ShardingSphere

基于 DistSQL 的 ShardingSphere 分布式数据库管理实践

兰城翔

兰城翔

SphereEx 研发工程师

Apache ShardingSphere Contributor

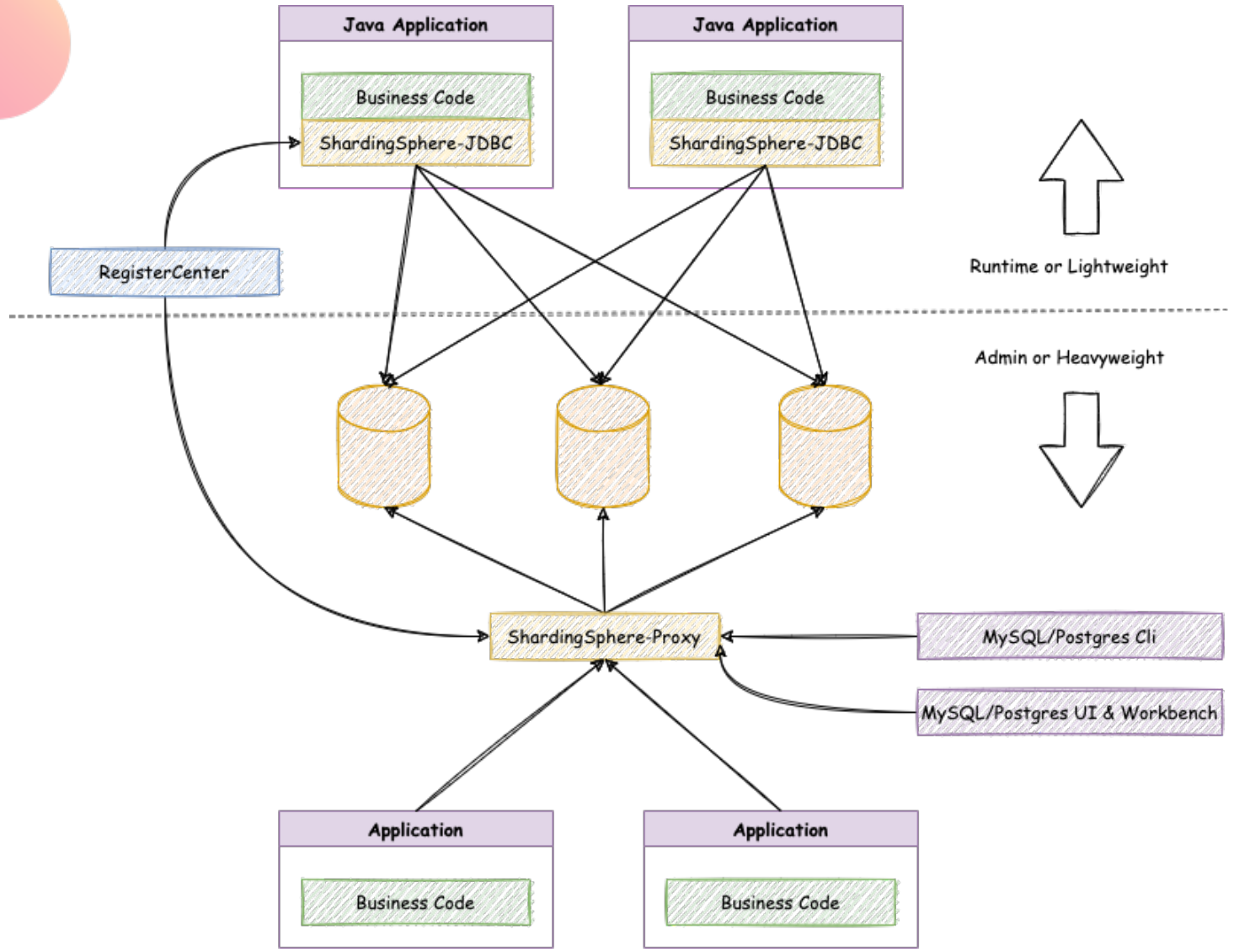
概览

01. ShardingSphere 体系下的分布式数据库场景介绍

02. 什么是 DistSQL

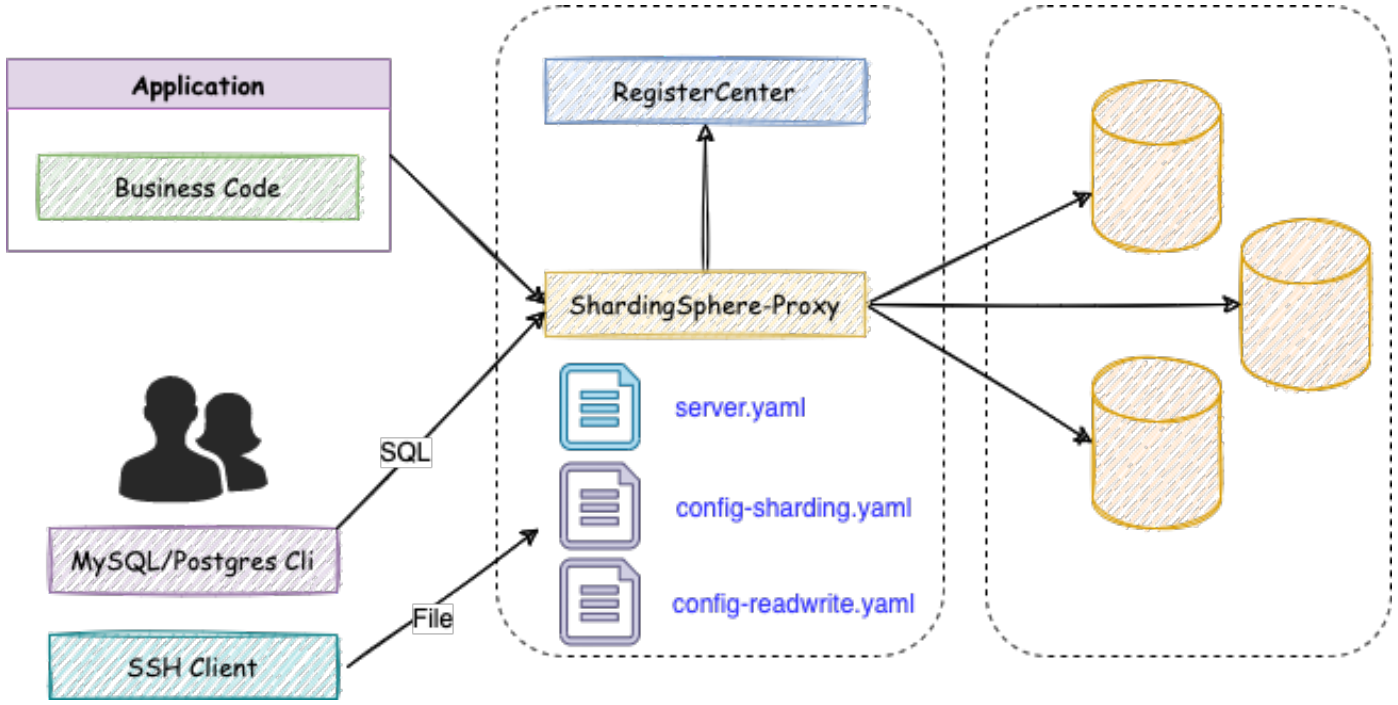
03. DistSQL 下 ShardingSphere 的多特性组合管理实践

ShardingSphere 体系下的分布式数据库场景



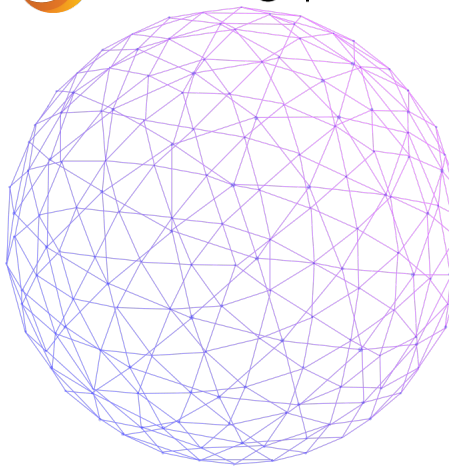
- 统一配置规则
- 动态调整配置规则

Proxy 管理流程



ShardingSphere-Proxy: v4.x

- 通过配置文件管理数据库资源
- 通过配置文件管理权限和规则
- 一个文件对应一个逻辑 schema
- 变更规则需要服务器写权限
- 变更规则需要重启 Proxy



DistSQL 简介

