# 一、sharding-jdbc

## **简介**

Sharding-JDBC直接封装JDBC API，可以理解为增强版的JDBC驱动，旧代码迁移成本几乎为零：

* 可适用于任何基于java的ORM框架，如：JPA, Hibernate, Mybatis, Spring JDBC Template或直接使用JDBC。
* 可基于任何第三方的数据库连接池，如：DBCP, C3P0, BoneCP, Druid等。
* 理论上可支持任意实现JDBC规范的数据库。虽然目前仅支持MySQL，但已有支持Oracle，SQLServer，DB2等数据库的计划。

Sharding-JDBC定位为轻量级java框架，使用客户端直连数据库，以jar包形式提供服务，未使用中间层，无需额外部署，无其他依赖，DBA也无需改变原有的运维方式。SQL解析使用Druid解析器，是目前性能最高的SQL解析器。

Sharding-JDBC功能灵活且全面：

* 分片策略灵活，可支持=，BETWEEN，IN等多维度分片，也可支持多分片键共用。
* SQL解析功能完善，支持聚合，分组，排序，Limit，OR等查询，并且支持Binding Table以及笛卡尔积的表查询。
* 支持柔性事务(目前仅最大努力送达型)。
* 支持读写分离。

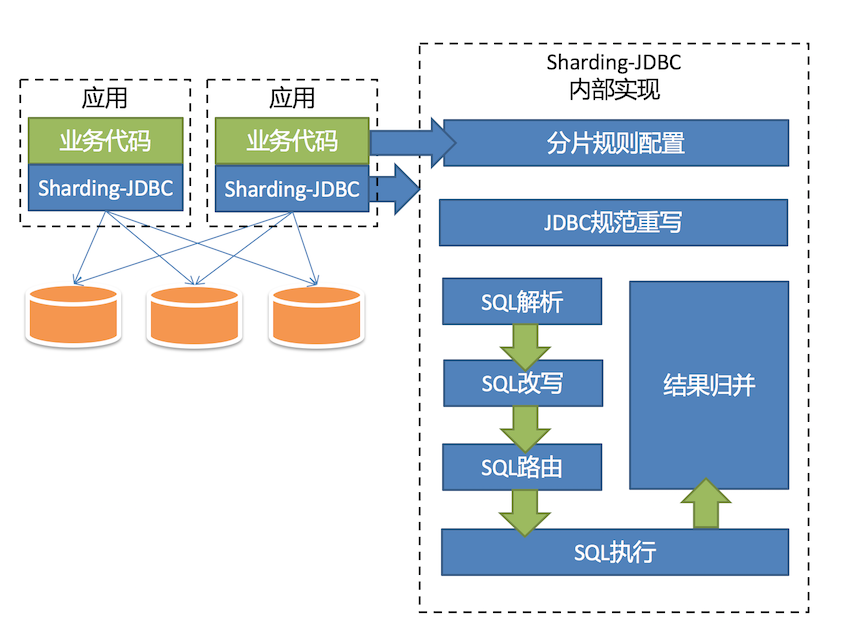
