ShardingSphere

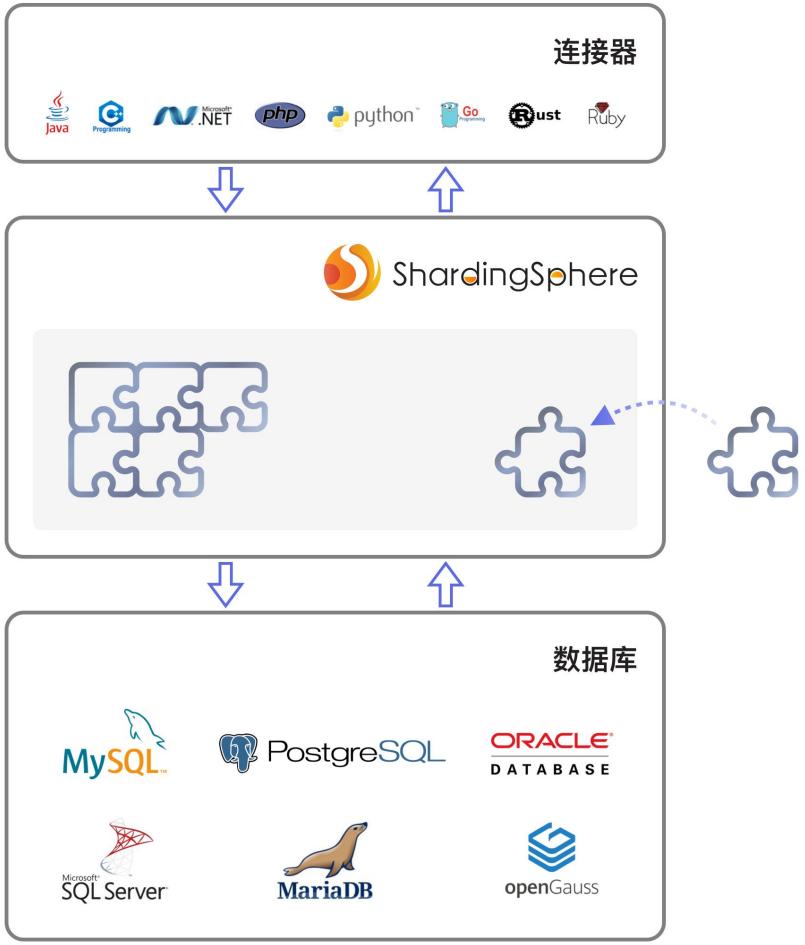
DistSQL 体系解读及分片场景实践

江龙滔

2022.07.23

目录

ShardingSphere 产品设计理念



Database Plus：一种分布式数据库系统的设计理念。通过在碎片化的同构或异构数据库之上搭建使用和交互的标准层和生态层，并叠加扩展更多计算能力，例如数据分片、数据加解密等，使得所有应用和数据库之间的交互面向 Database Plus 构建的标准层，从而屏蔽数据库碎片化对上层业务带来的差异化影响。