DistSQL（Distributed SQL）也称为分布式 SQL ，其发布于 Apache ShardingSphere 5.0.0-beta 版本， 是 ShardingSphere 特有的一种内置 SQL 语言，目前作用于 ShardingSphere-Proxy ，它在与标准 SQL 相似的基础上为 Proxy 提供标准 SQL 之外的功能。

DistSQL 设计的目的在于突破 ShardingSphere 作为中间件的局限，让开发者像使用数据库一样使用 ShardingSphere ，同时也为 ShardingSphere 从面向开发者的框架和中间件过渡到面向运维人员的基础设施提供支撑。

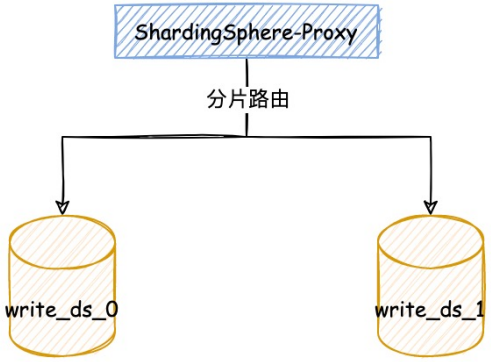
DistSQL 分类

资源与规则定义语言（RDL： Resource & Rule Definition Language）：负责资源和规则的创建、修改和删除，其语句包括动词 CREATE，ALTER 和 DROP。

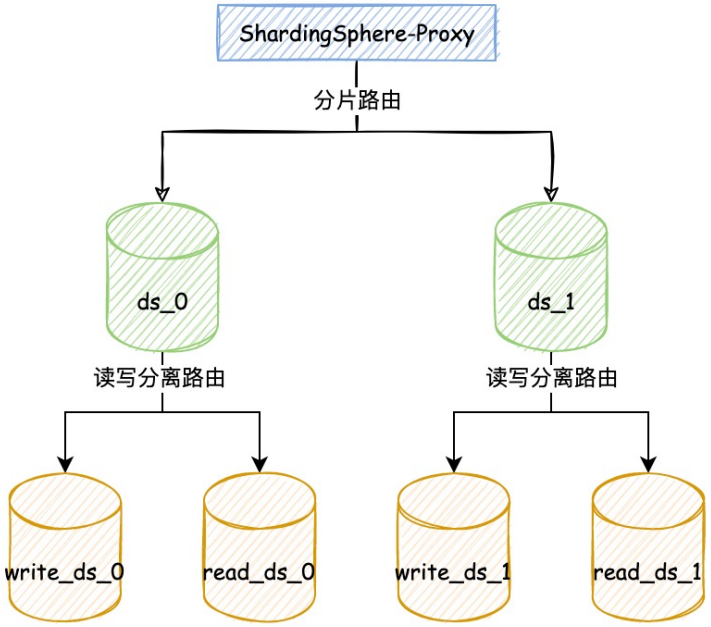
资源与规则查询语言（RQL： Resource & Rule Query Language）：负责资源和规则的查询和展现，其语句主要动词为 SHOW

资源与规则管理语言（RAL： Resource & Rule Administration Language）：负责事务类型切换、熔断控制、元数据管理、SQL 预览等增量功能的操作

