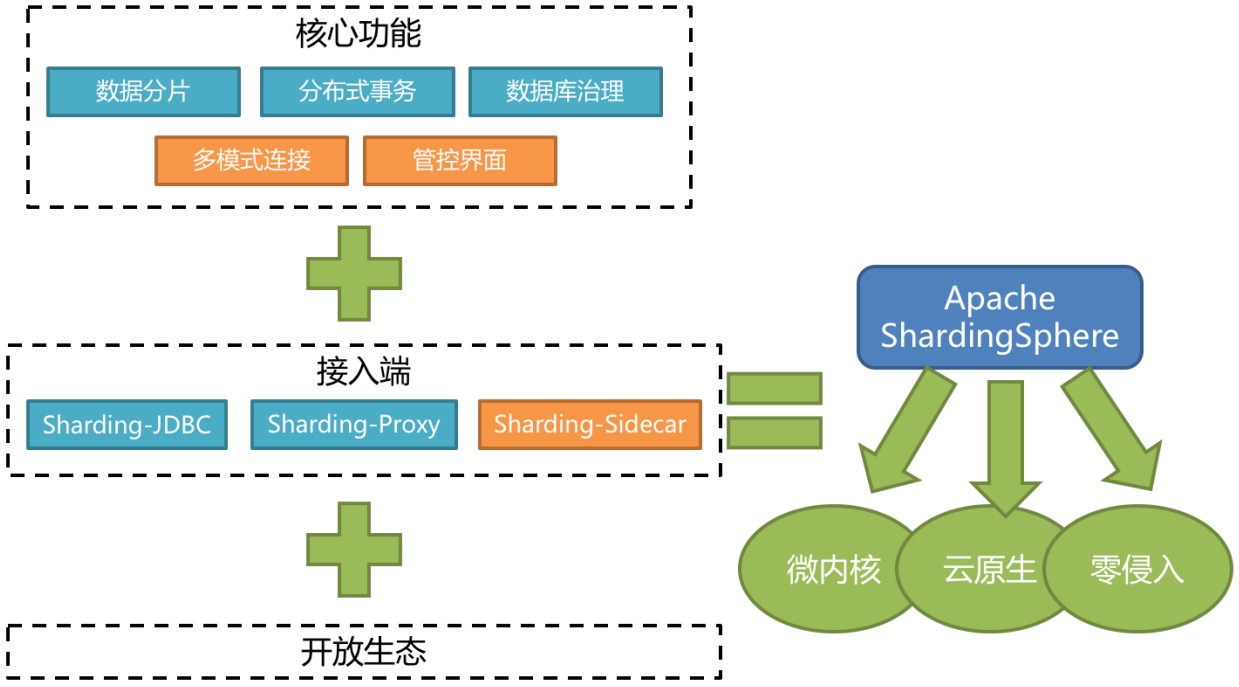
课程内容

1. **shardingsphere的基本介绍**
2. **shardingsphere的核心组成**
3. **shardingsphere的核心概念**
4. **shardingsphere+springboot+mybatis快速实战**

定位

ShardingSphere定位为关系型数据库中间件



功能列表：

	数据分片	分布式事务	数据库治理
功能列表	分库 & 分表	标准化事务接口	配置动态化
	读写分离	XA强一致事务	编排 & 治理
	分片策略定制化	柔性事务	数据脱敏
	无中心化分布式主键		可视化链路追踪