连接

连接数据与应用，关注异构数据库之间的协作

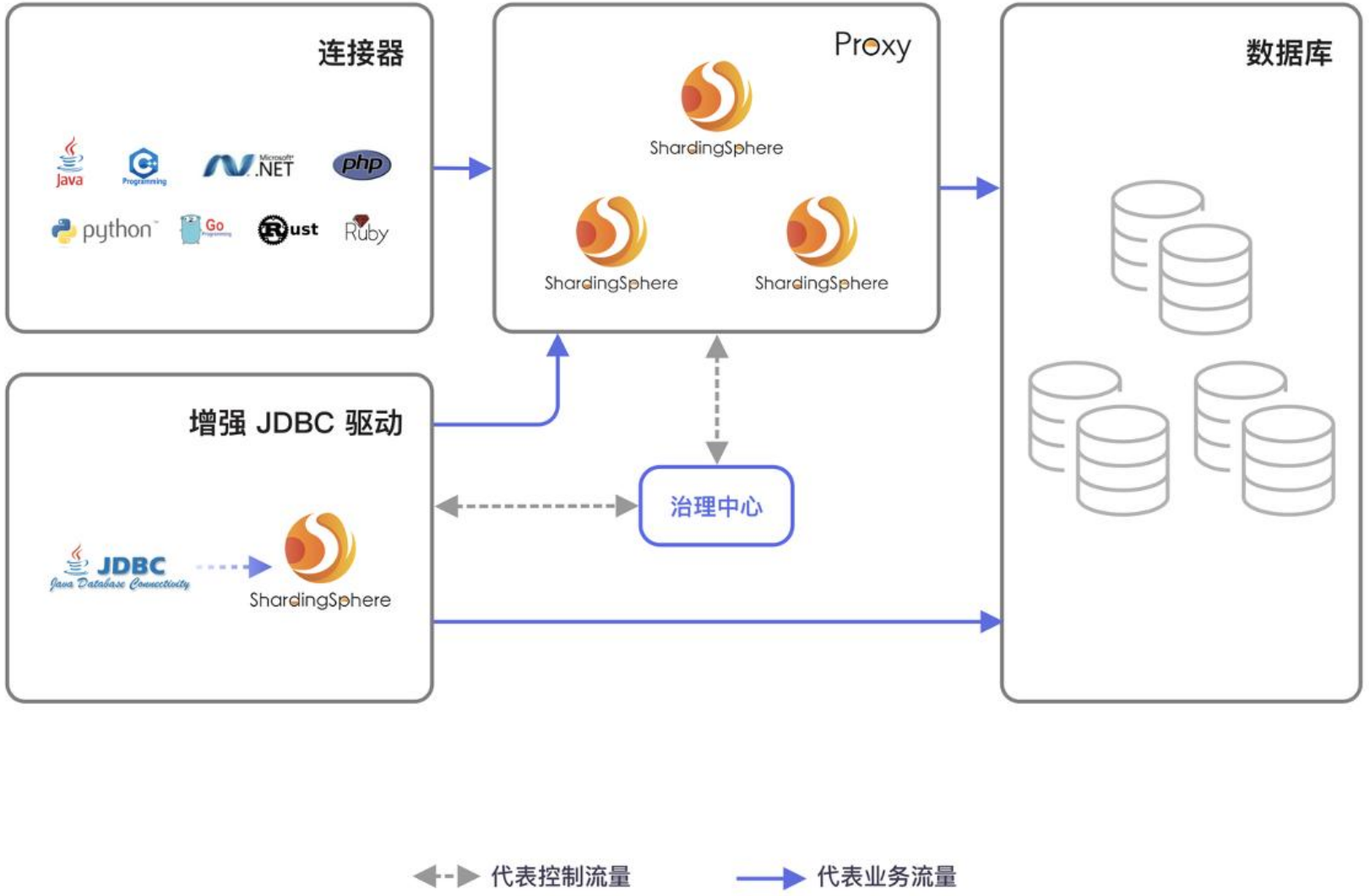
增强

为数据库数据计算提供功能增强服务

可插拔

采用可插拔架构设计，模块相互独立，可以单独使用也可以自由组合各模块使用。

ShardingSphere 物理部署图



前情提要

DistSQL：像数据库一样使用 Apache ShardingSphere

2021/07/20

6. 如何在 ShardingSphere 中开发自己的 DistSQL

2021/11/30

分片利器 AutoTable：为用户带来「管家式」分片配置体验

2021/09/10

8. 特性更新！DistSQL 集群治理能力详解

03/23

4. SCTL 涅槃重生：投入 RAL 的怀抱

2021/11/17

5. DistSQL 深度解析：打造动态化的分布式数据库

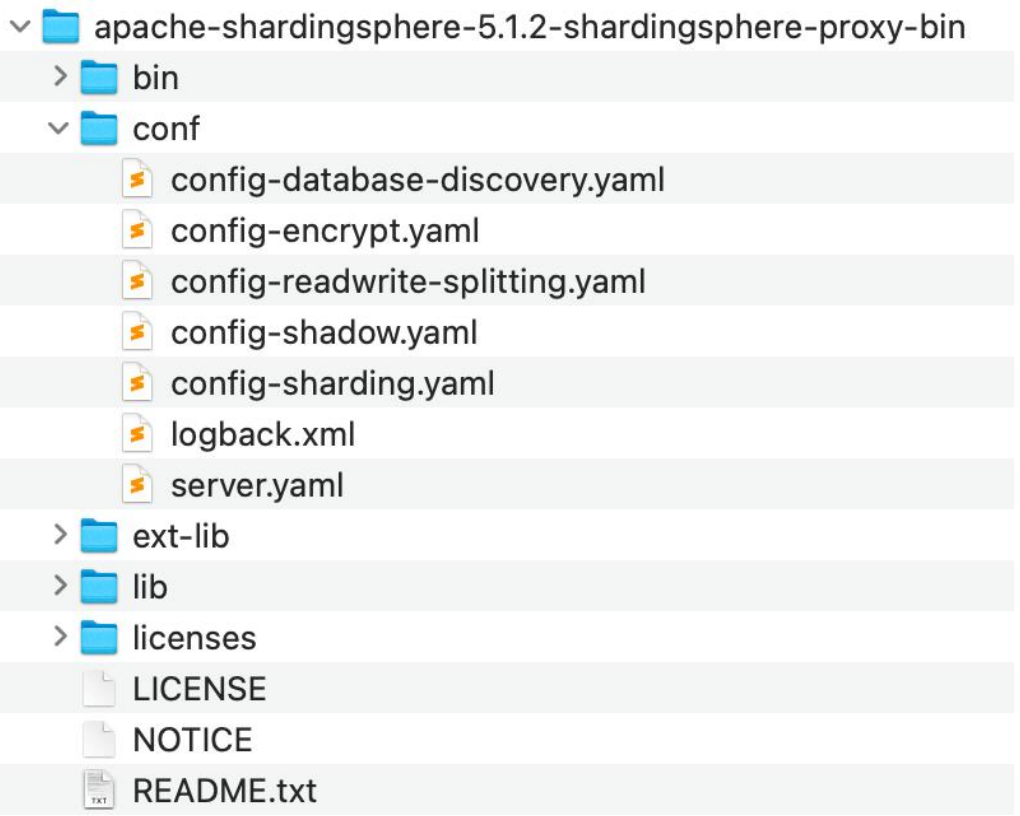
3天前

ShardingSphere官微

ShardingSphere 官方更新。ShardingSphere 是一款开源分布式数据库生态项目，旨在碎片化的异构数据库上层构建生态，最大限度复用数据库原生存算能力的前提下，进一步提供面向全局的扩展和叠加计算能力。

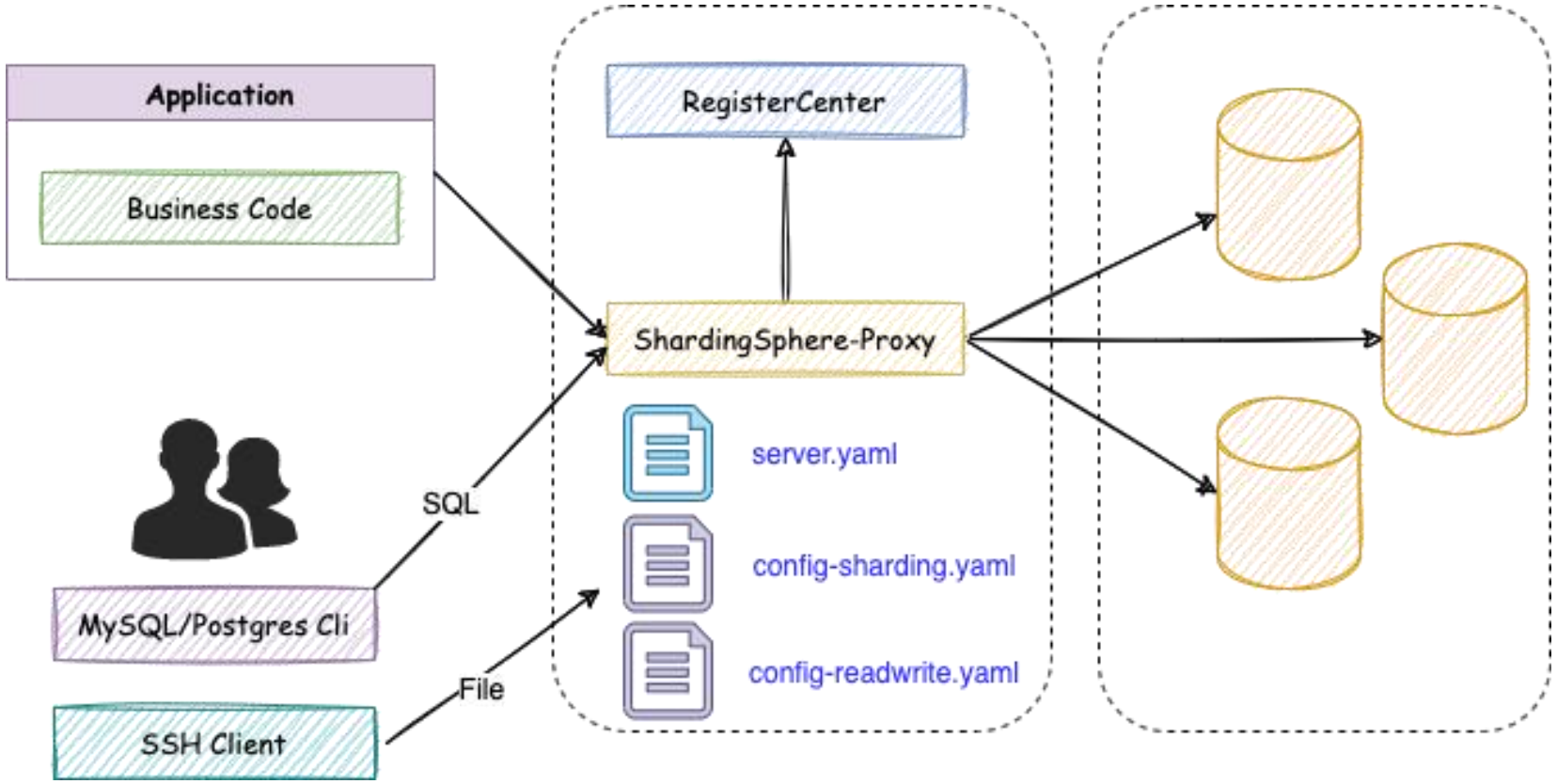
目录

使用 YAML 配置



```yaml
databaseName: sharding_db

dataSources:
  ds_0:
    url: jdbc:mysql://127.0.0.1:3306/demo_ds_0?serverTimezone=UTC&useSSL=false
    username: root
    password: 123456
  #  ...
  ds_1:
    url: jdbc:mysql://127.0.0.1:3306/demo_ds_1?serverTimezone=UTC&useSSL=false
    username: root
    password: 123456
  #  ...

rules:
- !SHARDING
  tables:
    t_order:
      actualDataNodes: ds_${0..1}.t_order_${0..1}
      databaseStrategy:
        standard:
          shardingColumn: user_id
          shardingAlgorithmName: database_inline
      tableStrategy:
        standard:
          shardingColumn: order_id
          shardingAlgorithmName: t_order_inline
  shardingAlgorithms:
    database_inline:
      type: INLINE
      props:
        algorithm-expression: ds_${user_id % 2}
    t_order_inline:
      type: INLINE
      props:
        algorithm-expression: t_order_${order_id % 2}

