# Sharding-jdbc之-分库分表初体验

什么是分库分表

数据分片sharding

为什么需要分库分表

随着我们业务越来越大，单机单个应用瓶颈问题。

数据库持久化硬盘如何去扩容

数据库瓶颈解决方案：

1. sql、表优化
2. 读写分离
3. 分库分表

# 读写分离：

读取（查询）、修改、插入、删除CRUD

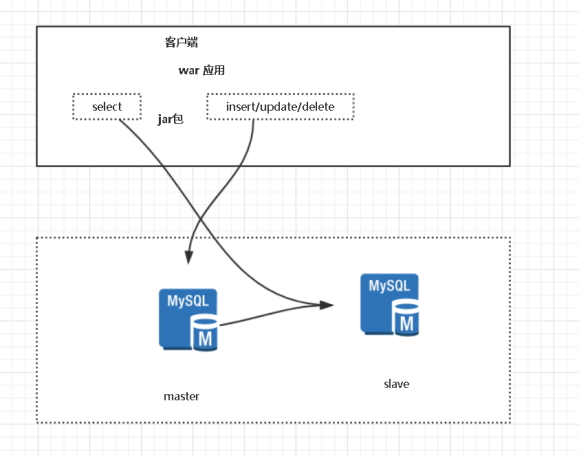
数据库角色：master（主库、写库） slave（从库 读库）

读写分离：insert update delete走主库，然后select走从库。

应用：读多写少 从库压力会很大 主库相对来说会低（双十一 写）

**读写分离java开源框架分类：**

**客户端（应用层）：TDDL、 Sharding-Jdbc**



**特点：**

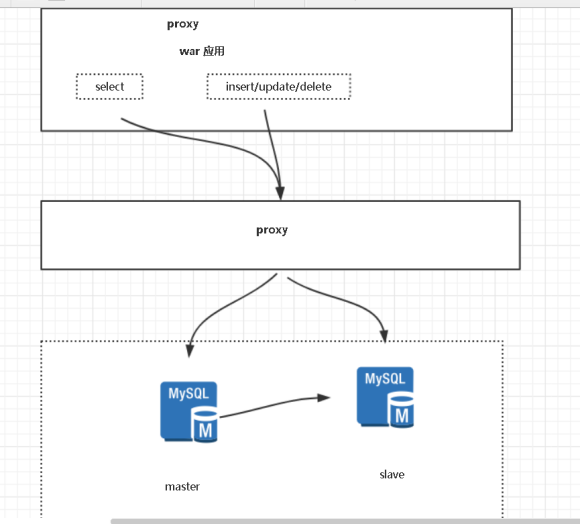
**优点：1、程序自动完成，数据源方便管理**

1. **不需要维护，因为没用中间件**
2. **理论支持任何数据库 （sql标准）**

**缺点：**

1. **增加了开发成本、代码有入侵**
2. **不能做到动态增加数据源**
3. **程序员开发完成，运维参与不了。**

**中间件（代理层proxy）：mysql proxy、mycat、altas**



**特点：**

**优点：1、数据增加了都程序没用任何影响**

**2、应用层（程序）不需要管数据库方面的事情**

**3、增加数据源不需要重启程序**

**缺点：**

1. 程序依赖中间件，导致切换数据库变的困难
2. 增加了proxy 性能下降
3. 增加了维护工作、高可用问题。

分库分表应用层：sharding-jdbc。

proxy层：mycat

# 分库分表：

分库分表其实基于我们读写分离上面提出的方案（也就是目前关系型数据库的终极解决方案）。

读写分离：master一个压力的问题 随着我们业务增大之后 过渡期 瓶颈 产生原因，解决

数据分片概念。

分库分表：目前数据库终极解决方案 解决高并发、数据分片。

## 分库（表）类型：

**垂直：**

将一个字段（属性）比较多的表拆分成多个小表，将不同字段放到不同的表中

降低单（表）库大小的目的来提高性能。

通俗：大表拆小表，拆分是基于关系型数据库的列（字段）来进行

特点：1、每个库（表）的结构都不一样

1. 每个库（表）数据都（至少有一列）一样
2. 每个库（表）的并集是整个数据库的全量数据。
3. 每个库（表）的数据量（count）不会变的。

user-info + user-base=user（字段）

**水平**：

某个字段按一定规律进行拆分，将一个表的数据分到多个表（库）中

降低表的数据量，优化查询数据量的方式，来提高性能。

user1+user2=user（数据量）

特点： 每个库（表）的结构都一样

每个库（表）的数据都不一样

每个库（表）的并集是整个数据库的全量数据

**分库分表常见算法 ：**

**Hash取模：通过表的一列字段进行hash取出code值来区分的**

**存在数据扩容迁移问题**

**Range范围: 按年份、按时间**

**List预定义：事先设定100找**

**分库分表之后带来的问题：**

1、查询数据结果集合并

2、sql的改变

3、分布式事务

4、全局唯一性id

如何设计一个永远都不需要迁移的方案？