Sharding-JDBC配置多样：

* 可支持YAML和Spring命名空间配置
* 灵活多样的inline方式

以下是常见的分库分表产品和Sharding-JDBC的对比：



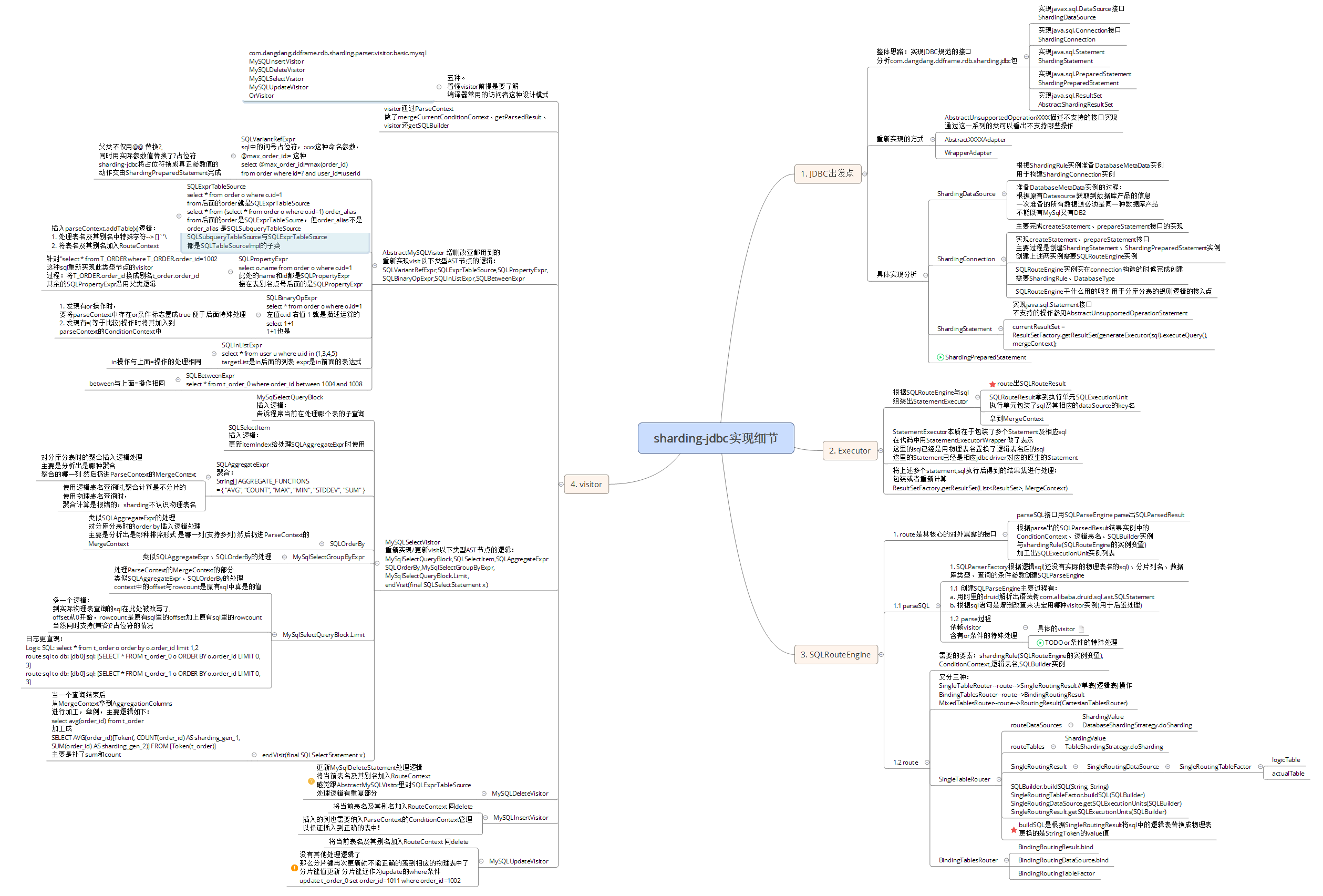
## **整体架构图**



参考资料：

http://chuansong.me/n/2361944

## sharding-jdbc实现细节



## 使用限制

### Statement和PreparedStatement接口

* 不支持返回多结果集的语句（即存储过程，非SELECT多条数据）
* 不支持国际化字符的操作

### JDBC 4.1

* 不支持JDBC 4.1接口新功能

查询所有未支持方法，请阅读com.dangdang.ddframe.rdb.sharding.jdbc.unsupported包。

## SQL语句限制

### 不支持DDL语句

### 不支持UNION和UNION ALL

### 不支持特殊INSERT

每条INSERT语句只能插入一条数据，不支持VALUES后有多行数据的语句

## 配置

### **引入maven依赖**

<!-- 引入sharding-jdbc核心模块 -->

<dependency>

<groupId>com.dangdang</groupId>