```

使用 YAML 配置



痛点：

- 通过 YAML 维护存储资源
- 通过 YAML 文件管理规则
- 多个 database 需要多个 YAML
- 查看配置需要登录服务器
- 修改配置需要服务器写权限
- 更新配置需要重启 Proxy

什么是 DistSQL

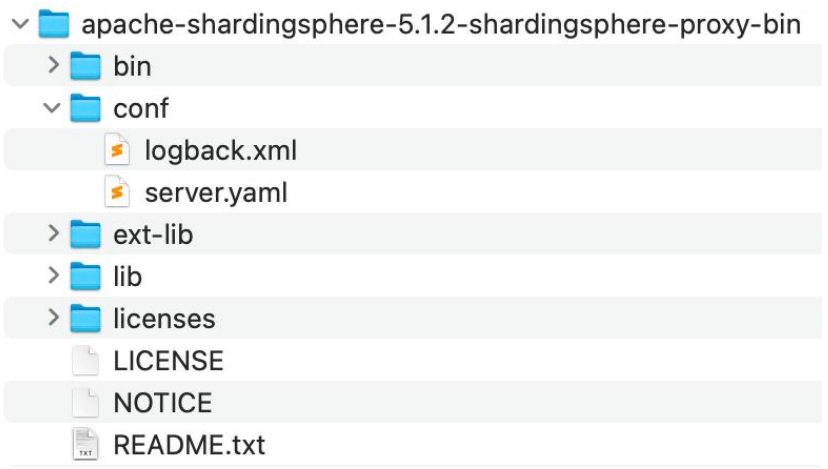
定义

DistSQL（Distributed SQL）是 Apache ShardingSphere 特有的内置 SQL 语言，提供了标准 SQL 之外的增量功能操作能力。

目标

打破中间件和数据库之间的界限，让开发者像使用数据库一样使用 Apache ShardingSphere，是 DistSQL 的设计目标。

使用 DistSQL 配置



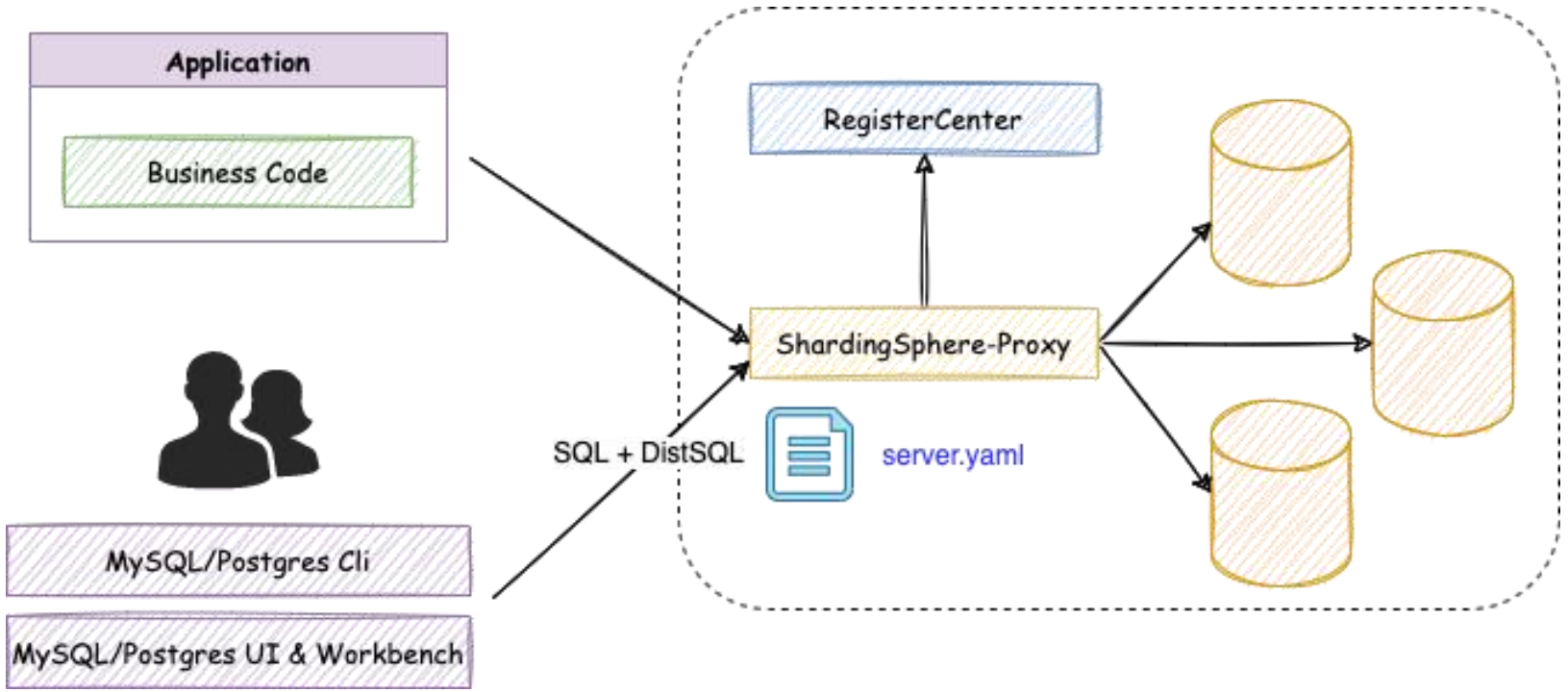

```
mysql> CREATE DATABASE sharding_db;
Query OK, 0 rows affected (0.03 sec)

mysql> USE sharding_db;
Database changed
mysql> ADD RESOURCE ds_0 (
    ->   HOST=127.0.0.1,
    ->   PORT=3306,
    ->   DB=demo_ds_0,
    ->   USER=root,
    ->   PASSWORD=123456
    -> ), ds_1 (
    ->   URL="jdbc:mysql://127.0.0.1:3306/demo_ds_1?serverTimezone=UTC&useSSL=false",
    ->   USER=root,
    ->   PASSWORD=123456
    -> );
Query OK, 0 rows affected (0.09 sec)

mysql> CREATE SHARDING TABLE RULE t_order (
    ->   RESOURCES(ds_0, ds_1),
    ->   SHARDING_COLUMN=order_id,
    ->   TYPE(NAME=MOD,PROPERTIES("sharding-count"=4))
    -> );
Query OK, 0 rows affected (0.06 sec)

mysql>
```

```yaml
databaseName: sharding_db

dataSources:
  ds_0:
    url: jdbc:mysql://127.0.0.1:3306/demo_ds_0?serverTimezone=UTC&useSSL=false
    username: root
    password: 123456
  #  ...
  ds_1:
    url: jdbc:mysql://127.0.0.1:3306/demo_ds_1?serverTimezone=UTC&useSSL=false
    username: root
    password: 123456
  #  ...

rules:
- !SHARDING
  tables:
    t_order:
      actualDataNodes: ds_${0..1}.t_order_${0..1}
      databaseStrategy:
        standard:
          shardingColumn: user_id
          shardingAlgorithmName: database_inline
      tableStrategy:
        standard:
          shardingColumn: order_id
          shardingAlgorithmName: t_order_inline
  shardingAlgorithms:
    database_inline:
      type: INLINE
      props:
        algorithm-expression: ds_${user_id % 2}
    t_order_inline:
      type: INLINE
      props:
        algorithm-expression: t_order_${order_id % 2}