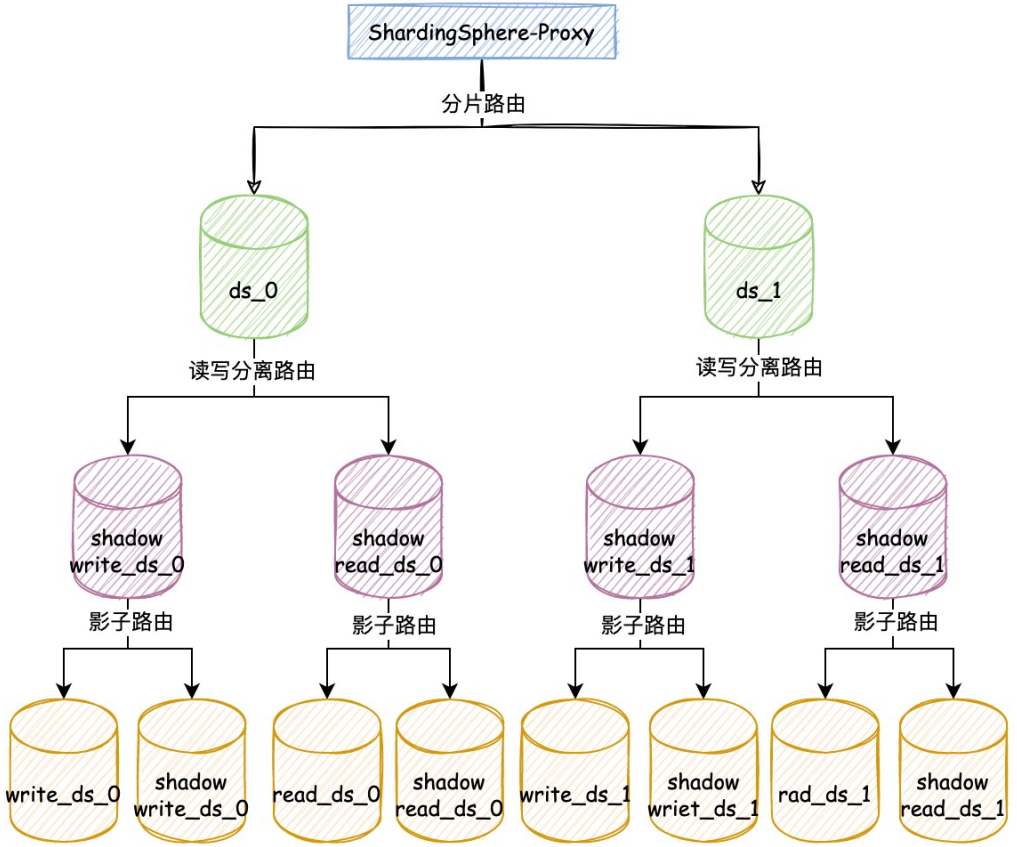
实践过程预览



t_order : 两库两表
t_order_item : 两库两表

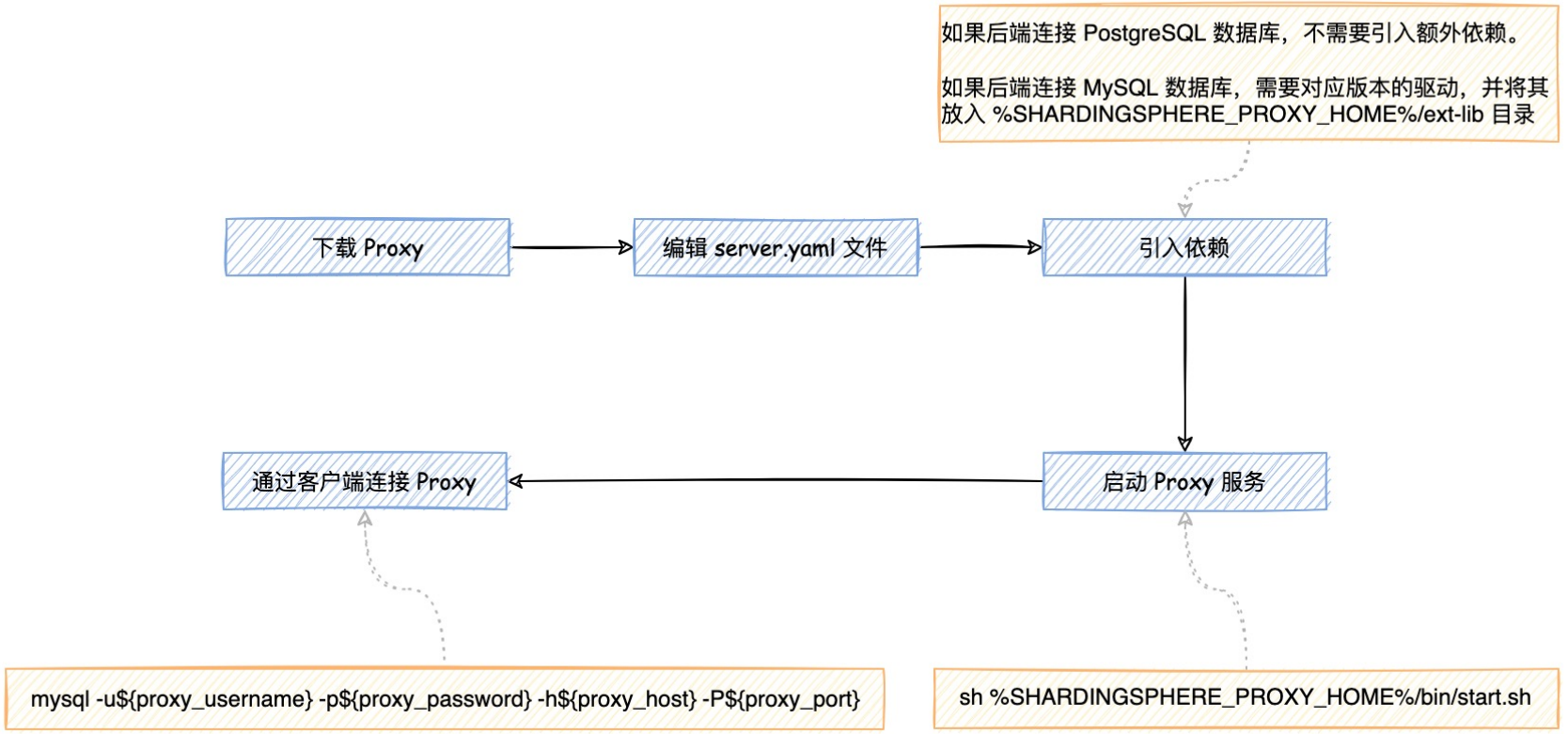


增加 read_ds_0、read_ds_1



增加影子库规则

多特性管理实践-启动


如果后端连接 PostgreSQL 数据库，不需要引入额外依赖。
如果后端连接 MySQL 数据库，需要对应版本的驱动，并将其放入 %SHARDINGSPHERE_PROXY_HOME%/ext-lib 目录
下载 Proxy
编辑 server.yaml 文件
引入依赖
通过客户端连接 Proxy
启动 Proxy 服务
mysql -u${proxy_username} -p${proxy_password} -h${proxy_host} -P${proxy_port}
sh %SHARDINGSPHERE_PROXY_HOME%/bin/start.sh

多特性组合实践-分片

Create Logic Database

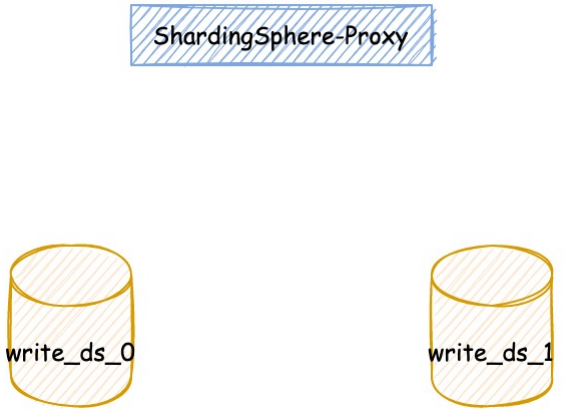
1. Create new schema and use

```
CREATE SCHEMA mixture_db; USE mixture_db;
```