核心三套件

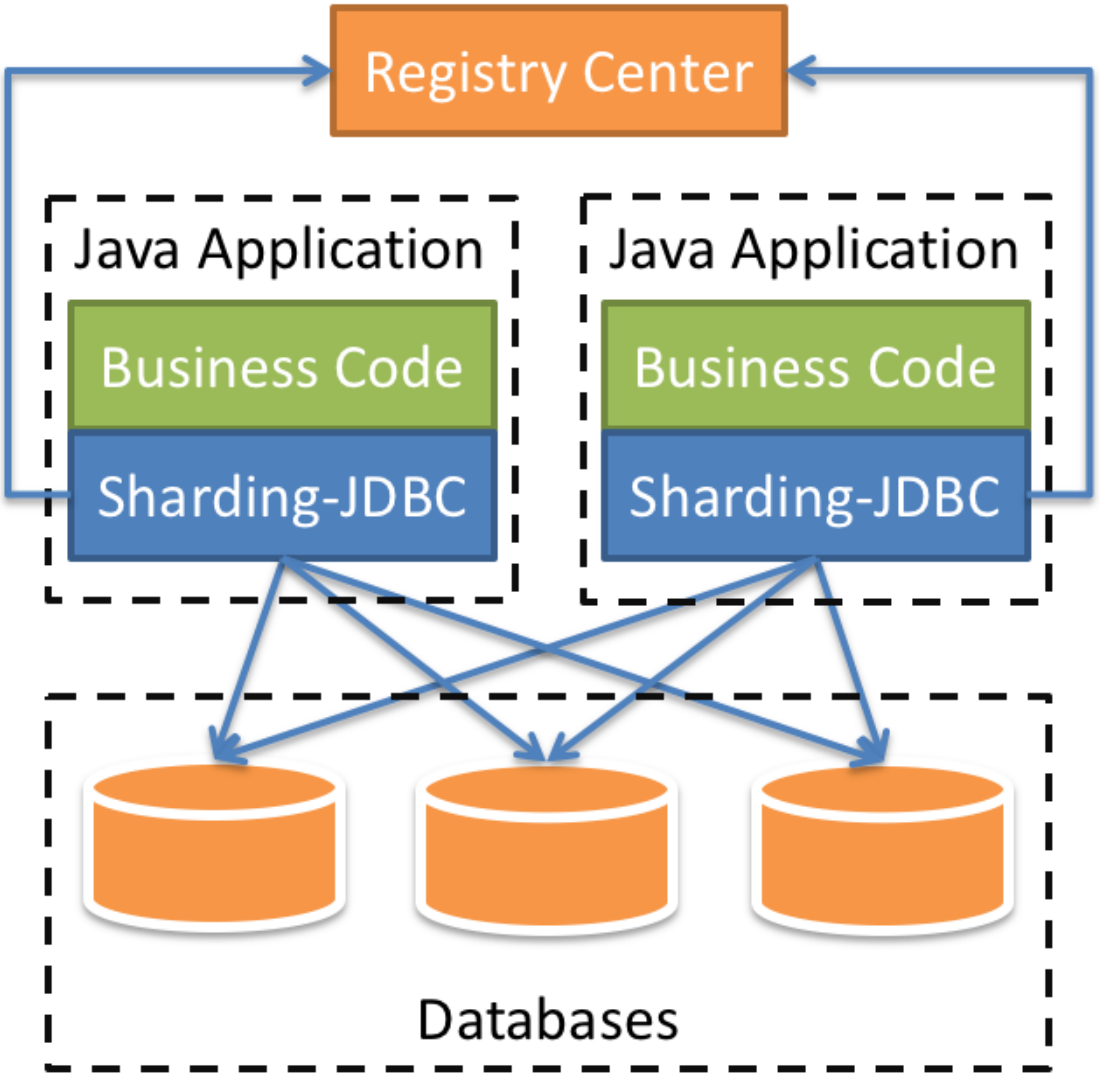
	Sharding-JDBC	*Sharding-Proxy*	*Sharding-Sidecar*
数据库	任意	MySQL	MySQL

