# 第五节课交易分库分表详解一

## 数据库的压力：

订单相关表都已经是超大表，最大表的数据量已经是几十亿，数据库处理能力已经到了极限；  
单库包含多个超大表，占用的硬盘空间已经接近了服务器的硬盘极限，很快将无空间可用；

过度解决：我们可以考虑到最直接的方式是增加大容量硬盘，或者对IO有更高要求，还可以考虑增加SSD硬盘来解决容量的问题。此方法无法解决单表数据量问题。

可以对数据表历史数据进行归档，但也需要频繁进行归档操作，而且不能解决性能问题

硬件上（大小）、单表容量（性能）

## 如何优化？有什么方式：

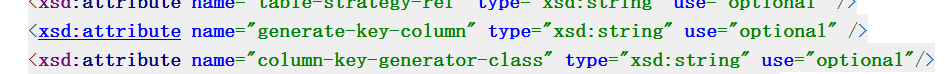
读写分离、换mysql》oracle、分库分表。

分库分表：

1. 散列hash：hashmap可以很好的去解决数据热点的问题，但是扩容 短板
2. Range增量：他的库容很多好，但是他就是没法解决数据热点的问题。

## 实战：

老的版本：不支持spring管理（分布式主键）

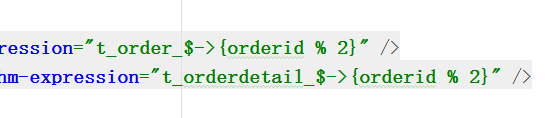


新的版本:支持（分布式主键）



## 问题：

表增加是可以通过创建，但是有一个地方我们改了也会有问题



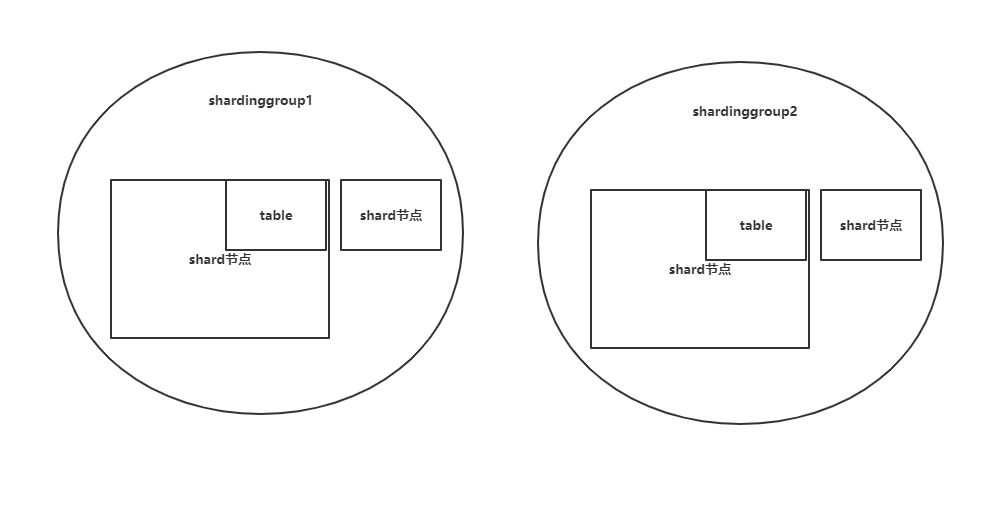
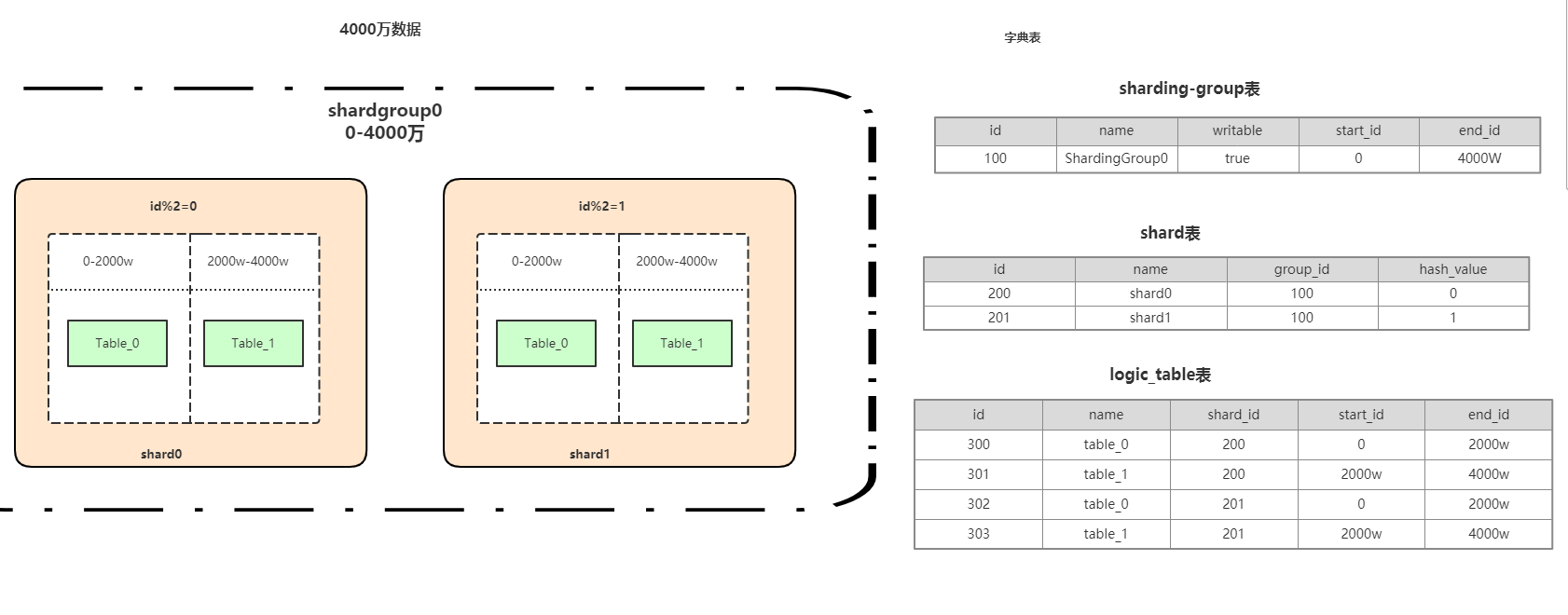
**面对这两种方案都是比较难的处境，shard扩容显得难了、**