<artifactId>sharding-jdbc-core</artifactId>

<version>${sharding-jdbc.version}</version>

</dependency>

<!-- <dependency>

<groupId>com.dangdang</groupId>

<artifactId>sharding-jdbc-transaction</artifactId>

<version>${sharding-jdbc.version}</version>

</dependency> -->

<dependency>

<groupId>com.dangdang</groupId>

<artifactId>sharding-jdbc-config-spring</artifactId>

<version>${sharding-jdbc.version}</version>

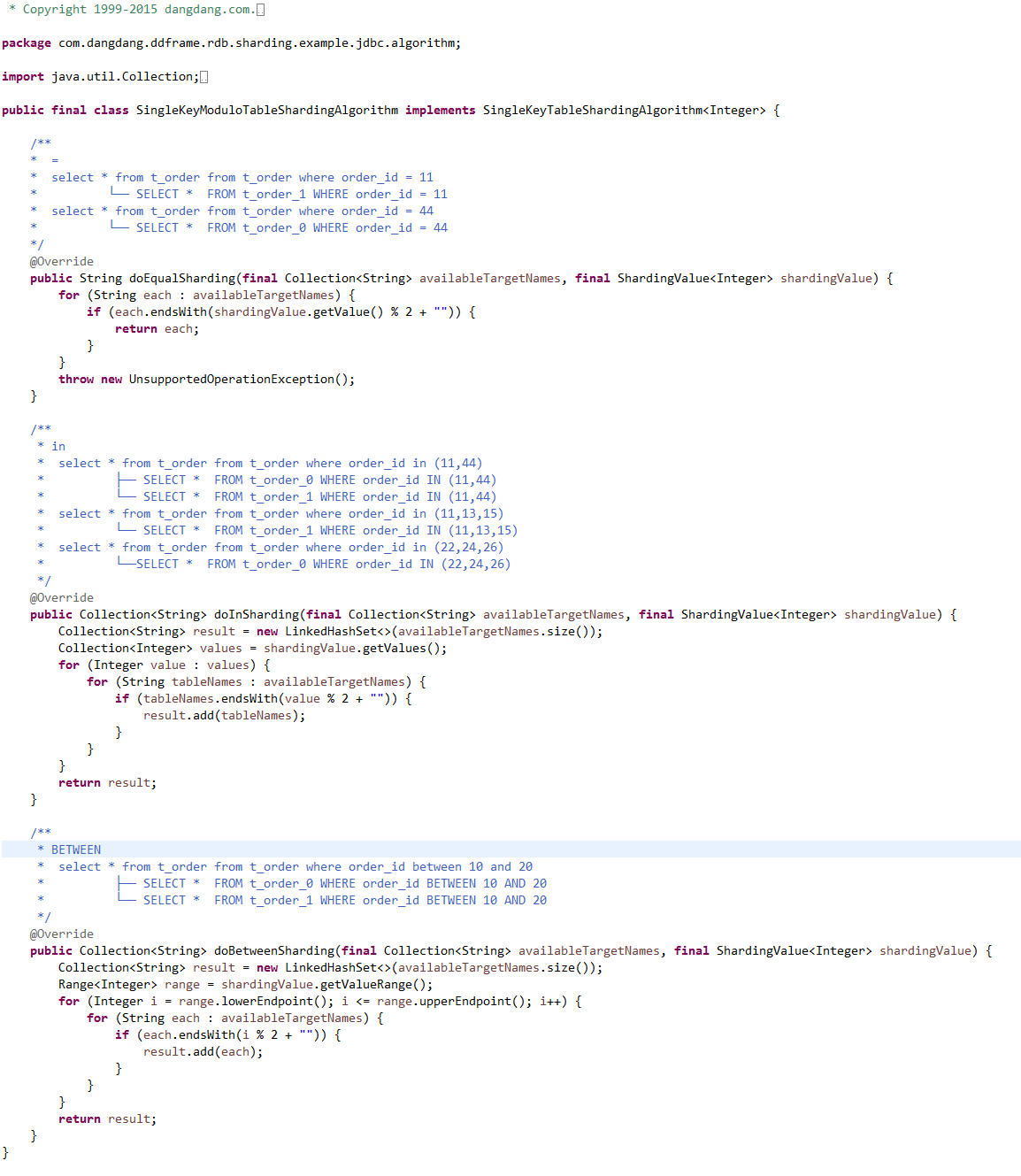
</dependency>

现在最新版本为1.3.3

### **编写分库计算规则**



### **编写分表计算规则**

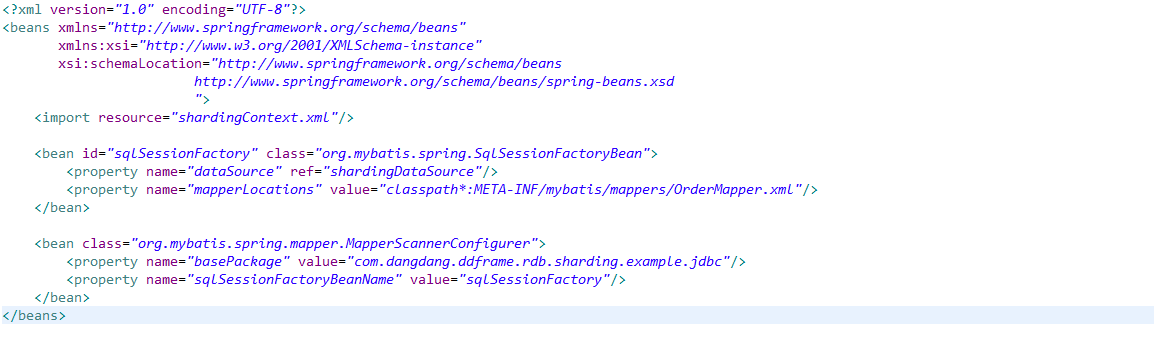


现测试编写两个表，如果后续表可以将取模大小弄成配置文件