连接消耗数	高	低	高
异构语言	仅Java	任意	任意
性能	损耗低	损耗略高	损耗低
无中心化	是	否	是
静态入口	无	有	无

Sharding-JDBC

客户端直连数据库，以jar包形式提供服务，无需额外部署和依赖，可理解为增强版的JDBC驱动，完全兼容JDBC和各种ORM框架。

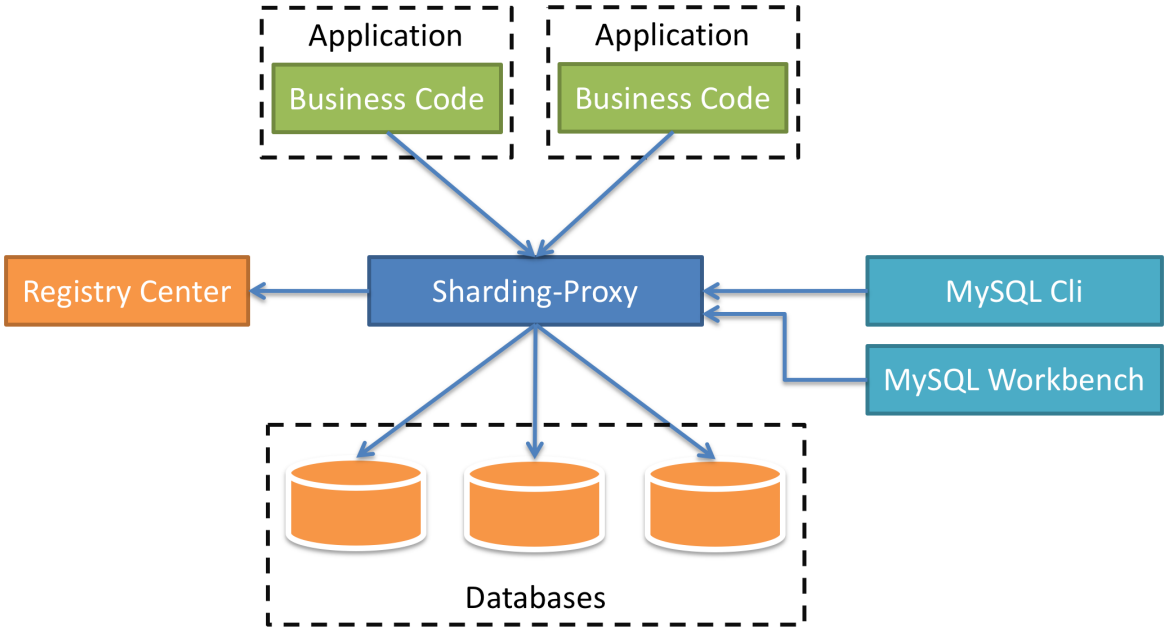
- 适用于任何基于JDBC的ORM框架：JPA, Hibernate, Mybatis, Spring JDBC Template或直接使用JDBC。
- 支持任何第三方的数据库连接池：DBCP, C3P0, BoneCP, Druid, HikariCP等。
- 支持任意实现JDBC规范的数据库。支持MySQL，Oracle，SQLServer，PostgreSQL等遵循SQL92标准的数据库。



Sharding-Proxy

透明化的数据库代理端，兼容所有MySQL/PostgreSQL协议的访问客户端。

- 向应用程序完全透明，可直接当做MySQL/PostgreSQL使用。
- 适用于任何兼容MySQL/PostgreSQL协议的的客户端。



核心概念

逻辑表

水平拆分的数据库（表）的相同逻辑和数据结构表的总称。例：订单数据根据主键尾数拆分为10张表，分别是t_order_0到t_order_9，他们的逻辑表名为t_order。

真实表

在分片的数据库中真实存在的物理表。即上个示例中的t_order_0到t_order_9。

数据节点

数据分片的最小单元。由数据源名称和数据表组成，例：ds_0.t_order_0。

绑定表

分片规则一致的主表和子表。例如：t_order表和t_order_item表，均按照order_id分片，则此两张表互为绑定表关系。绑定表之间的多表关联查询不会出现笛卡尔积关联，关联查询效率将大大提升。

