# ShardingSphere-JDBC

## 1.ShardingSphere-JDBC简介

ShardingJdbc的官网：

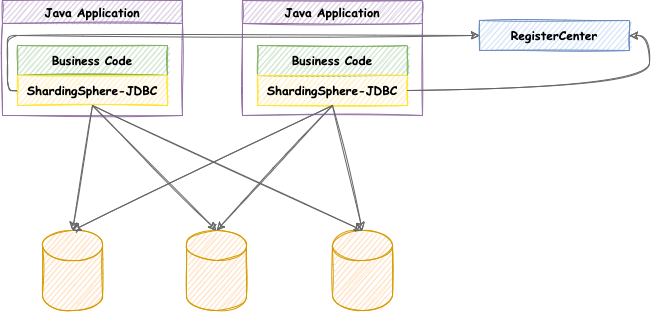
<http://shardingsphere.apache.org/index_zh.html>

Apache ShardingSphere 由 JDBC、Proxy 和 Sidecar（规划中）这 3 款既能够独立部署，又支持混合部署配合使用的产品组成。 它们均提供标准化的基于数据库作为存储节点的增量功能，可适用于如 Java 同构、异构语言、云原生等各种多样化的应用场景。

定位为轻量级 Java 框架，在 Java 的 JDBC 层提供的额外服务。 它使用客户端直连数据库，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。

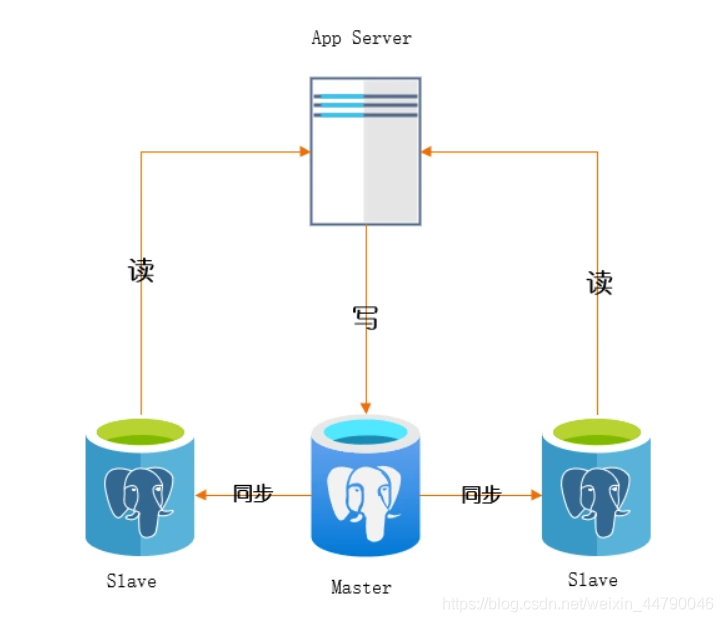
* 适用于任何基于 JDBC 的 ORM 框架，如：JPA, Hibernate, Mybatis, Spring JDBC Template 或直接使用 JDBC；
* 支持任何第三方的数据库连接池，如：DBCP, C3P0, BoneCP, HikariCP 等；
* 支持任意实现 JDBC 规范的数据库，目前支持 MySQL，PostgreSQL，Oracle，SQLServer 以及任何可使用 JDBC 访问的数据库。

SQL 解析 -> SQL 路由 -> SQL 改写 -> SQL 执行 -> 结果归并



### 1.1 ShardingSphere-JDBC读写分离

读写分离就是对于一条SQL该选择哪一个数据库去执行，至于谁来做选择数据库这件事，有两个，要么使用中间件帮我们做，要么程序自己做。一般来说，读写分离有两种实现方式。第一种是依靠中间件MyCat，也就是说应用程序连接到中间件，中间件帮我们做SQL分离，去选择指定的数据源；第二种是应用程序自己去做分离。这里我用程序自己来做，主要是利用Spring提供的路由数据源，以及AOP。



