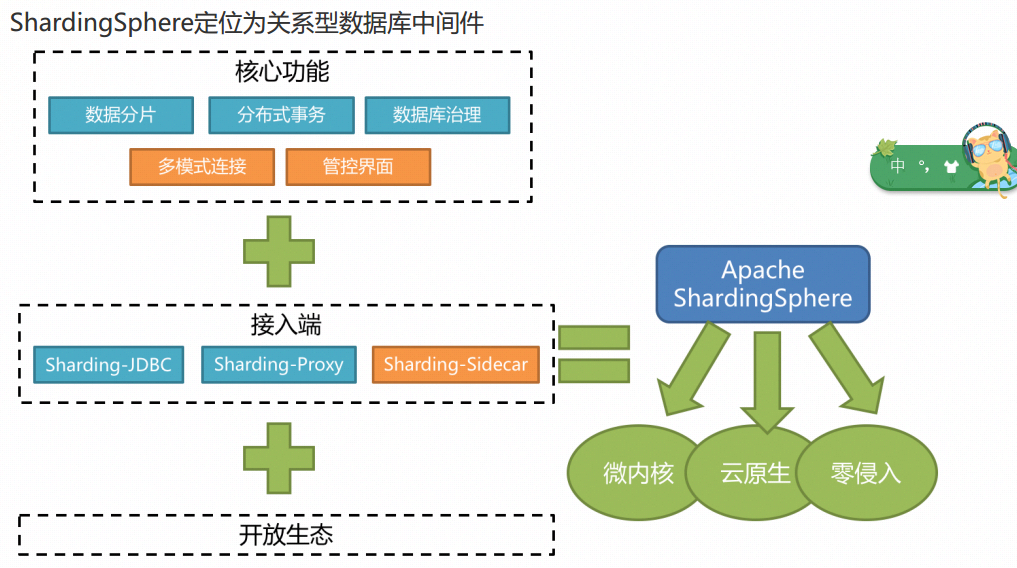
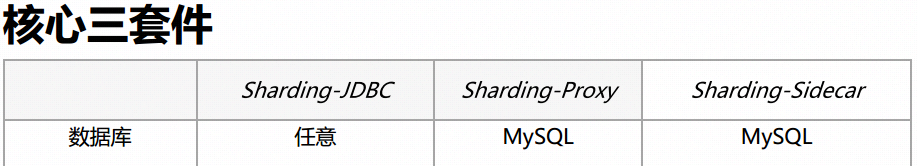
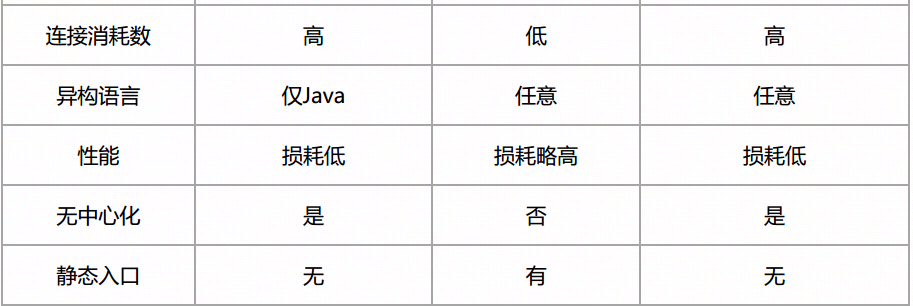
# 1、shardingsphere的简介