广播表

指所有的分片数据源中都存在的表，表结构和表中的数据在每个数据库中均完全一致。适用于数据量不大且需要与海量数据的表进行关联查询的场景。字典表就是典型的场景。

ShardingSphere快速启动

shardingsphere+springboot+mybatis集成使用

1、引入项目依赖

```
<dependencies>
<!-- springboot-->
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-web</artifactId>
<version>2.0.5.RELEASE</version>
</dependency>
<!-- mybatis -->
<dependency>
<groupId>org.mybatis.spring.boot</groupId>
<artifactId>mybatis-spring-boot-starter</artifactId>
<version>2.0.1</version>
<exclusions>
<exclusion>
<artifactId>spring-boot-starter</artifactId>
<groupId>org.springframework.boot</groupId>
</exclusion>
</exclusions>
</dependency>
<!-- shardingsphere-jdbc,这里使用的版本为apache孵化版本，4.0之前
都是没有捐献给apache基金会的版本,之前的版本都在-->
<dependency>
<groupId>org.apache.shardingsphere</groupId>
<artifactId>sharding-jdbc-spring-boot-starter</artifactId>
<version>4.0.0-RC2</version>
</dependency>
<!-- mysql 驱动 -->
<dependency>
<groupId>mysql</groupId>
<artifactId>mysql-connector-java</artifactId>
<version>5.1.48</version>
</dependency>
<!-- 可选，工具类 -->
<dependency>
<groupId>org.projectlombok</groupId>
<artifactId>lombok</artifactId>
<version>1.16.20</version>
<scope>provided</scope>
```

```
</dependency>
</dependencies>
```

分库不分表配置

```
# 配置ds0 和ds1两个数据源,这里有个坑(使用下划线可能会有异常产生，字符不支持,如：ds_0)
spring.shardingsphere.datasource.names=ds0,ds1

#ds0 配置
spring.shardingsphere.datasource.ds0.type=com.zaxxer.hikari.HikariDataSource
#数据库驱动
spring.shardingsphere.datasource.ds0.driver-class-name=com.mysql.jdbc.Driver
spring.shardingsphere.datasource.ds0.jdbc-url=jdbc:mysql://192.168.241.198:3306/shop_ds_0?serverTimezone=UTC&useSSL=false&useUnicode=true&characterEncoding=UTF-8
spring.shardingsphere.datasource.ds0.username=root
spring.shardingsphere.datasource.ds0.password=root
#ds1 配置
spring.shardingsphere.datasource.ds1.type=com.zaxxer.hikari.HikariDataSource
#数据库驱动
spring.shardingsphere.datasource.ds1.driver-class-name=com.mysql.jdbc.Driver
spring.shardingsphere.datasource.ds1.jdbc-url=jdbc:mysql://192.168.241.198:3306/shop_ds_1?serverTimezone=UTC&useSSL=false&useUnicode=true&characterEncoding=UTF-8
spring.shardingsphere.datasource.ds1.username=root
spring.shardingsphere.datasource.ds1.password=root

# 分库策略 根据id取模确定数据进哪个数据库
spring.shardingsphere.sharding.default-database-strategy.inline.sharding-column=user_id
spring.shardingsphere.sharding.default-database-strategy.inline.algorithm-expression=ds$->{user_id % 2}
# 绑定表
spring.shardingsphere.sharding.binding-tables=t_order,t_order_item
#广播表
spring.shardingsphere.sharding.broadcast-tables=t_address

# t_order表策略
```

```
spring.shardingsphere.sharding.tables.t_order.actual-data-nodes=ds$->
  {0..1}.t_order
# 使用SNOWFLAKE算法生成主键
spring.shardingsphere.sharding.tables.t_order.key-generator.column=order_id
spring.shardingsphere.sharding.tables.t_order.key-generator.type=SNOWFLAKE
spring.shardingsphere.sharding.tables.t_order.key-
  generator.props.worker.id=123
# t_order_item表策略
spring.shardingsphere.sharding.tables.t_order_item.actual-data-nodes=ds$->
  {0..1}.t_order_item
# 使用SNOWFLAKE算法生成主键
spring.shardingsphere.sharding.tables.t_order_item.key-generator.column=ord
  er_item_id
spring.shardingsphere.sharding.tables.t_order_item.key-generator.type=SNOWF
  LAKE
spring.shardingsphere.sharding.tables.t_order_item.key-generator.props.work
  er.id=123
```