### 1.2读写分离代码实现

### 1.2.1 数据库环境准备

提前准备好mysql一主一从数据库，做好主从同步开启binlog日志。(可以查看mysql的主从数据库搭建)

### 1.2.2 新建Springboot工程，引入相关依赖

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

</dependency>

<dependency>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

</dependency>

<!--shardingsphere 读写分离 分库分表-->

<dependency>

<groupId>org.apache.shardingsphere</groupId>

<artifactId>sharding-jdbc-spring-boot-starter</artifactId>

<version>4.0.0-RC1</version>

</dependency>

<dependency>

<groupId>com.alibaba</groupId>

<artifactId>druid-spring-boot-starter</artifactId>

<version>1.1.21</version>

</dependency>

<!-- 数据库驱动 -->

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

</dependency>

<!-- mybatis-plus -->

<dependency>

<groupId>com.baomidou</groupId>

<artifactId>mybatis-plus-boot-starter</artifactId>

<!--<version>3.4.3.1</version>-->

<version>3.5.1</version>

</dependency>

<!--knife4j 丝袜哥增强版-->

<dependency>

<groupId>com.github.xiaoymin</groupId>

<artifactId>knife4j-spring-boot-starter</artifactId>

<version>2.0.9</version>

</dependency>

</dependencies>

### 1.2.3 application.properties配置文件

#执行日志打印

mybatis-plus.configuration.log-impl=org.apache.ibatis.logging.stdout.StdOutImpl

##### sharding的读写分离

spring.main.allow-bean-definition-overriding=true

#显示sql

spring.shardingsphere.props.sql.show=true

#配置数据源

spring.shardingsphere.datasource.names=ds1,ds2

#master-ds1数据库连接信息

spring.shardingsphere.datasource.ds1.type=com.alibaba.druid.pool.DruidDataSource

spring.shardingsphere.datasource.ds1.driver-class-name=com.mysql.cj.jdbc.Driver

spring.shardingsphere.datasource.ds1.url=jdbc:mysql://10.207.0.161:3306/testshardingjdbc?useUnicode=true&characterEncoding=utf-8&serverTimezone=Asia/Shanghai

spring.shardingsphere.datasource.ds1.username=root

spring.shardingsphere.datasource.ds1.password=123456

spring.shardingsphere.datasource.ds1.maxPoolSize=100

spring.shardingsphere.datasource.ds1.minPoolSize=5

#slave-ds2数据库连接信息

spring.shardingsphere.datasource.ds2.type=com.alibaba.druid.pool.DruidDataSource

spring.shardingsphere.datasource.ds2.driver-class-name=com.mysql.cj.jdbc.Driver

spring.shardingsphere.datasource.ds2.url=jdbc:mysql://127.0.0.1:3306/testshardingjdbc?useUnicode=true&characterEncoding=utf-8&serverTimezone=Asia/Shanghai

spring.shardingsphere.datasource.ds2.username=root

spring.shardingsphere.datasource.ds2.password=123456

spring.shardingsphere.datasource.ds2.maxPoolSize=100

spring.shardingsphere.datasource.ds2.minPoolSize=5

#配置默认数据源ds1 默认数据源,主要用于写

spring.shardingsphere.sharding.default-data-source-name=ds1

#配置主从名称

spring.shardingsphere.masterslave.name=ms

#置主库master,负责数据的写入

spring.shardingsphere.masterslave.master-data-source-name=ds1

#配置从库slave节点

spring.shardingsphere.masterslave.slave-data-source-names=ds2

#配置slave节点的负载均衡均衡策略,采用轮询机制

spring.shardingsphere.masterslave.load-balance-algorithm-type=round\_robin

### 1.2.4 创建t\_user表

CREATE TABLE `t\_user` (

`id` int NOT NULL AUTO\_INCREMENT,

`nickname` varchar(100) CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci NULL DEFAULT NULL,

`password` varchar(100) CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci NULL DEFAULT NULL,