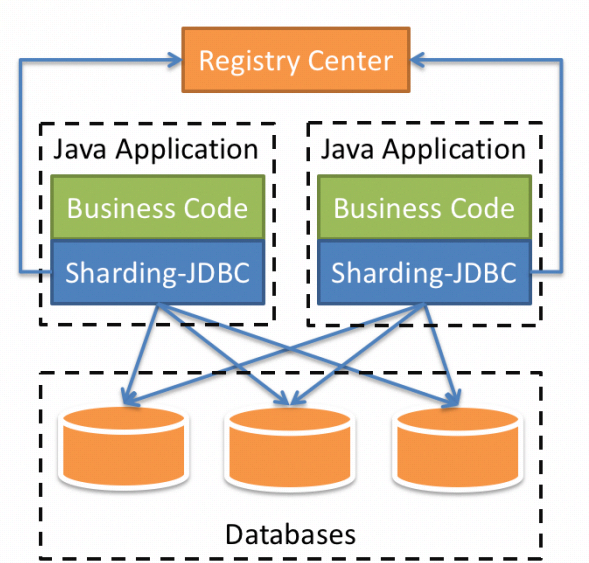




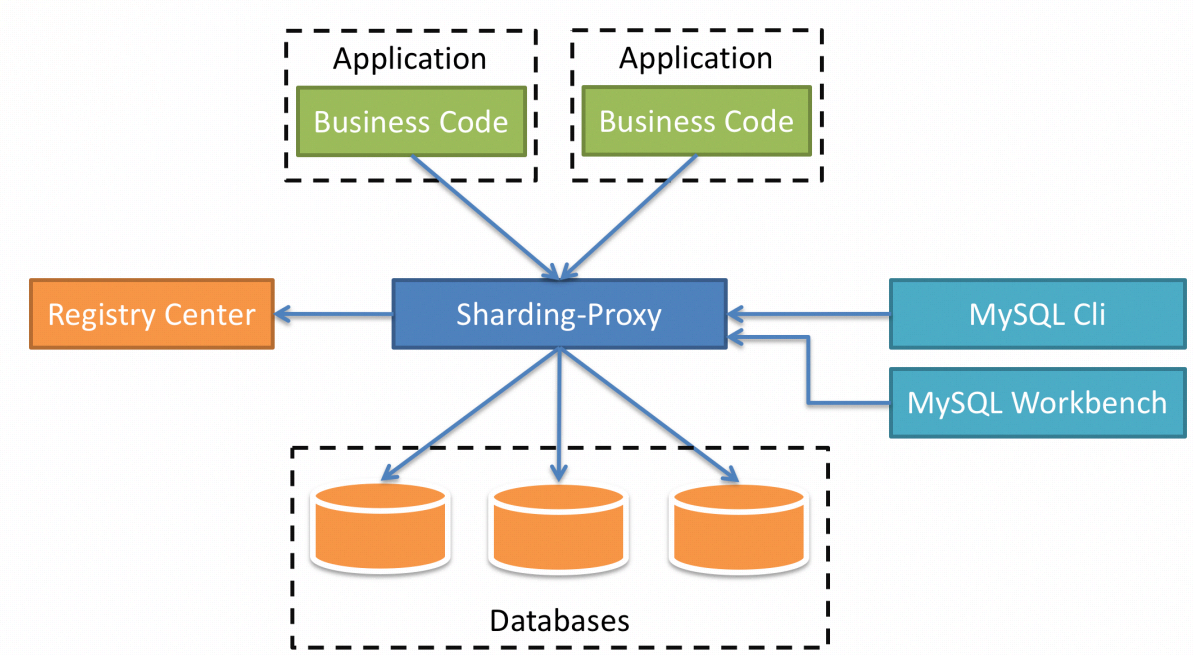




Sharding-JDBC  
客户端直连数据库， 以jar包形式提供服务， 无需额外部署和依赖， 可理解为增强版的  
JDBC驱动， 完全兼容JDBC和各种ORM框架。  
适用于任何基于JDBC的ORM框架： JPA, Hibernate, Mybatis, Spring JDBC Template或直接使用JDBC。  
支持任何第三方的数据库连接池： DBCP, C3P0, BoneCP, Druid, HikariCP等。  
支持任意实现JDBC规范的数据库。 支持MySQL， Oracle， SQLServer， PostgreSQL等遵循SQL92标准的数据库。



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。



# 2、shardingsphere的使用

## 2.1 shardingsphere的核心概念

### 1.1 shardingsphere中的实际有逻辑表概念

**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。，即ds\_0数据源下t\_order\_0实际表

**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。原理是两张表使用同一个字段进行分表，即查询时只需要关联相同的数据源下对应下标相同的表即可，如2库2表则需要查询4次，即a1只需要跟b1关联查询，a1不需要跟b2关联

**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。

### 1.2 数据分片

数据库的水平拆分与垂直拆分

## 2.2 shardingsphere的快速集成

1、引入依赖

<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>

2、配置文件的配置

### 1、分库不分表

# 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. HikariDataSourc  
e  
#数据库驱动  
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‐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 分库分表  
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

### 2、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‐column=user\_id  
spring. shardingsphere. sharding. default‐database‐strategy. inline. algorithm‐expression=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. sharding‐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. algorithm‐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