分库分表

```
# 配置ds0 和ds1两个数据源
spring.shardingsphere.datasource.names=ds0,ds1


#ds0 配置
spring.shardingsphere.datasource.ds0.type=com.zaxxer.hikari.HikariDataSourc
  e
spring.shardingsphere.datasource.ds0.driver-class-name=com.mysql.jdbc.Drive
  r
spring.shardingsphere.datasource.ds0.jdbc-
  url=jdbc:mysql://192.168.241.198:3306/shop_ds_0?
  serverTimezone=UTC&useSSL=false&useUnicode=true&characterEncoding=UTF-8
spring.shardingsphere.datasource.ds0.username=root
spring.shardingsphere.datasource.ds0.password=root
#ds1 配置
spring.shardingsphere.datasource.ds1.type=com.zaxxer.hikari.HikariDataSourc
  e
spring.shardingsphere.datasource.ds1.driver-class-name=com.mysql.jdbc.Drive
  r
spring.shardingsphere.datasource.ds1.jdbc-
  url=jdbc:mysql://192.168.241.198:3306/shop_ds_1?
  serverTimezone=UTC&useSSL=false&useUnicode=true&characterEncoding=UTF-8
spring.shardingsphere.datasource.ds1.username=root
spring.shardingsphere.datasource.ds1.password=root
```

```
# 分库策略 根据id取模确定数据进哪个数据库
spring.shardingsphere.sharding.default-database-strategy.inline.sharding-co
  lumn=user_id
spring.shardingsphere.sharding.default-database-strategy.inline.algorithm-e
  xpression=ds$->{user_id % 2}
# 绑定表（好像没什么卵用）
spring.shardingsphere.sharding.binding-tables=t_order,t_order_item
# 广播表
spring.shardingsphere.sharding.broadcast-tables=t_address

# 具体分表策略
# 节点 ds0.t_order_0,ds0.t_order_1,ds1.t_order_0,ds1.t_order_1
spring.shardingsphere.sharding.tables.t_order.actual-data-nodes=ds$->
  {0..1}.t_order_$->{0..1}
# 分表字段id
spring.shardingsphere.sharding.tables.t_order.table-strategy.inline.shardin
  g-column=order_id
# 分表策略 根据id取模,确定数据最终落在那个表中
spring.shardingsphere.sharding.tables.t_order.table-strategy.inline.algorit
  hm-expression = t_order_$->{order_id % 2}
# 使用SNOWFLAKE算法生成主键
spring.shardingsphere.sharding.tables.t_order.key-generator.column=order_id
spring.shardingsphere.sharding.tables.t_order.key-generator.type=SNOWFLAKE
spring.shardingsphere.sharding.tables.t_order.key-
  generator.props.worker.id=123

# 节点 ds0.t_order_item_0,ds0.t_order_item_1,ds1.t_order_item_0,ds1.t_order_
  item_1
spring.shardingsphere.sharding.tables.t_order_item.actual-data-nodes=ds$->
  {0..1}.t_order_item_$->{0..1}
# 分表字段id
spring.shardingsphere.sharding.tables.t_order_item.table-strategy.inline.sh
  arding-column=order_id
# 分表策略 根据id取模,确定数据最终落在那个表中
spring.shardingsphere.sharding.tables.t_order_item.table-strategy.inline.al
  gorithm-expression=t_order_item_$->{order_id % 2}
# 使用SNOWFLAKE算法生成主键
spring.shardingsphere.sharding.tables.t_order_item.key-generator.column=ord
  er_item_id
spring.shardingsphere.sharding.tables.t_order_item.key-generator.type=SNOWF
  LAKE
spring.shardingsphere.sharding.tables.t_order_item.key-generator.props.work
  er.id=123
```