`sex` int NULL DEFAULT NULL,

`birthday` varchar(50) CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci NULL DEFAULT NULL,

PRIMARY KEY (`id`) USING BTREE

) ENGINE = InnoDB AUTO\_INCREMENT = 7 CHARACTER SET = utf8mb4 COLLATE = utf8mb4\_0900\_ai\_ci ROW\_FORMAT = Dynamic;

### 1.2.5 定义Controller、Mapper、Entity

电脑萤幕的截图

描述已自动生成

电脑屏幕的截图

描述已自动生成

文本

描述已自动生成

### 1.2.6 验证

一主一从数据库初始化：

图形用户界面

中度可信度描述已自动生成

插入：主库 ds1

图形用户界面

描述已自动生成

查询：从库 ds2

文本

中度可信度描述已自动生成

强制查询主库：

文本

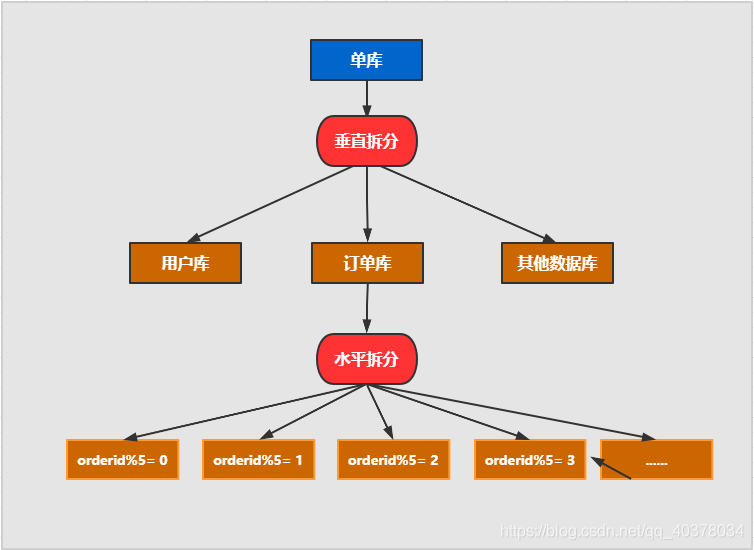
描述已自动生成

## 2.1 分库分表

### 2.1.1MySQL分库分表原理

水平拆分：同一个表的数据拆到不同的库不同的表中。可以根据时间、地区或某个业务键维度，也可以通过hash进行拆分，最后通过路由访问到具体的数据。拆分后的每个表结构保持一致

垂直拆分：就是把一个有很多字段的表给拆分成多个表，或者是多个库上去。每个库表的结构都不一样，每个库表都包含部分字段。一般来说，可以根据业务维度进行拆分，如订单表可以拆分为订单、订单支持、订单地址、订单商品、订单扩展等表；也可以，根据数据冷热程度拆分，20%的热点字段拆到一个表，80%的冷字段拆到另外一个表



考虑问题：怎么迁移数据

## 2.1.2分库分表实现

### ****2.1.3需求****：

对1000w的用户数据进行分库分表，对用户表的数据进行分表和分库的操作。根据年龄奇数存储在t\_user\_t1，偶数t\_user\_t0，同时性别奇数存储在ds1，偶数ds0