Create Sharding Rules

1. Add database resource for sharding rule

```
ADD RESOURCE write_ds_0 (
    HOST=127.0.0.1,
    PORT=3306,
    DB=demo_write_ds_0,
    USER=root,
    PASSWORD=root
), write_ds_1 (
    HOST=127.0.0.1,
    PORT=3306,
    DB=demo_write_ds_1,
    USER=root,
    PASSWORD=root
);
```



2. Show schema resources

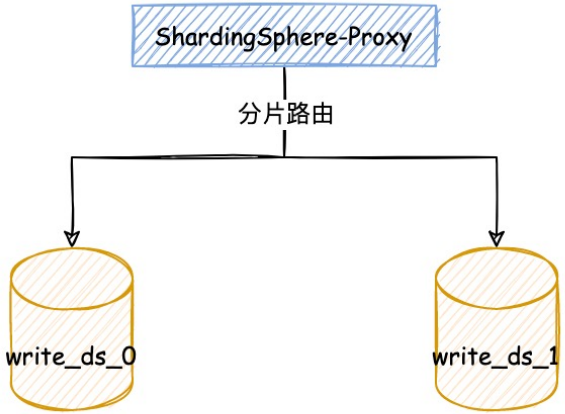
```
SHOW SCHEMA RESOURCES;
```

```
+------------------+-------+-----------+------+-----------------------+------------+
| name             | type  | host      | port | db                    | attribute  |
+------------------+-------+-----------+------+-----------------------+------------+
| write_ds_0       | MySQL | 127.0.0.1 | 3306 | demo_write_ds_0       | ...        |
| write_ds_1       | MySQL | 127.0.0.1 | 3306 | demo_write_ds_1       | ...        |
+------------------+-------+-----------+------+-----------------------+------------+
2 rows in set (0.01 sec)
```

多特性组合实践-分片

3. Create sharding table rules

```
-- 分片算法
CREATE SHARDING ALGORITHM t_order_inline (
    TYPE(NAME=inline,PROPERTIES("algorithm-expression"="t_order_${order_id % 2}"))
), t_order_item_inline (
    TYPE(NAME=inline,PROPERTIES("algorithm-expression"="t_order_item_${order_id % 2}"))
), database_inline (
    TYPE(NAME=inline,PROPERTIES("algorithm-expression"="write_ds_${user_id % 2}"))
);
-- 主键生成器
CREATE SHARDING KEY GENERATOR snowflake (
    TYPE(NAME=SNOWFLAKE, PROPERTIES("worker-id"=123))
);
-- 默认策略
CREATE DEFAULT SHARDING DATABASE STRATEGY (
    TYPE = standard,SHARDING_COLUMN=user_id,SHARDING_ALGORITHM=database_inline
);
-- 分片规则
CREATE SHARDING TABLE RULE t_order (
  DATANODES("write_ds_${0..1}.t_order_${0..1}"),
  TABLE_STRATEGY(TYPE=standard, SHARDING_COLUMN=order_id, SHARDING_ALGORITHM=t_order_inline),
  GENERATED_KEY(COLUMN=order_id, GENERATED_KEY_ALGORITHM=snowflake)
), t_order_item (
  DATANODES("write_ds_${0..1}.t_order_item_${0..1}"),
  TABLE_STRATEGY(TYPE=standard, SHARDING_COLUMN=order_id, SHARDING_ALGORITHM=t_order_item_inline),
  GENERATED_KEY(COLUMN=order_id, GENERATED_KEY_ALGORITHM=snowflake)
);
```



多特性组合实践-分片

4. Show sharding rules

```
SHOW SHARDING ALGORITHMS; SHOW SHARDING KEY GENERATORS; SHOW DEFAULT SHARDING STRATEGY; SHOW SHARDING TABLE RULES;
```

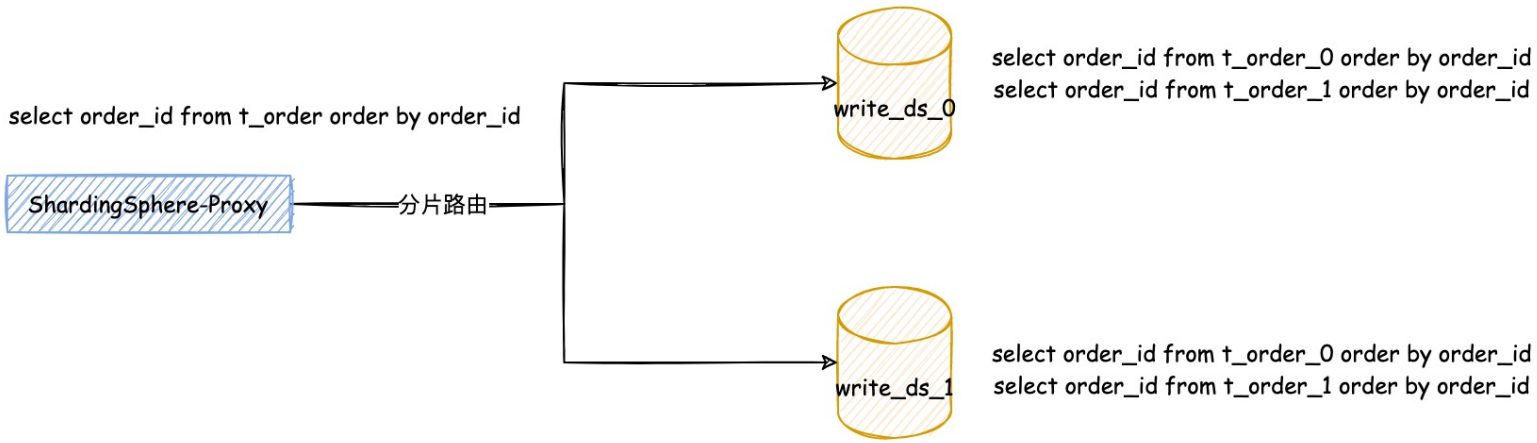
```
-- ALGORITHMS
+--------------------+--------+-------------------------------------------------+
| name               | type   | props                                           |
+--------------------+--------+-------------------------------------------------+
| database_inline    | inline | algorithm-expression=write_ds_${user_id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

-- KEY GENERATORS
+-----------+-----------+-----------------+
| name      | type      | props           |
+-----------+-----------+-----------------+
| snowflake | SNOWFLAKE | {worker-id=123} |
+-----------+-----------+-----------------+
1 row in set (0.00 sec)

-- DEFAULT SHARDING STRATEGY
+----------+----------+-----------------+-----------------+----------------+-----------------------------------------------+
| name     | type     | sharding_column | algorithm_name  | algorithm_type | algorithm_props                               |
+----------+----------+-----------------+-----------------+----------------+-----------------------------------------------+
| TABLE    | NONE     |                 |                 |                |                                               |
| DATABASE | STANDARD | user_id         | database_inline | inline         | {algorithm-expression=write_ds_${user_id % 2}} |
+----------+----------+-----------------+-----------------+----------------+-----------------------------------------------+

-- RULES
+--------------+-----------------------------------------+-------------------------+----------------------+------+
| table        | actual_data_nodes                       | database_sharding_column | table_sharding_column | ...  |
+--------------+-----------------------------------------+-------------------------+----------------------+-------
| t_order      | write_ds_${0..1}.t_order_${0..1}        | user_id                 | order_id             | ...  |
| t_order_item | write_ds_${0..1}.t_order_item_${0..1}   | user_id                 | order_id             | ...  |
+--------------+-----------------------------------------+-------------------------+----------------------+------+
2 rows in set (0.03 sec)
```