读写分离

```
#shardingsphere 读写分离,master-slave,可以一主多从
spring.shardingsphere.datasource.names=ds-master,ds-slave0
#主库
spring.shardingsphere.datasource.ds-master.type=com.zaxxer.hikari.HikariDat
  aSource
spring.shardingsphere.datasource.ds-master.driver-class-
  name=com.mysql.jdbc.Driver
spring.shardingsphere.datasource.ds-master.jdbc-url=jdbc:mysql://192.168.24
  1.198:3306/shop_ds_master?
  serverTimezone=UTC&useSSL=false&useUnicode=true&characterEncoding=UTF-8
spring.shardingsphere.datasource.ds-master.username=root
spring.shardingsphere.datasource.ds-master.password=root
#从库0
spring.shardingsphere.datasource.ds-slave0.type=com.zaxxer.hikari.HikariDat
  aSource
spring.shardingsphere.datasource.ds-slave0.driver-class-
  name=com.mysql.jdbc.Driver
spring.shardingsphere.datasource.ds-slave0.jdbc-url=jdbc:mysql://192.168.24
  1.199:3306/shop_ds_slave?
  serverTimezone=UTC&useSSL=false&useUnicode=true&characterEncoding=UTF-8
spring.shardingsphere.datasource.ds-slave0.username=root
spring.shardingsphere.datasource.ds-slave0.password=root
#从库1
#spring.shardingsphere.datasource.ds-slave1.type=com.zaxxer.hikari.HikariDa
  taSource
#spring.shardingsphere.datasource.ds-slave1.driver-class-name=com.mysql.jdb
  c.Driver
#spring.shardingsphere.datasource.ds-slave1.jdbc-url=jdbc:mysql://192.168.2
  41.199:3306/shop_ds_slave1?serverTimezone=UTC&useSSL=false&useUnicode=tru
  e&characterEncoding=UTF-8
#spring.shardingsphere.datasource.ds-slave1.username=root
#spring.shardingsphere.datasource.ds-slave1.password=root

#读写分离主从规则设置，当有2个以上从库时，从库读采用轮询的负载均衡机制(也可设置为
  随机读)
spring.shardingsphere.masterslave.load-balance-algorithm-type=round_robin
spring.shardingsphere.masterslave.name=ds
spring.shardingsphere.masterslave.master-data-source-name=ds-master
#如果有多个从库,在本配置项后加:,ds-slave1 即可
spring.shardingsphere.masterslave.slave-data-source-names=ds-slave0
```