```

```
mysql> CREATE DATABASE sharding_db;
Query OK, 0 rows affected (0.03 sec)

mysql> USE sharding_db;
Database changed
mysql> ADD RESOURCE ds_0 (
    ->   HOST=127.0.0.1,
    ->   PORT=3306,
    ->   DB=demo_ds_0,
    ->   USER=root,
    ->   PASSWORD=123456
    -> ), ds_1 (
    ->   URL="jdbc:mysql://127.0.0.1:3306/demo_ds_1?serverTimezone=UTC&useSSL=false",
    ->   USER=root,
    ->   PASSWORD=123456
    -> );
Query OK, 0 rows affected (0.09 sec)

mysql> CREATE SHARDING TABLE RULE t_order (
    ->   RESOURCES(ds_0, ds_1),
    ->   SHARDING_COLUMN=order_id,
    ->   TYPE(NAME=MOD,PROPERTIES("sharding-count"=4))
    -> );
Query OK, 0 rows affected (0.06 sec)

mysql>
```

```yaml
- !SHARDING
  autoTables:
    t_order:
      actualDataNodes: ds_0.t_order_0,ds_1.t_order_1,ds_0.t_order_2,ds_1.t_order_3
      actualDataSources: ds_0,ds_1
      logicTable: t_order
      shardingStrategy:
        standard:
          shardingAlgorithmName: t_order_mod
          shardingColumn: order_id
  shardingAlgorithms:
    t_order_mod:
      props:
        sharding-count: '4'
      type: mod
- !SINGLE {}
```

DistSQL 配置



亮点：

- 通过 DistSQL 维护存储资源
- 通过 DistSQL 管理权限和规则
- 只有一个 server.yaml 文件
- 通过 SQL 客户端查询规则
- 修改配置不需要文件权限
- 更新配置不需要重启 Proxy

目录

DistSQL 分类

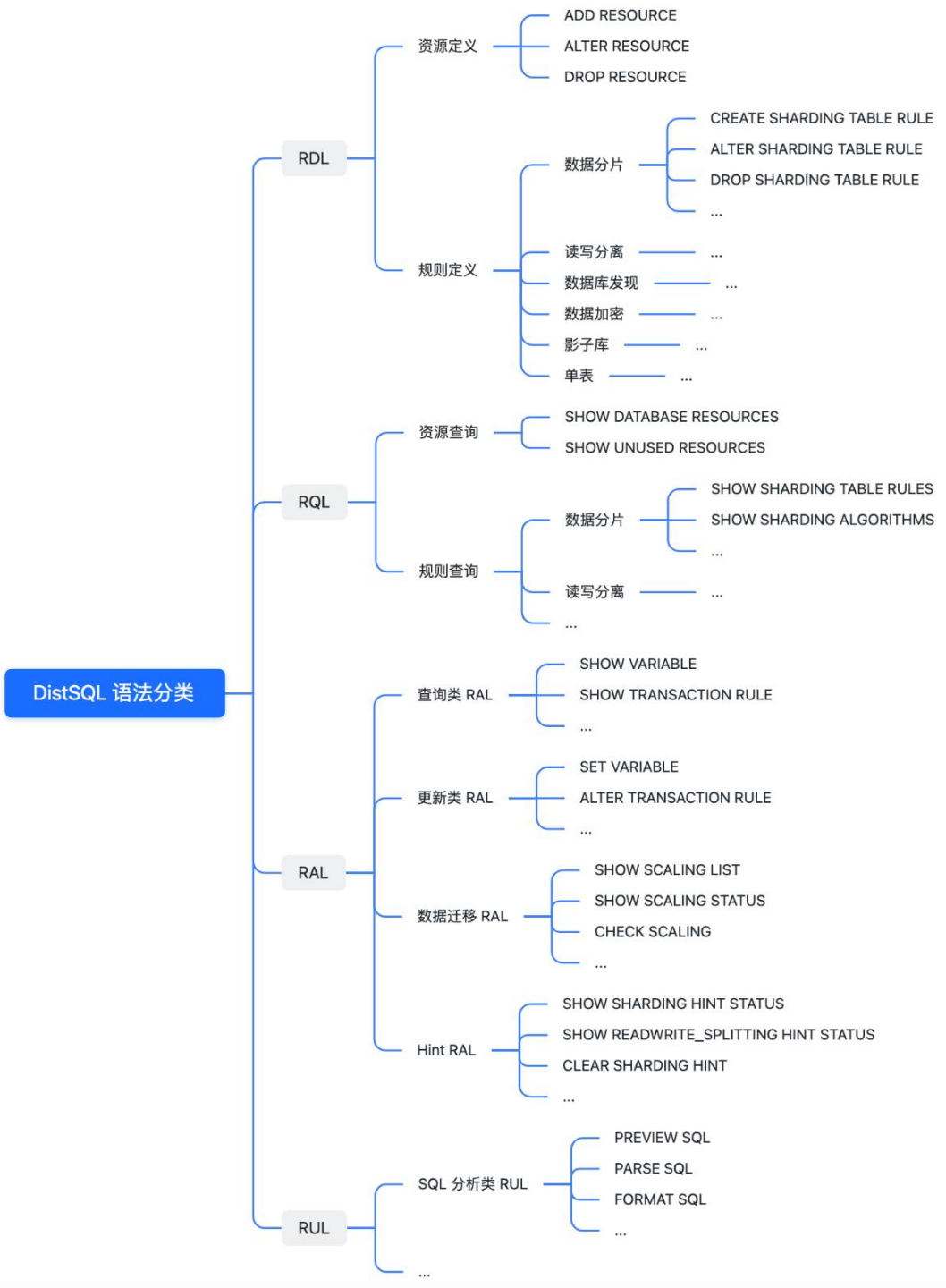
在 ShardingSphere 中， DistSQL 的语法目前主要划分为 RDL、RQL 和 RAL 三种类型：

- RDL（Resource & Rule Definition Language）负责资源和规则的创建、修改和删除；
- RQL（Resource & Rule Query Language）负责资源和规则的查询和展现；
- RAL（Resource & Rule Administration Language）负责Hint、事务类型切换、熔断控制、数据迁移、配置导入导出等管理功能的操作。

此外，新的 DistSQL 类型 RUL 已在设计中：

- RUL（Resource Utility Language）负责 SQL 解析、SQL 格式化、路由预览等 SQL 分析能力，未来还会加入更多辅助语法帮助用户使用 DistSQL。

DistSQL 分类


DistSQL 语法分类
RDL
资源定义
ADD RESOURCE
ALTER RESOURCE
DROP RESOURCE
规则定义
数据分片
CREATE SHARDING TABLE RULE
ALTER SHARDING TABLE RULE
DROP SHARDING TABLE RULE
...
读写分离
...
数据库发现
...
数据加密
...
影子库
...
单表
...
RQL
资源查询
SHOW DATABASE RESOURCES
SHOW UNUSED RESOURCES
规则查询
数据分片
SHOW SHARDING TABLE RULES
SHOW SHARDING ALGORITHMS
...
读写分离
...
...
RAL
查询类 RAL
SHOW VARIABLE
SHOW TRANSACTION RULE
...
更新类 RAL
SET VARIABLE
ALTER TRANSACTION RULE
...
数据迁移 RAL
SHOW SCALING LIST
SHOW SCALING STATUS
CHECK SCALING
...
Hint RAL
SHOW SHARDING HINT STATUS
SHOW READWRITE_SPLITTING HINT STATUS
CLEAR SHARDING HINT
...
RUL
SQL 分析类 RUL
PREVIEW SQL
PARSE SQL
FORMAT SQL
...
...