多特性组合实践-分片

5. Preview SQL

```
preview select order_id from t_order order by order_id;
```

```
+------------------+-----------------------------------------------------+
| data_source_name | sql                                                 |
+------------------+-----------------------------------------------------+
| write_ds_0       | select order_id from t_order_0 order by order_id |
| write_ds_0       | select order_id from t_order_1 order by order_id |
| write_ds_1       | select order_id from t_order_0 order by order_id |
| write_ds_1       | select order_id from t_order_1 order by order_id |
+------------------+-----------------------------------------------------+
```

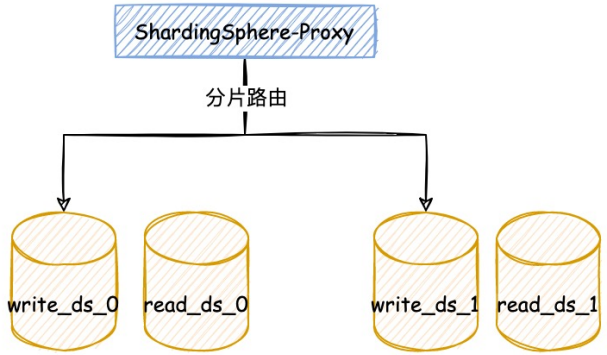


多特性组合实践-读写分离

Create Readwrite-splitting Rules

1. Add database resource for readwrite-splitting rule

```
ADD RESOURCE read_ds_0 (
    HOST=127.0.0.1,
    PORT=3306,
    DB=demo_read_ds_0,
    USER=root,
    PASSWORD=root
),  read_ds_1 (
    HOST=127.0.0.1,
    PORT=3306,
    DB=demo_read_ds_1,
    USER=root,
    PASSWORD=root
);
```

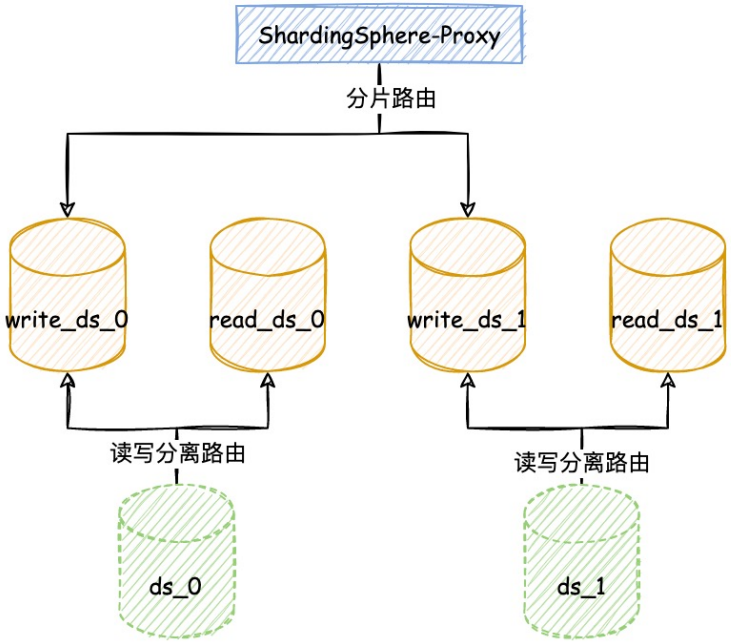


2. Show schema resources

```
SHOW SCHEMA RESOURCES;
```

3. Create readwrite-splitting rule

```
CREATE READWRITE_SPLITTING RULE ds_0 (
  WRITE_RESOURCE=write_ds_0,
  READ_RESOURCES(read_ds_0),
  TYPE(NAME=ROUND_ROBIN)
),ds_1 (
  WRITE_RESOURCE=write_ds_1,
  READ_RESOURCES(read_ds_1),
  TYPE(NAME=ROUND_ROBIN)
);
```