# worker. id机器序号  
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. sharding‐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. algorithm‐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=order\_item\_id  
spring. shardingsphere. sharding. tables. t\_order\_item. key‐generator. type=SNOWFLAKE  
spring. shardingsphere. sharding. tables. t\_order\_item. key‐generator. props. worker. id=123

### 3、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. HikariDataSource  
spring. shardingsphere. datasource. ds‐master. driver‐class‐name=com. mysql. jdbc. Driver  
spring. shardingsphere. datasource. ds‐master. jdbc‐url=jdbc: mysql: //192. 168. 241. 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. HikariDataSource  
spring. shardingsphere. datasource. ds‐slave0. driver‐class‐name=com. mysql. jdbc. Driver

spring. shardingsphere. datasource. ds‐slave0. jdbc‐url=jdbc: mysql: //192. 168. 241. 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. HikariDataSource  
#spring. shardingsphere. datasource. ds‐slave1. driver‐class‐name=com. mysql. jdbc. Driver  
#spring. shardingsphere. datasource. ds‐slave1. jdbc‐url=jdbc: mysql: //192. 168. 241. 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  
#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

### 4、读写分离加分库分表

读写分离加分库分表时，所有的写走主库，读走主库下的从库，主库设置时要注意一定要在主MySQL上

#shardingsphere 读写分离, master‐slave, 可以一主多从  
spring. shardingsphere. datasource. names=ds‐master0, ds‐slave0, ds‐master1, ds‐slave1  
#主库0  
spring. shardingsphere. datasource. ds‐master0. type=com. zaxxer.

hikari. HikariDataSource  
spring. shardingsphere. datasource. ds‐master0. driver‐class‐name=com. mysql. jdbc. Driver  
spring. shardingsphere. datasource. ds‐master0. jdbc‐url=jdbc: mysql: //192. 168. 241. 198: 3306/shop\_ds\_master?serverTimezone=UTC&

useSSL=false&useUnicode=true&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. HikariDataSource  
spring. shardingsphere. datasource. ds‐slave0. driver‐class‐  
name=com. mysql. jdbc. Driver  
spring. shardingsphere. datasource. ds‐slave0. jdbc‐url=jdbc: mysql: //192. 168. 241. 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. HikariDataSource  
spring. shardingsphere. datasource. ds‐master1. driver‐class‐name=com. mysql. jdbc. Driver  
spring. shardingsphere. datasource. ds‐master1. jdbc‐url=jdbc: mysql: //192. 168. 241. 198: 3306/shop\_ds\_master1?serverTimezone=UTC&

useSSL=false&useUnicode=true&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. HikariDataSource  
spring. shardingsphere. datasource. ds‐slave1. driver‐class‐  
name=com. mysql. jdbc. Driver  
spring. shardingsphere. datasource. ds‐slave1. jdbc‐url=jdbc: mysql: //192. 168. 241. 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‐column=user\_id  
spring. shardingsphere. sharding. default‐database‐strategy. inline. algorithm‐expression=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. sharding‐column=order\_id  
spring. shardingsphere. sharding. tables. t\_order. table‐strategy. inline. algorithm‐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. sharding‐column=order\_id  
spring. shardingsphere. sharding. tables. t\_order\_item. table‐strategy. inline. algorithm‐expression=t\_order\_item\_$‐>{order\_id % 2}  
spring. shardingsphere. sharding. tables. t\_order\_item. key‐generator. column=order\_item\_id  
spring. shardingsphere. sharding. tables. t\_order\_item. key‐generator. type=SNOWFLAKE  
spring. shardingsphere. sharding. tables. t\_order\_item. key‐generator. props. worker. id=123  
#读写分离数据源0  
spring. shardingsphere. sharding. master‐slave‐rules. ds\_0. master‐data‐source‐name=ds‐master0  
spring. shardingsphere. sharding. master‐slave‐rules. ds\_0. slave‐data‐source‐names=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‐algorithm‐type=ROUND\_ROBIN  
#读写分离数据源1  
spring. shardingsphere. sharding. master‐slave‐rules. ds\_1. master‐data‐source‐name=ds‐master1  
spring. shardingsphere. sharding. master‐slave‐rules. ds\_1. slave‐data‐source‐names=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‐algorithm‐type=ROUND\_ROBIN