扩容的问题？

**分库扩容理想状态：**

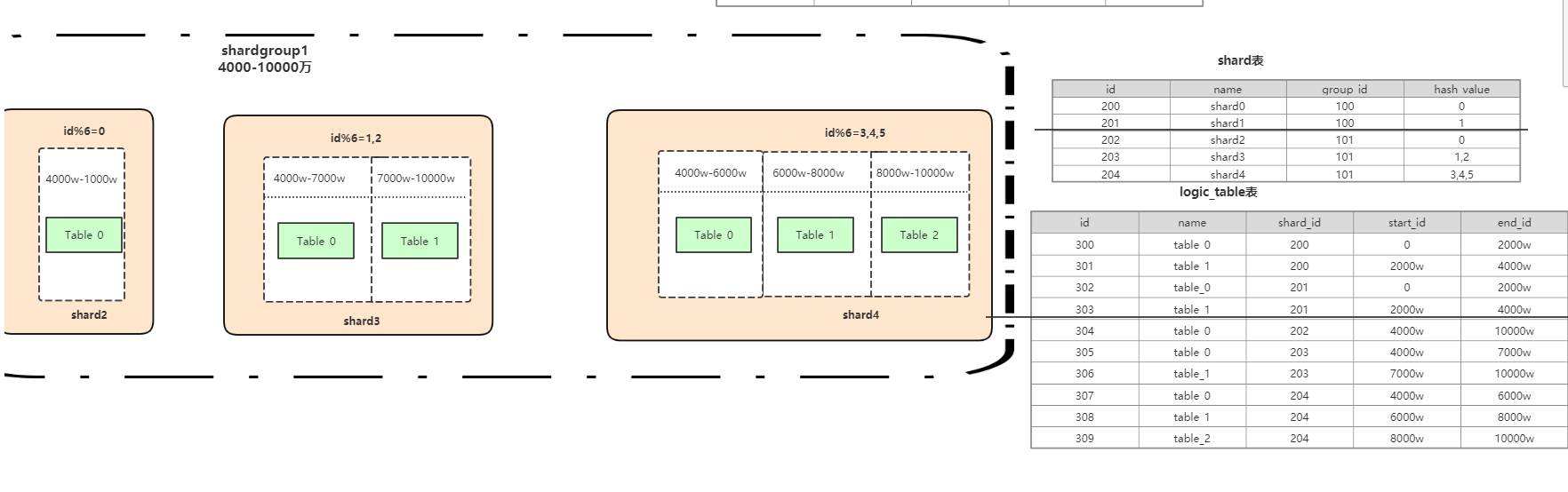
1. 最好不数据迁移（给团队带来的工作压力）
2. 可以达到1的要求，并且数据热点的问题。
3. 根据硬件资源设置不同阈值（判定）

## 如何优化： 散列+增量=活好人帅（数据热点+扩容）

热点：解决数据热点的问题（因为我们局部用了散列）

扩容：



## 总结：

1. 多查一次数据库（字典表）
2. 依赖全局的ID生成（美团+业务ID在区间自增）

**其他：**

前端：吞吐量高、并发高、相应速度快、但是业务简单（我）

我的订单

全部订单（我） userid=1

商品表（shopid）

后端：(并发不高、业务逻辑复杂、相应不快)

后台（另外）业务复杂、吞吐量不高