多特性组合实践-读写分离

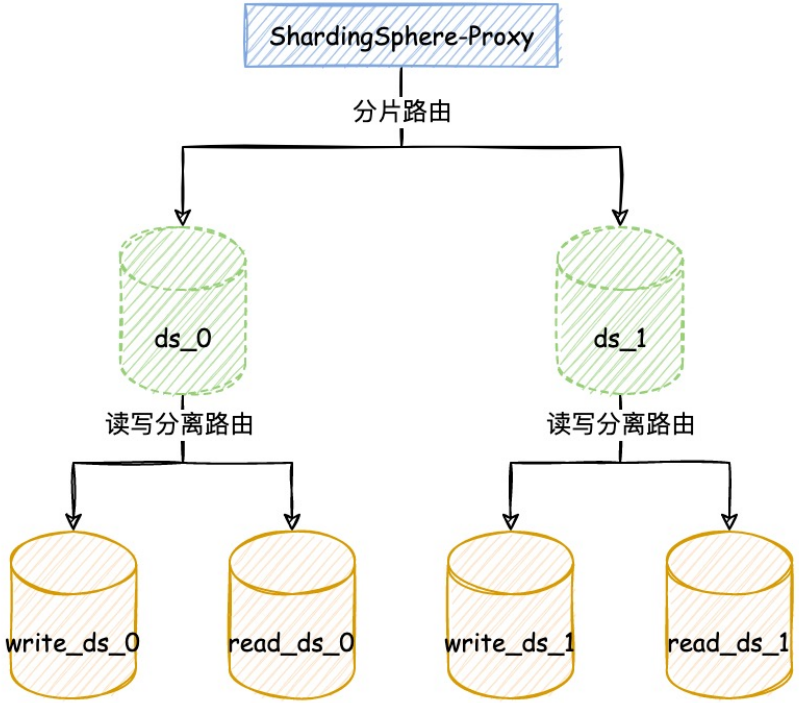
3. Update sharding database resource;

```
ALTER SHARDING TABLE RULE t_order (
  DATANODES("ds_${0..1}.t_order_${0..1}"),
  TABLE_STRATEGY(TYPE=standard, SHARDING_COLUMN=order_id, SHARDING_ALGORITHM=t_order_inline),
  GENERATED_KEY(COLUMN=order_id, GENERATED_KEY_ALGORITHM=snowflake)
), t_order_item (
  DATANODES("ds_${0..1}.t_order_item_${0..1}"),
  TABLE_STRATEGY(TYPE=standard, SHARDING_COLUMN=order_id, SHARDING_ALGORITHM=t_order_item_inline),
  GENERATED_KEY(COLUMN=order_id, GENERATED_KEY_ALGORITHM=snowflake)
);

ALTER SHARDING ALGORITHM database_inline (
    TYPE(NAME=inline,PROPERTIES("algorithm-expression"="ds_${user_id % 2}"))
)
```

4. Show sharding table rules

```
SHOW SHARDING ALGORITHMS; SHOW SHARDING TABLE RULES;
```

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

-- RULES
+--------------+---------------------------------+--------------------------+-----------------------+------+
| table        | actual_data_nodes               | database_sharding_column | table_sharding_column | ...  |
+--------------+---------------------------------+--------------------------+-----------------------+------+
| t_order      | ds_${0..1}.t_order_${0..1}      | user_id                  | order_id              | ...  |
| t_order_item | ds_${0..1}.t_order_item_${0..1} | user_id                  | order_id              | ...  |
+--------------+---------------------------------+--------------------------+-----------------------+------+
2 rows in set (0.03 sec)
```

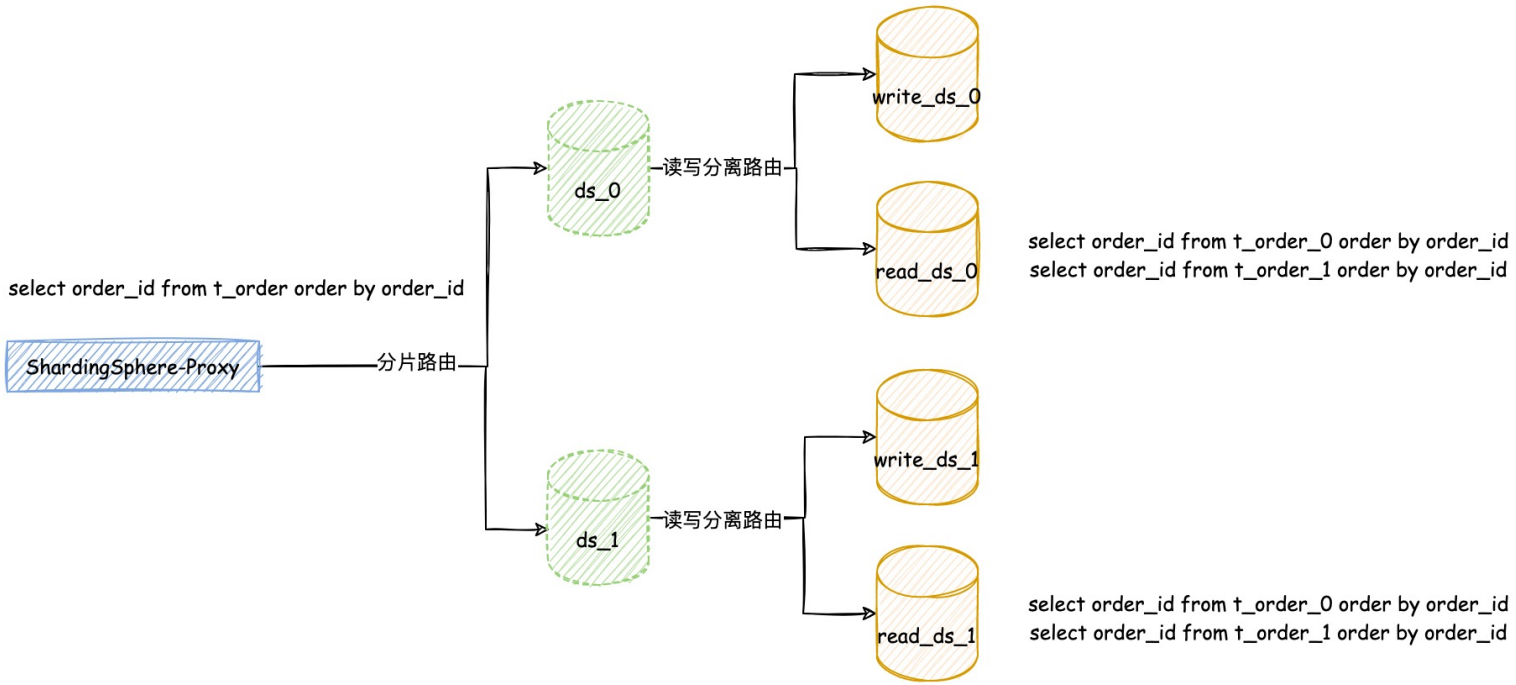


多特性组合实践-读写分离

5. Previw SQL