覆盖全部 Feature

ShardingSphere

读写分离

```
// Static
CREATE READWRITE_SPL
WRITE_RESOURCE=write
READ_RESOURCES(read_
TYPE(NAME=RANDOM)
);

// Dynamic
CREATE READWRITE_SPL
AUTO_AWARE_RESOURCE=
TYPE(NAME=RANDOM)
);

ALTER READWRITE_SPLI
WRITE_RESOURCE=write
READ_RESOURCES(read_
TYPE(NAME=RANDOM)
);

DROP READWRITE_SPLIT
```

数据库发现

```
CREATE DB_DI
RESOURCES(ds_
TYPE(NAME='M
HEARTBEAT(PR
);

ALTER DB_DIS
RESOURCES(ds_
TYPE(NAME='M
HEARTBEAT(PR
);

DROP DB_DISC

DROP DB_DISC

DROP DB_DISC
```

数据加密

```
CREATE ENC
COLUMNS(
(NAME=user
ERTIES('ae
(NAME=orde
)));

ALTER ENCR
COLUMNS(
(NAME=user
ERTIES('ae
(NAME=orde
));

DROP ENCRY
```

影子库压测

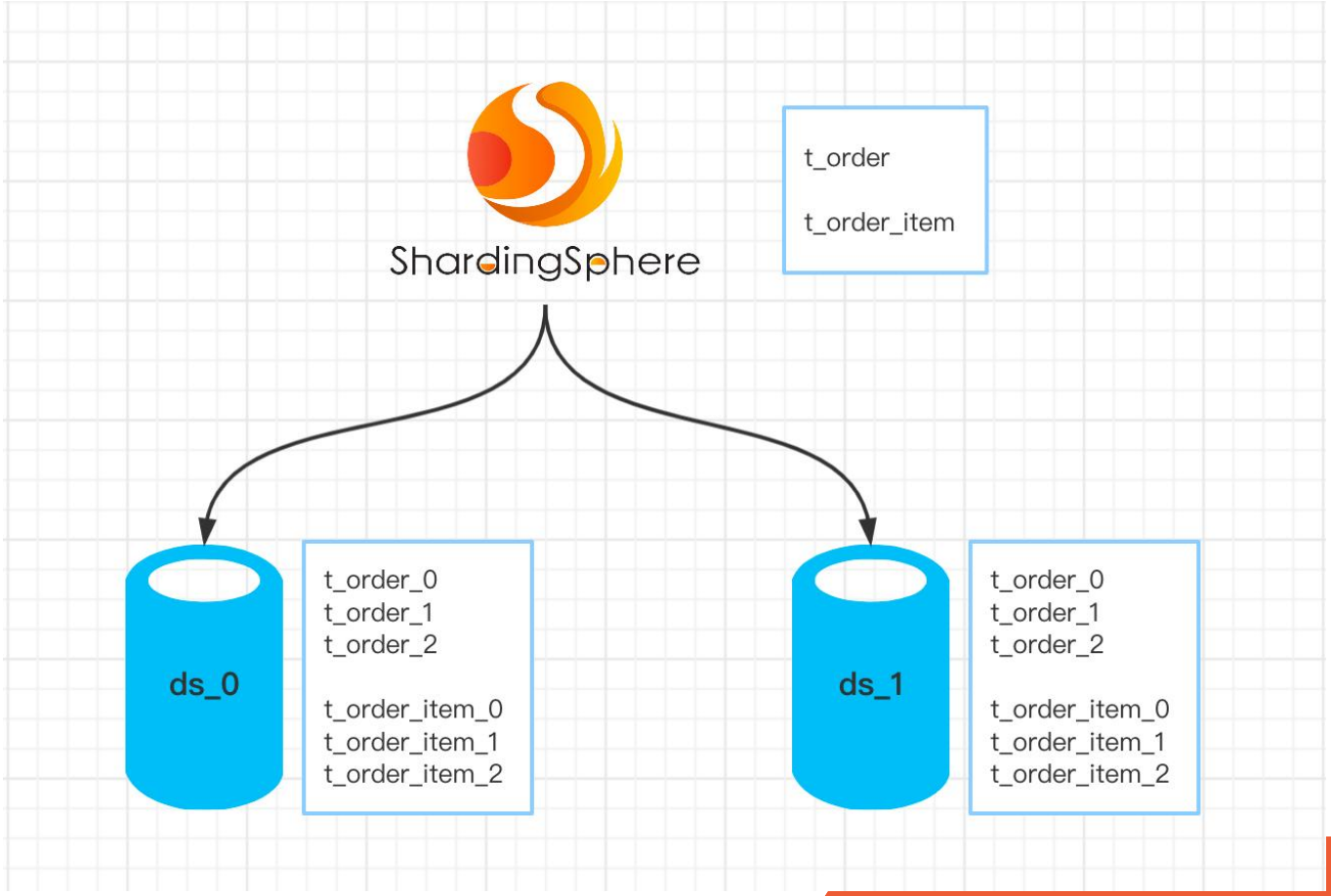
```
CREATE SHADOW RULE shadow_rule(
SOURCE=demo_ds,
SHADOW=demo_ds_shadow,
t_order((simple_note_algorithm, TYPE(NAME=SIMPLE_NOTE, PROPERTIES("shadow"="true",
foo="bar"))),(TYPE(NAME=COLUMN_REGEX_MATCH,
PROPERTIES("operation"="insert","column"="user_id", "regex"='[1]'))),
t_order_item((TYPE(NAME=SIMPLE_NOTE, PROPERTIES("shadow"="true", "foo"="bar")))));

ALTER SHADOW RULE shadow_rule(
SOURCE=demo_ds,
SHADOW=demo_ds_shadow,
t_order((simple_note_algorithm, TYPE(NAME=SIMPLE_NOTE, PROPERTIES("shadow"="true",
foo="bar"))),(TYPE(NAME=COLUMN_REGEX_MATCH,
PROPERTIES("operation"="insert","column"="user_id", "regex"='[1]'))),
t_order_item((TYPE(NAME=SIMPLE_NOTE, PROPERTIES("shadow"="true", "foo"="bar")))));

ALTER SHADOW ALGORITHM
(simple_note_algorithm, TYPE(NAME=SIMPLE_NOTE, PROPERTIES("shadow"="true", "foo"="bar"))
(user_id_match_algorithm, TYPE(NAME=COLUMN_REGEX_MATCH,PROPERTIES("operation"="insert",
"column"="user_id", "regex"='[1]')));

DROP SHADOW RULE shadow_rule;

DROP SHADOW ALGORITHM simple_note_algorithm;
```

目录

分片场景实践

需求：

- 创建两张分片表 `t_order` 和 `t_order_item`；
- 两张表均以 `user_id` 字段分库，以 `order_id` 字段分表；
- 分片数量为 2 库 x 3 表；



创建分布式数据库&添加存储资源

```
-- 1. Create Database
CREATE DATABASE sharding_db;
USE sharding_db;

-- 2. Add Resources
ADD RESOURCE ds_0 (
    HOST=127.0.0.1,
    PORT=3306,
    DB=demo_ds_0,
    USER=root,
    PASSWORD=123456
), ds_1(
    HOST=127.0.0.1,
    PORT=3306,
    DB=demo_ds_1,
    USER=root,
    PASSWORD=123456
);
```

```
mysql> CREATE DATABASE sharding_db;
Query OK, 0 rows affected (0.04 sec)

mysql> USE sharding_db;
Database changed
mysql> ADD RESOURCE ds_0 (
    ->     HOST=127.0.0.1,
    ->     PORT=3306,
    ->     DB=demo_ds_0,
    ->     USER=root,
    ->     PASSWORD=123456
    -> ), ds_1(
    ->     HOST=127.0.0.1,
    ->     PORT=3306,
    ->     DB=demo_ds_1,
    ->     USER=root,
    ->     PASSWORD=123456
    -> );
Query OK, 0 rows affected (0.12 sec)
```