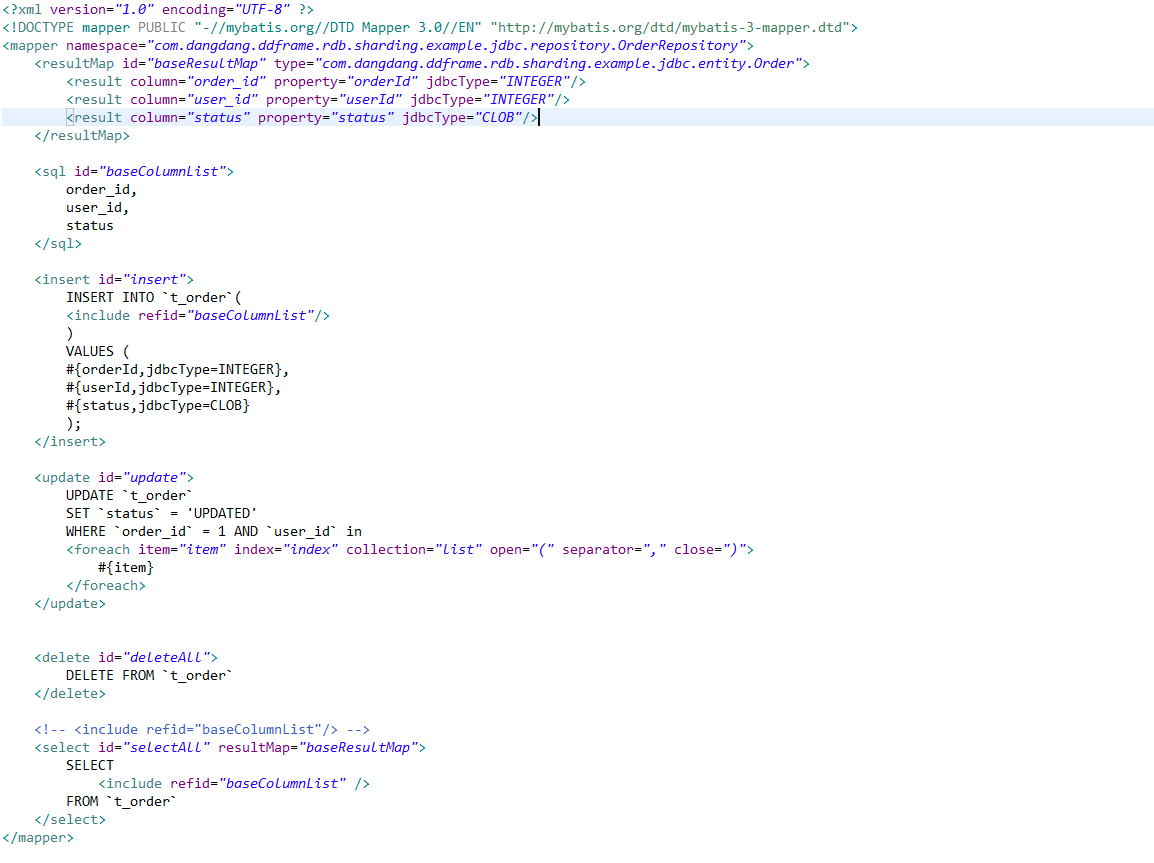
### **配置策略文件**



### **Mybatis配置**



### **Mybatis mapper文件**



Mapper文件中的表名应写成【配置策略】文件中的logic-table名称

### **修改主键生成方法**

因为数据分表保存，不能使用identify方式生成数据表主键。如果主键是String类型，可以考虑使用uuid生成方法，但它查询效率会相对比较低。

如果使用long型主键，可以使用其他方式，一定要确保各个子表中的主键不重复。

参考资料：

<https://github.com/dangdangdotcom/sharding-jdbc/blob/master/README_cn.md>

<https://github.com/dangdangdotcom/sharding-jdbc/releases>

<http://blog.csdn.net/xiexingshishu/article/details/50668137>

# **二、Mysql MERGE**

在Mysql数据库中，[Mysql Merge](http://database.51cto.com/art/201010/230057.htm)表有点类似于视图。