```
PREVIEW select order_id from t_order order by order_id;
```

```
+------------------+-----------------------------------------------------+
| data_source_name | sql                                                 |
+------------------+-----------------------------------------------------+
| read_ds_0        | select order_id from t_order_0 order by order_id |
| read_ds_0        | select order_id from t_order_1 order by order_id |
| read_ds_1        | select order_id from t_order_0 order by order_id |
| read_ds_1        | select order_id from t_order_1 order by order_id |
+------------------+-----------------------------------------------------+
```

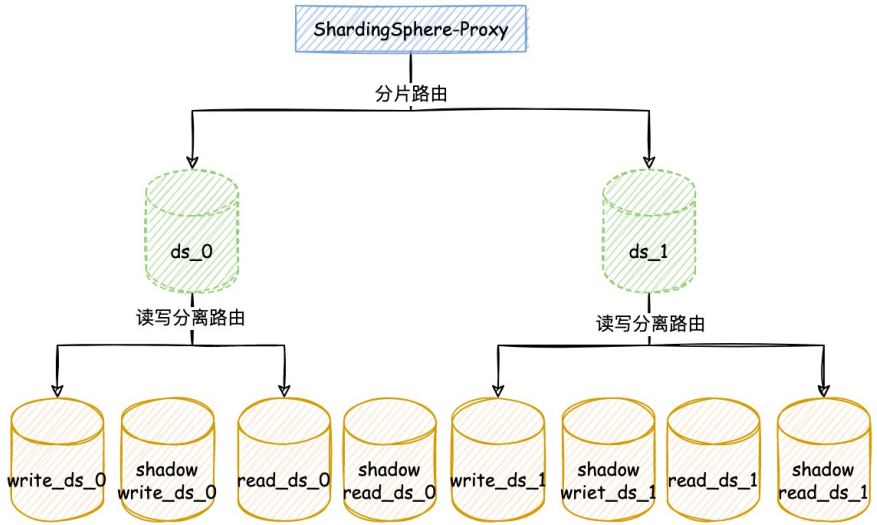


多特性组合实践-影子库

Create Shadow rules

1. Add database resource for shadow rule

```
ADD RESOURCE shadow_write_ds_0 (
    HOST=127.0.0.1,
    PORT=3306,
    DB=shadow_demo_write_ds_0,
    USER=root,
    PASSWORD=root
), shadow_read_ds_0 (
    HOST=127.0.0.1,
    PORT=3306,
    DB=shadow_demo_read_ds_0,
    USER=root,
    PASSWORD=root
), shadow_write_ds_1 (
    HOST=127.0.0.1,
    PORT=3306,
    DB=shadow_demo_write_ds_1,
    USER=root,
    PASSWORD=root
), shadow_read_ds_1 (
    HOST=127.0.0.1,
    PORT=3306,
    DB=shadow_demo_read_ds_1,
    USER=root,
    PASSWORD=root
);
```



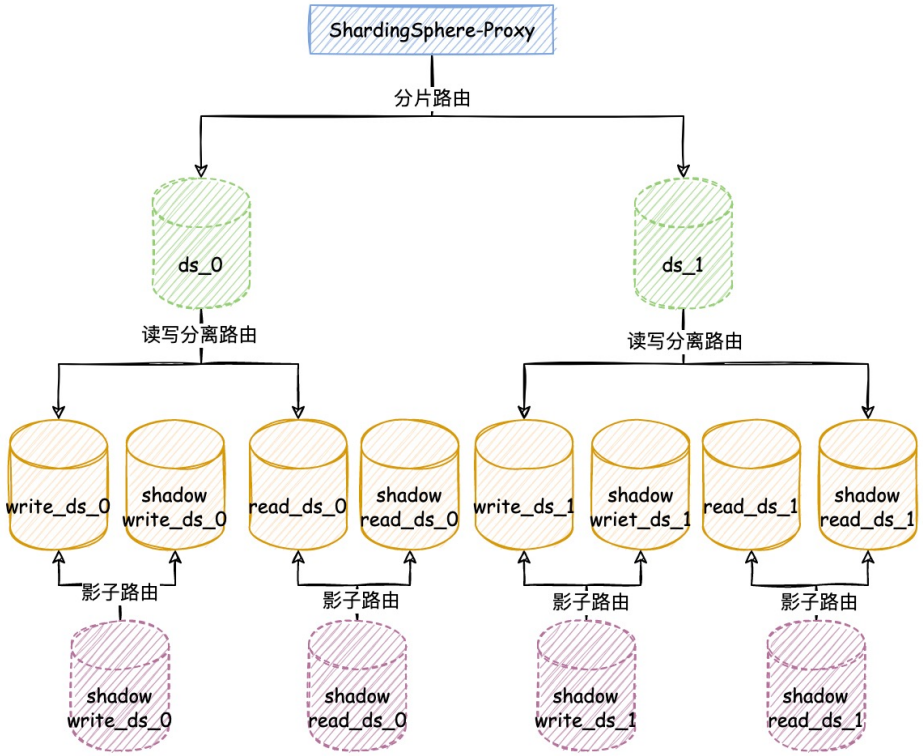
2. Show schema resources

```
SHOW SCHEMA RESOURCES;
```

多特性管理实践

3. Create Shadow Rule

```
CREATE SHADOW RULE shadow_write_ds_0(
  SOURCE=write_ds_0,
  SHADOW=shadow_write_ds_0,
  t_order((TYPE(NAME=SIMPLE_HINT, PROPERTIES(shadow="true")))),
  t_order_item((TYPE(NAME=SIMPLE_HINT, PROPERTIES(shadow="true"))))
), shadow_read_ds_0(
  SOURCE=read_ds_0,
  SHADOW=shadow_read_ds_0,
  t_order((TYPE(NAME=SIMPLE_HINT, PROPERTIES(shadow="true")))),
  t_order_item((TYPE(NAME=SIMPLE_HINT, PROPERTIES(shadow="true"))))
), shadow_write_ds_1(
  SOURCE=write_ds_1,
  SHADOW=shadow_write_ds_1,
  t_order((TYPE(NAME=SIMPLE_HINT, PROPERTIES(shadow="true")))),
  t_order_item((TYPE(NAME=SIMPLE_HINT, PROPERTIES(shadow="true"))))
), shadow_read_ds_1(
  SOURCE=read_ds_1,
  SHADOW=shadow_read_ds_1,
  t_order((TYPE(NAME=SIMPLE_HINT, PROPERTIES(shadow="true")))),
  t_order_item((TYPE(NAME=SIMPLE_HINT, PROPERTIES(shadow="true"))))
);
```



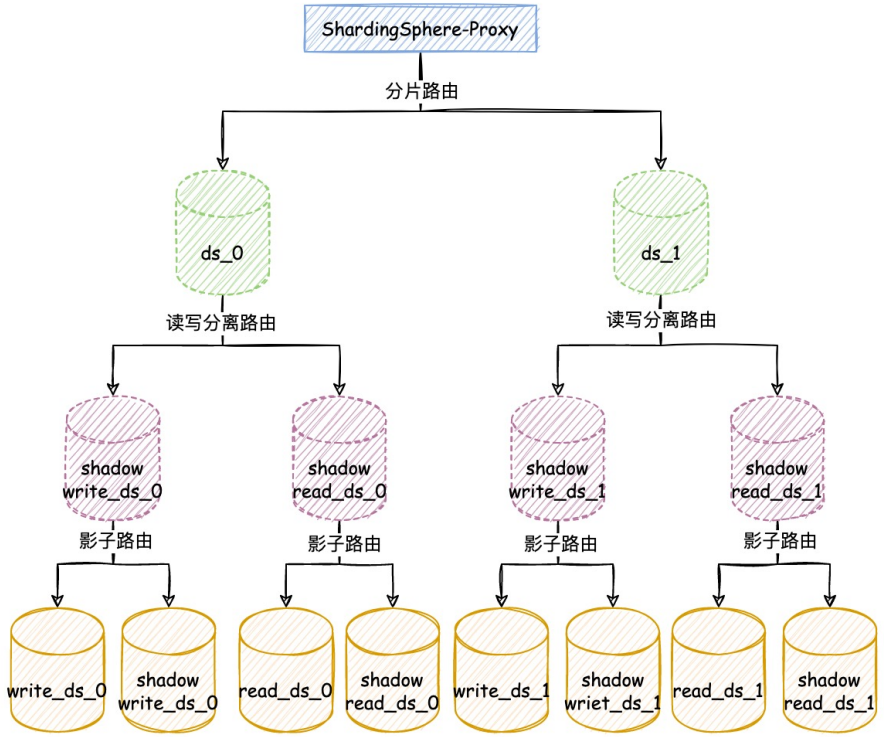
多特性管理实践

4. Update readwrite-splitting database resource