### 2.1.4逻辑表 t\_user\_t

CREATE TABLE `t\_user\_t0` (

`id` bigint(20) DEFAULT NULL,

`nickname` varchar(200) DEFAULT NULL,

`password` varchar(200) DEFAULT NULL,

`age` int(11) DEFAULT NULL,

`sex` int(11) DEFAULT NULL,

`birthday` varchar(100) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

CREATE TABLE `t\_user\_t1` (

`id` bigint(20) DEFAULT NULL,

`nickname` varchar(200) DEFAULT NULL,

`password` varchar(200) DEFAULT NULL,

`age` int(11) DEFAULT NULL,

`sex` int(11) DEFAULT NULL,

`birthday` varchar(100) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

图形用户界面, 应用程序

描述已自动生成

### 2.1.5 application.properties

#执行日志打印

mybatis-plus.configuration.log-impl=org.apache.ibatis.logging.stdout.StdOutImpl

##### sharding的分库分表

spring.main.allow-bean-definition-overriding=true

#显示sql

spring.shardingsphere.props.sql.show=true

#配置数据源

spring.shardingsphere.datasource.names=ds0,ds1

#ds0数据库连接信息

spring.shardingsphere.datasource.ds0.type=com.alibaba.druid.pool.DruidDataSource

spring.shardingsphere.datasource.ds0.driver-class-name=com.mysql.cj.jdbc.Driver

spring.shardingsphere.datasource.ds0.url=jdbc:mysql://10.207.0.161:3306/testshardingjdbc?useUnicode=true&characterEncoding=utf-8&serverTimezone=Asia/Shanghai

spring.shardingsphere.datasource.ds0.username=root

spring.shardingsphere.datasource.ds0.password=123456

spring.shardingsphere.datasource.ds0.maxPoolSize=100

spring.shardingsphere.datasource.ds0.minPoolSize=5

#ds1数据库连接信息

spring.shardingsphere.datasource.ds1.type=com.alibaba.druid.pool.DruidDataSource

spring.shardingsphere.datasource.ds1.driver-class-name=com.mysql.cj.jdbc.Driver

spring.shardingsphere.datasource.ds1.url=jdbc:mysql://127.0.0.1:3306/testshardingjdbc?useUnicode=true&characterEncoding=utf-8&serverTimezone=Asia/Shanghai

spring.shardingsphere.datasource.ds1.username=root

spring.shardingsphere.datasource.ds1.password=123456

spring.shardingsphere.datasource.ds1.maxPoolSize=100

spring.shardingsphere.datasource.ds1.minPoolSize=5

spring.shardingsphere.sharding.tables.t\_user\_t.actual-data-nodes=ds$->{0..1}.t\_user\_t$->{0..1}

#数据源分片策略

spring.shardingsphere.sharding.tables.t\_user\_t.database-strategy.inline.sharding-column=sex

#数据源分片算法

spring.shardingsphere.sharding.tables.t\_user\_t.database-strategy.inline.algorithm-expression=ds$->{sex%2}

#表分片策略

spring.shardingsphere.sharding.tables.t\_user\_t.table-strategy.inline.sharding-column=age

#表分片算法

spring.shardingsphere.sharding.tables.t\_user\_t.table-strategy.inline.algorithm-expression=t\_user\_t$->{age%2}

#主键的列名

spring.shardingsphere.sharding.tables.t\_user\_t.key-generator.column=id

spring.shardingsphere.sharding.tables.t\_user\_t.key-generator.type=SNOWFLAKE

### 2.1.6插入、查询验证：

插入年龄、性别都是奇数：

电脑萤幕的截图

描述已自动生成

文本

描述已自动生成

插入年龄偶数、性别奇数：

文本

描述已自动生成

文本

描述已自动生成

插入年龄奇数、性别偶数：

文本

描述已自动生成

图片包含 文本

描述已自动生成

插入年龄偶数、性别偶数：

文本

描述已自动生成

图片包含 图表

描述已自动生成

查询验证：

文本

描述已自动生成

1.分库的分片键性别有、分表的分片键年龄都有：路由到ds0库的t\_user\_t0 表

文本

描述已自动生成

2.分库的分片键性别有、分表的分片键年龄没有：路由到ds0库的t\_user\_t0、t\_user\_t1表

文本

描述已自动生成

3.分库的分片键性别没有、分表的分片键年龄有：路由到ds0、ds1库的t\_user\_t0表

文本

描述已自动生成

4.分库的分片键性别没有、分表的分片键年龄没有：全局路由

图形用户界面, 文本

中度可信度描述已自动生成