查看存储资源

```
mysql> show database resources\G;
*************************** 1. row ***************************
                        name: ds_1
                        type: MySQL
                        host: 127.0.0.1
                        port: 3306
                          db: demo_ds_1
connection_timeout_milliseconds: 30000
      idle_timeout_milliseconds: 60000
      max_lifetime_milliseconds: 2100000
                max_pool_size: 50
                min_pool_size: 1
                    read_only: false
              other_attributes: {"dataSourceProperties":{"maintainTimeStats":"false","rewriteBatchedStatements":"true","tinyInt1isB
it":"false","cacheResultSetMetadata":"false","useServerPrepStmts":"true","netTimeoutForStreamingResults":"0","useSSL":"false","prepS
tmtCacheSqlLimit":"2048","elideSetAutoCommits":"true","cachePrepStmts":"true","serverTimezone":"UTC","zeroDateTimeBehavior":"round",
"prepStmtCacheSize":"8192","useLocalSessionState":"true","cacheServerConfiguration":"true"},"healthCheckProperties":{},"initializati
onFailTimeout":1,"validationTimeout":5000,"leakDetectionThreshold":0,"poolName":"HikariPool-13","registerMbeans":false,"allowPoolSus
pension":false,"autoCommit":true,"isolateInternalQueries":false}
*************************** 2. row ***************************
                        name: ds_0
                        type: MySQL
                        host: 127.0.0.1
                        port: 3306
                          db: demo_ds_0
connection_timeout_milliseconds: 30000
      idle_timeout_milliseconds: 60000
      max_lifetime_milliseconds: 2100000
                max_pool_size: 50
                min_pool_size: 1
                    read_only: false
              other_attributes: {"dataSourceProperties":{"maintainTimeStats":"false","rewriteBatchedStatements":"true","tinyInt1isB
it":"false","cacheResultSetMetadata":"false","useServerPrepStmts":"true","netTimeoutForStreamingResults":"0","useSSL":"false","prepS
tmtCacheSqlLimit":"2048","elideSetAutoCommits":"true","cachePrepStmts":"true","serverTimezone":"UTC","zeroDateTimeBehavior":"round",
"prepStmtCacheSize":"8192","useLocalSessionState":"true","cacheServerConfiguration":"true"},"healthCheckProperties":{},"initializati
onFailTimeout":1,"validationTimeout":5000,"leakDetectionThreshold":0,"poolName":"HikariPool-14","registerMbeans":false,"allowPoolSus
pension":false,"autoCommit":true,"isolateInternalQueries":false}
2 rows in set (0.01 sec)
```

创建分片算法

```
-- Create Sharding Algorithms
-- 分库时按 user_id 对 2 取模
CREATE SHARDING ALGORITHM databas
TYPE(NAME=INLINE,PROPERTIES("algo
);
-- 分表时按 order_id 对 3 取模
CREATE SHARDING ALGORITHM t_order
TYPE(NAME=INLINE,PROPERTIES("algo
);
CREATE SHARDING ALGORITHM t_order
TYPE(NAME=INLINE,PROPERTIES("algo
);
```

```
mysql> CREATE SHARDING ALGORITHM database_inline (
    -> TYPE(NAME=INLINE,PROPERTIES("algorithm-expression"="ds_${user_id % 2}"))
    -> );
Query OK, 0 rows affected (0.06 sec)

mysql> CREATE SHARDING ALGORITHM t_order_inline (
    -> TYPE(NAME=INLINE,PROPERTIES("algorithm-expression"="t_order_${order_id % 3}"))
    -> );
Query OK, 0 rows affected (0.06 sec)

mysql> CREATE SHARDING ALGORITHM t_order_item_inline (
    -> TYPE(NAME=INLINE,PROPERTIES("algorithm-expression"="t_order_item_${order_id % 3}"))
    -> );
Query OK, 0 rows affected (0.06 sec)

mysql> SHOW SHARDING ALGORITHMS;
+--------------------+--------+-----------------------------------------------------+
| name               | type   | props                                               |
+--------------------+--------+-----------------------------------------------------+
| database_inline    | inline | algorithm-expression=ds_${user_id % 2}              |
| t_order_inline     | inline | algorithm-expression=t_order_${order_id % 3}        |
| t_order_item_inline | inline | algorithm-expression=t_order_item_${order_id % 3} |
+--------------------+--------+-----------------------------------------------------+
3 rows in set (0.01 sec)
```

创建默认分片策略

```
-- Create Defaul
CREATE DEFAULT S
TYPE=STANDARD,SH
);
```

```
mysql> CREATE DEFAULT SHARDING DATABASE STRATEGY (
    -> TYPE=STANDARD,SHARDING_COLUMN=user_id,SHARDING_ALGORITHM=database_inline
    -> );
Query OK, 0 rows affected (0.06 sec)

mysql> SHOW DEFAULT SHARDING STRATEGY\G;
*************************** 1. row ***************************
                  name: TABLE
                  type: NONE
       sharding_column:
 sharding_algorithm_name:
 sharding_algorithm_type:
sharding_algorithm_props:
*************************** 2. row ***************************
                  name: DATABASE
                  type: STANDARD
       sharding_column: user_id
 sharding_algorithm_name: database_inline
 sharding_algorithm_type: inline
sharding_algorithm_props: {algorithm-expression=ds_${user_id % 2}}
2 rows in set (0.01 sec)
```

创建分片规则

```
-- Create Sharding
CREATE SHARDING TA
DATANODES("ds_${0.
TABLE_STRATEGY(TYP
               SHA
);
CREATE SHARDING TA
DATANODES("ds_${0.
TABLE_STRATEGY(TYP
               SHA
);
```

```
mysql> CREATE SHARDING TABLE RULE t_order (
    -> DATANODES("ds_${0..1}.t_order_${0..2}"),
    -> TABLE_STRATEGY(TYPE=STANDARD,SHARDING_COLUMN=order_id,
    ->                SHARDING_ALGORITHM=t_order_inline)
    -> );
Query OK, 0 rows affected (0.15 sec)

mysql> CREATE SHARDING TABLE RULE t_order_item (
    -> DATANODES("ds_${0..1}.t_order_item_${0..2}"),
    -> TABLE_STRATEGY(TYPE=STANDARD,SHARDING_COLUMN=order_id,
    ->                SHARDING_ALGORITHM=t_order_item_inline)
    -> );
Query OK, 0 rows affected (0.08 sec)
```

查看分片规则

```
mysql> SHOW SHARDING TABLE RULES\G;
*************************** 1. row ***************************
                           table: t_order
              actual_data_nodes: ds_${0..1}.t_order_${0..2}
             actual_data_sources:
          database_strategy_type: STANDARD
        database_sharding_column: user_id
 database_sharding_algorithm_type: inline
database_sharding_algorithm_props: algorithm-expression=ds_${user_id % 2}
            table_strategy_type: STANDARD
          table_sharding_column: order_id
   table_sharding_algorithm_type: inline
  table_sharding_algorithm_props: algorithm-expression=t_order_${order_id % 3}
            key_generate_column:
             key_generator_type:
            key_generator_props:
*************************** 2. row ***************************
                           table: t_order_item
              actual_data_nodes: ds_${0..1}.t_order_item_${0..2}
             actual_data_sources:
          database_strategy_type: STANDARD
        database_sharding_column: user_id
 database_sharding_algorithm_type: inline
database_sharding_algorithm_props: algorithm-expression=ds_${user_id % 2}
            table_strategy_type: STANDARD
          table_sharding_column: order_id
   table_sharding_algorithm_type: inline
  table_sharding_algorithm_props: algorithm-expression=t_order_item_${order_id % 3}
            key_generate_column:
             key_generator_type:
            key_generator_props:
2 rows in set (0.01 sec)
```