```
ALTER READWRITE_SPLITTING RULE ds_0 (
  WRITE_RESOURCE=shadow_write_ds_0,
  READ_RESOURCES(shadow_read_ds_0),
), ds_1 (
  WRITE_RESOURCE=shadow_write_ds_1,
  READ_RESOURCES(shadow_read_ds_1),
);
```

5. Show readwrite-splitting rules

```
+------+-------------------------+-------------------------+--------------------+-----+
| name | write_data_source_name | read_data_source_names | load_balancer_type | ... |
+------+-------------------------+-------------------------+--------------------+-----+
| ds_0 | shadow_write_ds_0       | shadow_read_ds_0        | ROUND_ROBIN        | ... |
| ds_1 | shadow_write_ds_1       | shadow_write_ds_0       | ROUND_ROBIN        | ... |
+------+-------------------------+-------------------------+--------------------+-----+
2 rows in set (0.02 sec)
```



多特性管理实践

6. Preview SQL

```
PREVIEW select order_id from t_order order by order_id;
```

```
+------------------+-----------------------------------------------+
| data_source_name | sql                                           |
+------------------+-----------------------------------------------+
| read_ds_0        | select order_id from t_order_0 order by order_id |
| read_ds_0        | select order_id from t_order_1 order by order_id |
| read_ds_1        | select order_id from t_order_0 order by order_id |
| read_ds_1        | select order_id from t_order_1 order by order_id |
+------------------+-----------------------------------------------+
```

```
PREVIEW select order_id from t_order order by order_id /*shadow:true*/;
```

```
+------------------+-------------------------------------------------------------------+
| data_source_name | sql                                                               |
+------------------+-------------------------------------------------------------------+
| shadow_read_ds_0 | select order_id from t_order_0 order by order_id /*shadow:true*/ |
| shadow_read_ds_0 | select order_id from t_order_1 order by order_id /*shadow:true*/ |
| shadow_read_ds_1 | select order_id from t_order_0 order by order_id /*shadow:true*/ |
| shadow_read_ds_1 | select order_id from t_order_1 order by order_id /*shadow:true*/ |
+------------------+-------------------------------------------------------------------+
```

管理进阶

- SHOW SHARDING TABLE NODES
- SHOW READWRITE_SPLITTING READ RESOURCES
- PREVIEW SQL
- SHOW ALL VARIABLES
- SHOW INSTANCE LIST
- 更多

管理进阶

展示分片表分布

```
SHOW SHARDING TABLE NODES
```

```
+--------------+-----------------------------------------------------------------------------------------+
| name         | nodes                                                                                   |
+--------------+-----------------------------------------------------------------------------------------+
| t_order      | ds_0.t_order_0, ds_0.t_order_1, ds_1.t_order_0, ds_1.t_order_1                          |
| t_order_item | ds_0.t_order_item_0, ds_0.t_order_item_1, ds_1.t_order_item_0, ds_1.t_order_item_1      |
+--------------+-----------------------------------------------------------------------------------------+
2 rows in set (0.70 sec)
```

展示读写分离读库状态

```
SHOW READWRITE_SPLITTING READ RESOURCES;
```

```
DISABLE READWRITE_SPLITTING READ shadow_read_ds_0;
```

```
+------------------+--------+
| resource         | status |
+------------------+--------+
| shadow_read_ds_0 | enable |
| shadow_read_ds_1 | enable |
+------------------+--------+
2 rows in set (0.06 sec)
```

管理进阶

展示所有配置参数

```
SHOW ALL VARIABLES;
```

```
+---------------------------------------+----------------+
| variable_name                         | variable_value |
+---------------------------------------+----------------+
| sql_show                              | true           |
| sql_simple                            | false          |
| kernel_executor_size                  | 0              |
| max_connections_size_per_query        | 1              |
| check_table_metadata_enabled          | false          |
| proxy_frontend_database_protocol_type |                |
| proxy_frontend_flush_threshold        | 128            |
| proxy_opentracing_enabled             | false          |
| proxy_hint_enabled                    | true           |
| show_process_list_enabled             | false          |
| lock_wait_timeout_milliseconds        | 50000          |
| proxy_backend_query_fetch_size        | -1             |
| check_duplicate_table_enabled         | false          |
| proxy_frontend_executor_size          | 0              |
| proxy_backend_executor_suitable       | OLAP           |
| proxy_frontend_max_connections        | 0              |
| sql_federation_enabled                | false          |
| agent_plugins_enabled                 | false          |
| cached_connections                    | 0              |
| transaction_type                      | LOCAL          |
+---------------------------------------+----------------+
20 rows in set (0.02 sec)
```

修改配置参数

```
SET VARIABLE sql_show = false;
SET VARIABLE transaction_type = XA;
SET VARIABLE proxy_hint_enabled = false;
```

后期规划

后期规划

- DistSQL 权限管理
- 元数据展示
- JDBC 支持
- 注册中心配置导出 YAML
- 开发者工具
- 更多特性