable<?>> result = new ArrayList<>(orderByItems.size());
  for (OrderByItem each : orderByItems) {
    Object value = queryResult.getValue(each.getIndex(), Object.class);
    Preconditions.checkState(null == value || value instanceof Comparable, "Order by value must implements Comparable");
    result.add((Comparable<?>) value);
  }
  return result;
}
```

After using resultSet.getObject(int index), for TimeStamp oracle, the system will decide whether to return java.sql.TimeStamp or define oralce.sql.TIMESTAMP according to the property of oracle.jdbc.J2EE13Compliant. See oracle.jdbc.driver.TimestampAccessor#getObject(int var1) method in ojdbc codes for more detail:

```
Object getObject(int var1) throws SQLException {
  Object var2 = null;
  if(this.rowSpaceIndicator == null) {
    DatabaseError.throwSqlException(21);
  }

  if(this.rowSpaceIndicator[this.indicatorIndex + var1] != -1) {
    if(this.externalType != 0) {
      switch(this.externalType) {
      case 93:
        return this.getTimestamp(var1);
      default:
        DatabaseError.throwSqlException(4);
        return null;
      }
    }

    if(this.statement.connection.j2ee13Compliant) {
      var2 = this.getTimestamp(var1);
    } else {
      var2 = this.getTIMESTAMP(var1);
    }
  }

  return var2;
}
```

### 11.9.25 [Other] In Windows environment,when cloning DBPlusEngine source code through Git, why prompt filename too long and how to solve it?

Answer:

To ensure the readability of source code,the DBPlusEngine Coding Specification requires that the naming of classes,methods and variables be literal and avoid abbreviations,which may result in Some source files have long names.

Since the Git version of Windows is compiled using msys,it uses the old version of Windows Api,limiting the file name to no more than 260 characters.

The solutions are as follows:

Open cmd.exe (you need to add git to environment variables) and execute the following command to allow git supporting log paths:

```
git config --global core.longpaths true
```

If we use windows 10, also need enable win32 log paths in registry editor or group strategy(need reboot): > Create the registry key HKLM\SYSTEM\CurrentControlSet\Control\FileSystem LongPathsEnabled (Type: REG_DWORD) in registry editor, and be set to 1. > Or click "setting" button in system menu, print "Group Policy" to open a new window "Edit Group Policy", and then click 'Computer Configuration' > 'Administrative Templates' > 'System' > 'Filesystem', and then turn on 'Enable Win32 long paths' option.

Reference material:

https://docs.microsoft.com/zh-cn/windows/desktop/FileIO/naming-a-file https://ourcodeworld.com/articles/read/109/how-to-solve-filename-too-long-error-in-git-powershell-and-github-application-for-windows

### 11.9.26 [Other] How to solve Type is required error?

Answer:

In DBPlusEngine, many functionality implementation are uploaded through SPI, such as Distributed Primary Key. These functions load SPI implementation by configuring the type， so the type must be specified in the configuration file.

### 11.9.27 [Other] How to speed up the metadata loading when service starts up?

Answer:

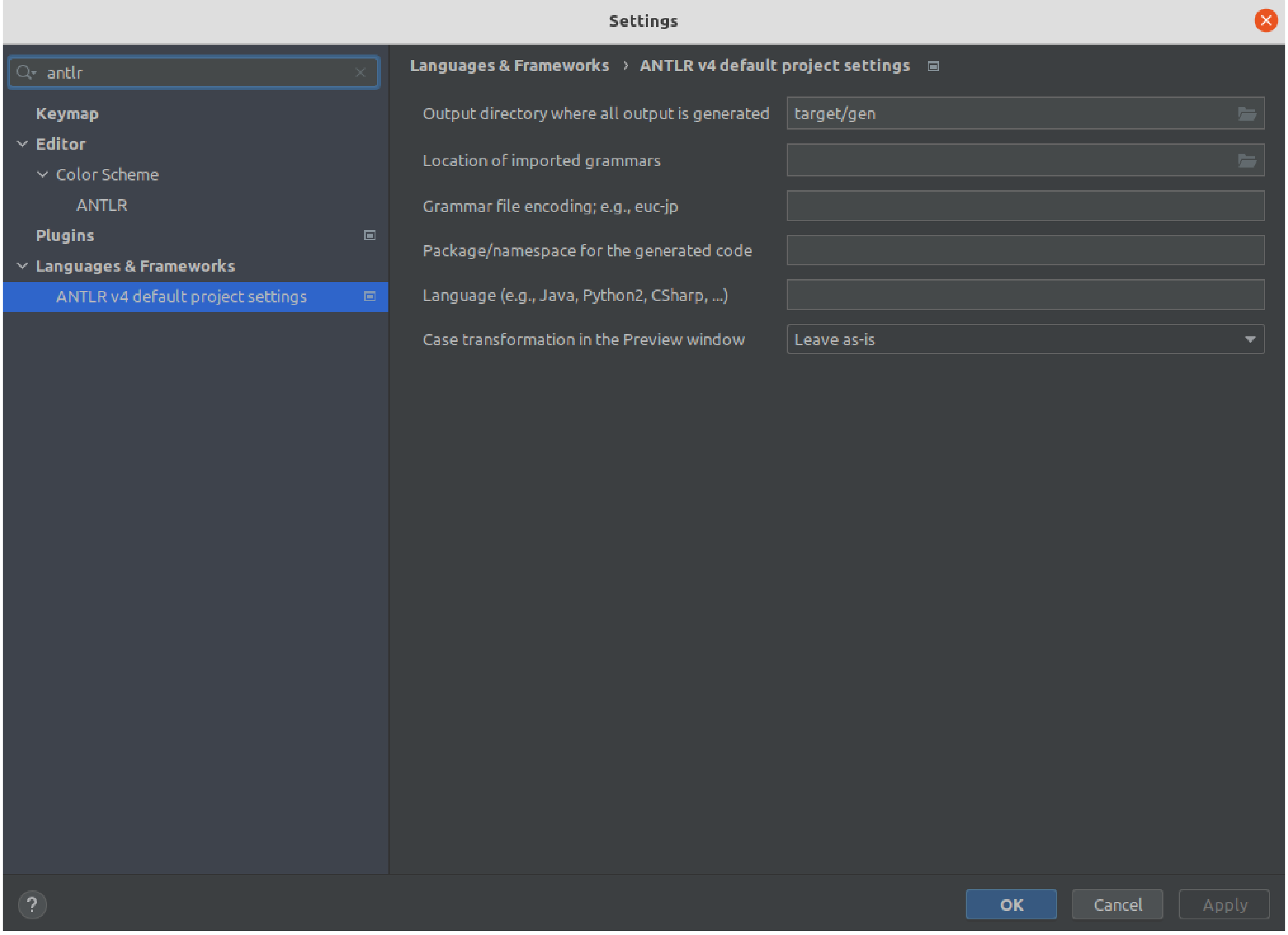
1. Update to 4.0.1 above, which helps speed up the process of loading table metadata.
2. Configure:

- max.connections.size.per.query (Default value is 1) higher referring to connection pool you adopt(Version >= 3.0.0.M3 & Version < 5.0.0).
- max-connections-size-per-query (Default value is 1) higher referring to connection pool you adopt(Version >= 5.0.0).

### 11.9.28 [Other] The ANTLR plugin generates codes in the same level directory as src, which is easy to commit by mistake. How to avoid it?

Answer:

Goto Settings -> Languages & Frameworks -> ANTLR v4 default project settings and configure the output directory of the generated code as target/gen as shown:



Languages & Frameworks › ANTLR v4 default project settings

Output directory where all output is generated: target/gen

Location of imported grammars:

Grammar file encoding; e.g., euc-jp:

Package/namespace for the generated code:

Language (e.g., Java, Python2, CSharp, ...):

Case transformation in the Preview window: Leave as-is

Fig. 30: Configure ANTLR plugin

### 11.9.29 [Other] Why is the database sharding result not correct when using Proxool?

Answer:

When using Proxool to configure multiple data sources, each one of them should be configured with alias. It is because Proxool would check whether existing alias is included in the connection pool or not when acquiring connections, so without alias, each connection will be acquired from the same data source.

The followings are core codes from ProxoolDataSource getConnection method in Proxool:

```
if(!ConnectionPoolManager.getInstance().isPoolExists(this.alias)) {
   this.registerPool();
}
```

For more alias usages, please refer to Proxool official website.

### 11.9.30 [Other] The property settings in the configuration file do not take effect when integrating DBPlusEngine with Spring Boot 2.x ?

Answer:

Note that the property name in the Spring Boot 2.x environment is constrained to allow only lowercase letters, numbers and short transverse lines, [a-z][0-9] and -.

Reasons:

In the Spring Boot 2.x environment, DBPlusEngine binds the properties through Binder, and the unsatisfied property name (such as camel case or underscore.) can throw a NullPointerException exception when the property setting does not work to check the property value. Refer to the following error examples:

Underscore case: database_inline

```
spring.shardingsphere.rules.sharding.sharding-algorithms.database_inline.type=INLINE
spring.shardingsphere.rules.sharding.sharding-algorithms.database_inline.props.algorithm-expression=ds-$->{user_id % 2}
```

```
Caused by: org.springframework.beans.factory.BeanCreationException: Error creating bean with name 'database_inline':
Initialization of bean failed; nested exception is java.lang.NullPointerException: Inline sharding algorithm expression cannot be
null.
  ...
Caused by: java.lang.NullPointerException: Inline sharding algorithm expression cannot be null.
  at com.google.common.base.Preconditions.checkNotNull(Preconditions.java:897)
  at org.apache.shardingsphere.sharding.algorithm.sharding.inline.InlineShardingAlgorithm.
getAlgorithmExpression(InlineShardingAlgorithm.java:58)
  at org.apache.shardingsphere.sharding.algorithm.sharding.inline.InlineShardingAlgorithm.init(InlineShardingAlgorithm.
java:52)
  at org.apache.shardingsphere.spring.boot.registry.AbstractAlgorithmProvidedBeanRegistry.
postProcessAfterInitialization(AbstractAlgorithmProvidedBeanRegistry.java:98)
  at org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.
applyBeanPostProcessorsAfterInitialization(AbstractAutowireCapableBeanFactory.java:431)
  ...
```

Camel case： databaseInline

```
spring.shardingsphere.rules.sharding.sharding-algorithms.databaseInline.type=INLINE
spring.shardingsphere.rules.sharding.sharding-algorithms.databaseInline.props.algorithm-expression=ds-$->{user_id % 2}
```

```
Caused by: org.springframework.beans.factory.BeanCreationException: Error creating bean with name 'databaseInline':
Initialization of bean failed; nested exception is java.lang.NullPointerException: Inline sharding algorithm expression cannot be
null.
  ...
Caused by: java.lang.NullPointerException: Inline sharding algorithm expression cannot be null.
  at com.google.common.base.Preconditions.checkNotNull(Preconditions.java:897)
  at org.apache.shardingsphere.sharding.algorithm.sharding.inline.InlineShardingAlgorithm.
getAlgorithmExpression(InlineShardingAlgorithm.java:58)
  at org.apache.shardingsphere.sharding.algorithm.sharding.inline.InlineShardingAlgorithm.init(InlineShardingAlgorithm.
java:52)
  at org.apache.shardingsphere.spring.boot.registry.AbstractAlgorithmProvidedBeanRegistry.
postProcessAfterInitialization(AbstractAlgorithmProvidedBeanRegistry.java:98)
  at org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.
applyBeanPostProcessorsAfterInitialization(AbstractAutowireCapableBeanFactory.java:431)
  ...
```

From the exception stack, the AbstractAlgorithmProvidedBeanRegistry.registerBean method calls PropertyUtil.containPropertyPrefix (environment, prefix) , and PropertyUtil.containPropertyPrefix (environment, prefix) determines that the configuration of the specified prefix does not exist, while the method uses Binder in an unsatisfied property name (such as camelcase or underscore) causing property settings does not to take effect.

# 12 Obtain

Contact Us in order to get the DBPlusEngine trial version.