读写分离+分库分表

```
#shardingsphere 读写分离,master-slave,可以一主多从
spring.shardingsphere.datasource.names=ds-master0,ds-slave0,ds-master1,ds-s
  lave1
#主库0
spring.shardingsphere.datasource.ds-master0.type=com.zaxxer.hikari.HikariDa
  taSource
spring.shardingsphere.datasource.ds-master0.driver-class-name=com.mysql.jdb
  c.Driver
spring.shardingsphere.datasource.ds-master0.jdbc-url=jdbc:mysql://192.168.2
  41.198:3306/shop_ds_master?serverTimezone=UTC&useSSL=false&useUnicode=tru
  e&characterEncoding=UTF-8
spring.shardingsphere.datasource.ds-master0.username=root
spring.shardingsphere.datasource.ds-master0.password=root
#从库0
spring.shardingsphere.datasource.ds-slave0.type=com.zaxxer.hikari.HikariDat
  aSource
spring.shardingsphere.datasource.ds-slave0.driver-class-
  name=com.mysql.jdbc.Driver
spring.shardingsphere.datasource.ds-slave0.jdbc-url=jdbc:mysql://192.168.24
  1.199:3306/shop_ds_slave?
  serverTimezone=UTC&useSSL=false&useUnicode=true&characterEncoding=UTF-8
spring.shardingsphere.datasource.ds-slave0.username=root
spring.shardingsphere.datasource.ds-slave0.password=root
#主库1
spring.shardingsphere.datasource.ds-master1.type=com.zaxxer.hikari.HikariDa
  taSource
spring.shardingsphere.datasource.ds-master1.driver-class-name=com.mysql.jdb
  c.Driver
spring.shardingsphere.datasource.ds-master1.jdbc-url=jdbc:mysql://192.168.2
  41.198:3306/shop_ds_master1?serverTimezone=UTC&useSSL=false&useUnicode=tr
  ue&characterEncoding=UTF-8
spring.shardingsphere.datasource.ds-master1.username=root
spring.shardingsphere.datasource.ds-master1.password=root
#从库1
spring.shardingsphere.datasource.ds-slave1.type=com.zaxxer.hikari.HikariDat
  aSource
spring.shardingsphere.datasource.ds-slave1.driver-class-
  name=com.mysql.jdbc.Driver
spring.shardingsphere.datasource.ds-slave1.jdbc-url=jdbc:mysql://192.168.24
  1.199:3306/shop_ds_slave1?
  serverTimezone=UTC&useSSL=false&useUnicode=true&characterEncoding=UTF-8
spring.shardingsphere.datasource.ds-slave1.username=root
```

```
spring.shardingsphere.datasource.ds-slave1.password=root

# 分库策略 根据id取模确定数据进哪个数据库
spring.shardingsphere.sharding.default-database-strategy.inline.sharding-co
  lumn=user_id
spring.shardingsphere.sharding.default-database-strategy.inline.algorithm-e
  xpression=ds_$->{user_id % 2}
#绑定表
sharding.jdbc.config.sharding.binding-tables=t_order,t_order_item
spring.shardingsphere.sharding.broadcast-tables=t_address

#分表策略
spring.shardingsphere.sharding.tables.t_order.actual-data-nodes=ds_$->
  {0..1}.t_order_$->{0..1}
spring.shardingsphere.sharding.tables.t_order.table-strategy.inline.shardin
  g-column=order_id
spring.shardingsphere.sharding.tables.t_order.table-strategy.inline.algorit
  hm-expression=t_order_$->{order_id % 2}
spring.shardingsphere.sharding.tables.t_order.key-generator.column=order_id
spring.shardingsphere.sharding.tables.t_order.key-generator.type=SNOWFLAKE
spring.shardingsphere.sharding.tables.t_order.key-
  generator.props.worker.id=123
spring.shardingsphere.sharding.tables.t_order_item.actual-data-nodes=ds_$->
  {0..1}.t_order_item_$->{0..1}
spring.shardingsphere.sharding.tables.t_order_item.table-strategy.inline.sh
  arding-column=order_id
spring.shardingsphere.sharding.tables.t_order_item.table-strategy.inline.al
  gorithm-expression=t_order_item_$->{order_id % 2}
spring.shardingsphere.sharding.tables.t_order_item.key-generator.column=ord
  er_item_id
spring.shardingsphere.sharding.tables.t_order_item.key-generator.type=SNOWF
  LAKE
spring.shardingsphere.sharding.tables.t_order_item.key-generator.props.work
  er.id=123

#读写分离数据源0
spring.shardingsphere.sharding.master-slave-rules.ds_0.master-data-source-n
  ame=ds-master0
spring.shardingsphere.sharding.master-slave-rules.ds_0.slave-data-source-na
  mes=ds-slave0
#读写分离主从规则设置，当有2个以上从库时，从库读采用轮询的负载均衡机制
spring.shardingsphere.sharding.master-slave-rules.ds_0.load-balance-algorit
  hm-type=ROUND_ROBIN
```

```
#读写分离数据源1
spring.shardingsphere.sharding.master-slave-rules.ds_1.master-data-source-n
  ame=ds-master1
spring.shardingsphere.sharding.master-slave-rules.ds_1.slave-data-source-na
  mes=ds-slave1
#读写分离主从规则设置，当有2个以上从库时，从库读采用轮询的负载均衡机制
spring.shardingsphere.sharding.master-slave-rules.ds_1.load-balance-algorit
  hm-type=ROUND_ROBIN
```