一步到位

```
mysql> CREATE SHARDING TABLE RULE t_order_detail (
    -> DATANODES("ds_${0..1}.t_order_detail_${0..1}"),
    -> DATABASE_STRATEGY(TYPE=STANDARD,SHARDING_COLUMN=user_id,SHARDING_ALGORITHM(TYPE(NAME=INLINE,PROPERTIES("algorithm-expression"
="ds_${user_id % 2}")))),
    -> TABLE_STRATEGY(TYPE=STANDARD,SHARDING_COLUMN=order_id,SHARDING_ALGORITHM(TYPE(NAME=INLINE,PROPERTIES("algorithm-expression"="
t_order_detail_${order_id % 3}"))))
    -> );
Query OK, 0 rows affected (0.08 sec)

mysql>
```

```
);
```

```
mysql> SHOW SHARDING ALGORITHMS;
+-------------------------------+--------+-----------------------------------------------+
| name                          | type   | props                                         |
+-------------------------------+--------+-----------------------------------------------+
| database_inline               | inline | algorithm-expression=ds_${user_id % 2}        |
| t_order_inline                | inline | algorithm-expression=t_order_${order_id % 3}  |
| t_order_item_inline           | inline | algorithm-expression=t_order_item_${order_id % 3} |
| t_order_detail_database_inline | inline | algorithm-expression=ds_${user_id % 2}        |
| t_order_detail_table_inline   | inline | algorithm-expression=t_order_detail_${order_id % 3} |
+-------------------------------+--------+-----------------------------------------------+
5 rows in set (0.00 sec)
```

查看分片节点分布

语法：SHOW SHARDING TABLE NODES

```
mysql> SHOW SHARDING TABLE NODES t_order;
+---------+-----------------------------------------------------------------------------------------------+
| name    | nodes                                                                                         |
+---------+-----------------------------------------------------------------------------------------------+
| t_order | ds_0.t_order_0, ds_0.t_order_1, ds_0.t_order_2, ds_1.t_order_0, ds_1.t_order_1, ds_1.t_order_2 |
+---------+-----------------------------------------------------------------------------------------------+
1 row in set (0.01 sec)

mysql> SHOW SHARDING TABLE NODES t_order_item;
+-------------+-----------------------------------------------------------------------------------------------------------------------------------
----------+
| name        | nodes
          |
+-------------+-----------------------------------------------------------------------------------------------------------------------------------
----------+
| t_order_item | ds_0.t_order_item_0, ds_0.t_order_item_1, ds_0.t_order_item_2, ds_1.t_order_item_0, ds_1.t_order_item_1, ds_1.t_ord
er_item_2 |
+-------------+-----------------------------------------------------------------------------------------------------------------------------------
----------+
1 row in set (0.00 sec)

mysql> SHOW SHARDING TABLE NODES t_order_detail;
+---------------+-------------------------------------------------------------------------------------------+
| name          | nodes                                                                                     |
+---------------+-------------------------------------------------------------------------------------------+
| t_order_detail | ds_0.t_order_detail_0, ds_0.t_order_detail_1, ds_1.t_order_detail_0, ds_1.t_order_detail_1 |
+---------------+-------------------------------------------------------------------------------------------+
1 row in set (0.00 sec)
```

预览 DDL

语法：PREVIEW SQL

```
mysql> PREVIEW 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`)
    -> );
+-----------------+------------------------------------------------------------------
-----------------------------------------------------------------------------------+
| data_source_name | actual_sql
                                                                                    |
+-----------------+------------------------------------------------------------------
-----------------------------------------------------------------------------------+
| ds_1             | CREATE TABLE `t_order_0` (
  `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`)
) |
| ds_1             | CREATE TABLE `t_order_1` (
  `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`)
) |
| ds_1             | CREATE TABLE `t_order_2` (
  `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`)
) |
| ds_0             | CREATE TABLE `t_order_0` (
  `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`)
) |
| ds_0             | CREATE TABLE `t_order_1` (
  `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`)
) |
| ds_0             | CREATE TABLE `t_order_2` (
  `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`)
) |
+-----------------+------------------------------------------------------------------
```

预览 DML

语法：PREVIEW SQL

```
mysql> PREVIEW INSERT INTO t_order (order_id, user_id, status) VALUES
    -> (1,1,'OK'),
    -> (2,2,'OK');
+------------------+-----------------------------------------------------------------------+
| data_source_name | actual_sql                                                            |
+------------------+-----------------------------------------------------------------------+
| ds_1             | INSERT INTO t_order_1 (order_id, user_id, status) VALUES
(1, 1, 'OK') |
| ds_0             | INSERT INTO t_order_2 (order_id, user_id, status) VALUES
(2, 2, 'OK') |
+------------------+-----------------------------------------------------------------------+
2 rows in set (0.16 sec)

mysql> PREVIEW SELECT * FROM t_order;
+------------------+-------------------------------------------------------------------------------------------------+
| data_source_name | actual_sql                                                                                      |
+------------------+-------------------------------------------------------------------------------------------------+
| ds_0             | SELECT * FROM t_order_0 UNION ALL SELECT * FROM t_order_1 UNION ALL SELECT * FROM t_order_2 |
| ds_1             | SELECT * FROM t_order_0 UNION ALL SELECT * FROM t_order_1 UNION ALL SELECT * FROM t_order_2 |
+------------------+-------------------------------------------------------------------------------------------------+
2 rows in set (0.01 sec)
```

目录

1、更多算法支持

例：自动时间范围分片

```
mysql> CREATE SHARDING TABLE RULE t_order_range(
    -> RESOURCES(ds_0,ds_1),
    -> SHARDING_COLUMN=create_time,
    -> TYPE(NAME=AUTO_INTERVAL,PROPERTIES(
    ->   "datetime-lower"="2022-07-23 00:00:00",
    ->   "datetime-upper"="2022-07-25 00:00:00",
    ->   "sharding-seconds"="86400"))
    -> );
Query OK, 0 rows affected (0.08 sec)

mysql> SHOW SHARDING TABLE NODES t_order_range;
+---------------+-----------------------------------------------------------------------------------+
| name          | nodes                                                                             |
+---------------+-----------------------------------------------------------------------------------+
| t_order_range | ds_0.t_order_range_0, ds_1.t_order_range_1, ds_0.t_order_range_2, ds_1.t_order_range_3 |
+---------------+-----------------------------------------------------------------------------------+
1 row in set (0.01 sec)
```

还支持：复合分片算法、Hint 算法、自定义算法等

2、查看未使用的存储资源、算法

```
mysql> ADD RESOURCE ds_4 (
    ->     HOST=127.0.0.1,
    ->     PORT=3306,
    ->     DB=demo_ds_4,
    ->     USER=root,
    ->     PASSWORD=123456
    -> );
Query OK, 0 rows affected (0.30 sec)

mysql> CREATE SHARDING ALGORITHM useless (
    -> TYPE(NAME=INLINE,PROPERTIES("algorithm-expression"="ds_${user_id % 2}"))
    -> );
Query OK, 0 rows affected (0.11 sec)

mysql> CREATE SHARDING KEY GENERATOR snowflake_key_generator (
    -> TYPE(NAME=SNOWFLAKE)
    -> );
Query OK, 0 rows affected (0.11 sec)
```

2、查看未使用的存储资源、算法

语法：SHOW UNUSED xxx

```
mysql> SHOW UNUSED RESOURCES\G;
*************************** 1. row ***************************
                        name: ds_4
                        type: MySQL
                        host: 127.0.0.1
                        port: 3306
                          db: demo_ds_4
connection_timeout_milliseconds: 30000
      idle_timeout_milliseconds: 60000
      max_lifetime_milliseconds: 2100000
                max_pool_size: 50
                min_pool_size: 1
                    read_only: false
              other_attributes: {"dataSourceProperties":{"maintainTimeStats":"false","rewriteBatchedStatements":"true","tinyInt1isBit"
:"false","cacheResultSetMetadata":"false","useServerPrepStmts":"true","netTimeoutForStreamingResults":"0","useSSL":"false","prepStmtCac
heSqlLimit":"2048","elideSetAutoCommits":"true","cachePrepStmts":"true","serverTimezone":"UTC","zeroDateTimeBehavior":"round","prepStmt
CacheSize":"8192","useLocalSessionState":"true","cacheServerConfiguration":"true"},"healthCheckProperties":{},"initializationFailTimeou
t":1,"validationTimeout":5000,"leakDetectionThreshold":0,"poolName":"HikariPool-16","registerMbeans":false,"allowPoolSuspension":false,
"autoCommit":true,"isolateInternalQueries":false}
1 row in set (0.01 sec)
```

```
mysql> SHOW UNUSED SHARDING KEY GENERATORS;
+-------------------------+-----------+-------+
| name                    | type      | props |
+-------------------------+-----------+-------+
| snowflake_key_generator | snowflake |       |
+-------------------------+-----------+-------+
1 row in set (0.00 sec)

mysql> SHOW UNUSED SHARDING ALGORITHMS;
+---------+--------+---------------------------------------+
| name    | type   | props                                 |
+---------+--------+---------------------------------------+
| useless | inline | algorithm-expression=ds_${user_id % 2} |
+---------+--------+---------------------------------------+
1 row in set (0.00 sec)
```