## **优点**

1、分离静态的和动态的数据  
2、利用结构接近的的数据来优化查询  
3、 查询时可以访问更少的数据  
4、更容易维护大数据集  
5、可以通过修改.mrg文件来修改Merge表，当然也可以用alter进行修改，修改后要通过FLUSH TABLES刷新表缓存，此法可以动态增加减少子表

## **注意点**

1、每个子表的结构必须一致，主表和子表的结构需要一致，

2、每个子表的索引在merge表中都会存在，所以在merge表中不能根据该索引进行唯一性检索。

3、子表需要是MyISAM引擎

4、AUTO\_INCREMENT 不会按照你所期望的方式工作。

5、创建Mysql Merge表的参数 INSERT\_METHOD有几个参数 。  
 LAST  如果你执行insert 指令来操作merge表时，插入操作会把数据添加到最后一个子表中。

FIRST  同理，执行插入数据时会把数据添加到第一个子表中。

1. 一个 merge 表不能在整个表上维持 unique 约束。当你执行一个 insert，数据进入第一个或者最后一个 myisam 表（取决于 insert\_method 选项的值）。mysql 确保唯一键值在那个 myisam 表里保持唯一，但不是跨集合里所有的表。
2. 也需要通过编码进行分表插入操作

## **例：**

#初始化数据表，方便测试

CREATE TABLE member(

id BIGINT AUTO\_INCREMENT PRIMARY KEY,

NAME VARCHAR(20),

sex TINYINT NOT NULL DEFAULT '0'

)ENGINE=MYISAM DEFAULT CHARSET=utf8 AUTO\_INCREMENT=1;

#初始化数据

INSERT INTO member(id,NAME,sex) VALUES (1,'jacson','0');

#重复执行，生成多条数据

INSERT INTO member(NAME,sex) SELECT NAME,sex FROM member;

###############################

#对表进行拆分

DROP TABLE IF EXISTS tb\_member1;

CREATE TABLE tb\_member1(

id BIGINT PRIMARY KEY AUTO\_INCREMENT ,

NAME VARCHAR(20),

sex TINYINT NOT NULL DEFAULT '0'

)ENGINE=MYISAM DEFAULT CHARSET=utf8 AUTO\_INCREMENT=1 ;

DROP TABLE IF EXISTS tb\_member2;

CREATE TABLE tb\_member2(

id BIGINT PRIMARY KEY AUTO\_INCREMENT ,

NAME VARCHAR(20),

sex TINYINT NOT NULL DEFAULT '0'

)ENGINE=MYISAM DEFAULT CHARSET=utf8 AUTO\_INCREMENT=1 ;

#######################################

DROP TABLE IF EXISTS tb\_member;

CREATE TABLE tb\_member(

id BIGINT PRIMARY KEY AUTO\_INCREMENT ,

NAME VARCHAR(20),

sex TINYINT NOT NULL DEFAULT '0'

)ENGINE=MERGE UNION=(tb\_member1,tb\_member2) INSERT\_METHOD=LAST CHARSET=utf8 AUTO\_INCREMENT=1 ;

#不指定子表进行数据插入

INSERT INTO tb\_member(NAME,sex) VALUES ('jacson2','2');

参考资料：

<http://dev.mysql.com/doc/refman/5.7/en/merge-storage-engine.html>

# 生成全局唯一ID

## **[Redis](http://lib.csdn.net/base/redis" \o "Redis知识库" \t "http://blog.csdn.net/longjef/article/details/_blank)**生成ID****

当使用数据库来生成ID性能不够要求的时候，我们可以尝试使用Redis来生成ID。这主要依赖于Redis是单线程的，所以也可以用于生成全局唯一的ID。可以用Redis的原子操作 INCR和INCRBY来实现。  
  
可以使用Redis集群来获取更高的吞吐量。假如一个集群中有5台Redis A,B,C,D,E。可以初始化每台Redis的值分别是1,2,3,4,5，然后步长都是5。各个Redis生成的ID为：

|  |
| --- |
|  |

|  |
| --- |
| A：1,6,11,16,21  B：2,7,12,17,22  C：3,8,13,18,23  D：4,9,14,19,24  E：5,10,15,20,25 |

## CommonSelfIdGenerator生成