3、查询引用了特定资源、算法的规则

语法：SHOW RULES USED xxx

```
mysql> SHOW RULES USED RESOURCE ds_0;
+----------+----------------+
| type     | name           |
+----------+----------------+
| sharding | t_order_range  |
| sharding | t_order        |
| sharding | t_order_item   |
| sharding | t_order_detail |
+----------+----------------+
4 rows in set (0.01 sec)

mysql> SHOW SHARDING TABLE RULES USED KEY GENERATOR snowflake_key_generator;
Empty set (0.01 sec)

mysql> SHOW SHARDING TABLE RULES USED ALGORITHM database_inline;
+-------+--------------+
| type  | name         |
+-------+--------------+
| table | t_order      |
| table | t_order_item |
+-------+--------------+
2 rows in set (0.01 sec)
```

4、导出 YAML 配置

语法：EXPORT DATABASE CONFIG

```
----------------------------------------------------------------------------------------------------------------
----------------+
| databaseName: sharding_db
dataSources:
  ds_4:
    password: 123456
    url: jdbc:mysql://127.0.0.1:3306/demo_ds_4
    username: root
    minPoolSize: 1
    connectionTimeoutMilliseconds: 30000
    maxLifetimeMilliseconds: 2100000
    readOnly: false
    idleTimeoutMilliseconds: 60000
    maxPoolSize: 50
  ds_1:
    password: 123456
    url: jdbc:mysql://127.0.0.1:3306/demo_ds_1
    username: root
    minPoolSize: 1
    connectionTimeoutMilliseconds: 30000
    maxLifetimeMilliseconds: 2100000
    readOnly: false
    idleTimeoutMilliseconds: 60000
    maxPoolSize: 50
  ds_0:
    password: 123456
    url: jdbc:mysql://127.0.0.1:3306/demo_ds_0
    username: root
    minPoolSize: 1
    connectionTimeoutMilliseconds: 30000
    maxLifetimeMilliseconds: 2100000
    readOnly: false
    idleTimeoutMilliseconds: 60000
    maxPoolSize: 50
rules:
- !SHARDING
  autoTables:
    t_order_range:
      actualDataNodes: ds_0.t_order_range_0,ds_1.t_order_range_1,ds_0.t_order_range_2,ds_1.t_order_range_3
      actualDataSources: ds_0,ds_1
      logicTable: t_order_range
      shardingStrategy:
        standard:
          shardingAlgorithmName: t_order_range_auto_interval
          shardingColumn: create_time
  defaultDatabaseStrategy:
    standard:
      shardingAlgorithmName: database_inline
      shardingColumn: user_id
  keyGenerators:
    snowflake_key_generator:
```

4、导出 YAML 配置

语法：EXPORT DATABASE CONFIG FILE = xxx

```
mysql> EXPORT DATABASE CONFIG FILE = "/Users/rai
+-----------------------------------------------
| result
+-----------------------------------------------
| Successfully exported to: '/Users/raigor/meetu
+-----------------------------------------------
1 row in set (0.00 sec)

mysql>
```

export.yaml

```yaml
databaseName: sharding_db
dataSources:
  ds_4:
    password: 123456
    url: jdbc:mysql://127.0.0.1:3306/demo_ds_4
    username: root
    minPoolSize: 1
    connectionTimeoutMilliseconds: 30000
    maxLifetimeMilliseconds: 2100000
    readOnly: false
    idleTimeoutMilliseconds: 60000
    maxPoolSize: 50
  ds_1:
    password: 123456
    url: jdbc:mysql://127.0.0.1:3306/demo_ds_1
    username: root
    minPoolSize: 1
    connectionTimeoutMilliseconds: 30000
    maxLifetimeMilliseconds: 2100000
    readOnly: false
    idleTimeoutMilliseconds: 60000
    maxPoolSize: 50
  ds_0:
    password: 123456
    url: jdbc:mysql://127.0.0.1:3306/demo_ds_0
    username: root
    minPoolSize: 1
    connectionTimeoutMilliseconds: 30000
    maxLifetimeMilliseconds: 2100000
    readOnly: false
    idleTimeoutMilliseconds: 60000
    maxPoolSize: 50
rules:
- !SHARDING
  autoTables:
    t_order_range:
      actualDataNodes: ds_0.t_order_range_0,ds_1.t_order_range_1,ds_0.t_order_range_2,ds_1.t_order_range_3
      actualDataSources: ds_0,ds_1
      logicTable: t_order_range
      shardingStrategy:
        standard:
          shardingAlgorithmName: t_order_range_auto_interval
          shardingColumn: create_time
  defaultDatabaseStrategy:
    standard:
      shardingAlgorithmName: database_inline
      shardingColumn: user_id
  keyGenerators:
    snowflake_key_generator:
      type: snowflake
  shardingAlgorithms:
    database_inline:
      props:
        algorithm-expression: ds_${user_id % 2}
      type: inline
    t_order_inline:
      props:
        algorithm-expression: t_order_${order_id % 3}
      type: inline
    t_order_item_inline:
      props:
        algorithm-expression: t_order_item_${order_id % 3}
      type: inline
    t_order_detail_database_inline:
```

4、CONVERT YAML 配置

语法：CONVERT YAML CONFIG

```
mysql> CONVERT YAML CONFIG FILE = "/Users/raigor/meetup/export.yaml";
+------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------+
| distsql




                                       |
+------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------+
| CREATE DATABASE sharding_db;
USE sharding_db;
ADD RESOURCE ds_4 (
URL="jdbc:mysql://127.0.0.1:3306/demo_ds_4",
USER=root,
PASSWORD="123456",
PROPERTIES("minPoolSize"="1","connectionTimeoutMilliseconds"="30000","maxLifetimeMilliseconds"="2100000","readOnly"="false","idleTimeou
tMilliseconds"="60000","maxPoolSize"="50")),
 ds_1 (
URL="jdbc:mysql://127.0.0.1:3306/demo_ds_1",
USER=root,
PASSWORD="123456",
PROPERTIES("minPoolSize"="1","connectionTimeoutMilliseconds"="30000","maxLifetimeMilliseconds"="2100000","readOnly"="false","idleTimeou
tMilliseconds"="60000","maxPoolSize"="50")),
 ds_0 (
URL="jdbc:mysql://127.0.0.1:3306/demo_ds_0",
USER=root,
PASSWORD="123456",
PROPERTIES("minPoolSize"="1","connectionTimeoutMilliseconds"="30000","maxLifetimeMilliseconds"="2100000","readOnly"="false","idleTimeou
tMilliseconds"="60000","maxPoolSize"="50")); |
+------------------------------------------------------------------------------------------------------------------------------------------
```

总结

01. 统一交互体验

02. 配置动态生效

03. 覆盖全部场景

04. 更多独占特性

欢迎关注我们！

技术干货

加入交流群

SphereEx 官网：https://sphere-ex.com

Apache ShardingSphere Website：https://shardingsphere.apache.org

Apache ShardingSphere GitHub：https://github.com/apache/shardingsphere

Apache ShardingSphere Slack Channel：https://apacheshardingsphere.slack.com

ShardingSphere

谢谢观看