# mysql的分库分表

关系型数据库本身比较容易成为系统瓶颈，单机存储容量、连接数、处理能力都有限。当单表的数据量达到1000W或100G以后，由于查询维度较多，即使添加从库、优化索引，做很多操作时性能仍下降严重。此时就要考虑对其进行切分了，切分的目的就在于减少数据库的负担，缩短查询时间。

数据切分根据其切分类型，可以分为两种方式：垂直（纵向）切分和水平（横向）切分。

## 1.垂直切分

垂直切分常见有垂直分库和垂直分表两种。

垂直分库就是根据业务耦合性，将关联度低的不同表存储在不同的数据库。

垂直分表是基于数据库中的"列"进行，某个表字段较多，可以新建一张扩展表，将不经常用或字段长度较大的字段拆分出去到扩展表中。在字段很多的情况下（例如一个大表有100多个字段），通过"大表拆小表"，更便于开发与维护，也能避免跨页问题，MySQL底层是通过数据页存储的，一条记录占用空间过大会导致跨页，造成额外的性能开销。另外数据库以行为单位将数据加载到内存中，这样表中字段长度较短且访问频率较高，内存能加载更多的数据，命中率更高，减少了磁盘IO，从而提升了数据库性能。

垂直切分的优点：

* 解决业务系统层面的耦合，业务清晰
* 与微服务的治理类似，也能对不同业务的数据进行分级管理、维护、监控、扩展等
* 高并发场景下，垂直切分一定程度的提升IO、数据库连接数、单机硬件资源的瓶颈

缺点：

* 部分表无法join，只能通过接口聚合方式解决，提升了开发的复杂度
* 分布式事务处理复杂
* 依然存在单表数据量过大的问题（需要水平切分）

## 2.水平切分

当一个应用难以再细粒度的垂直切分，或切分后数据量行数巨大，存在单库读写、存储性能瓶颈，这时候就需要进行水平切分了。

水平切分分为**库内分表和分库分表**，是根据表内数据内在的逻辑关系，将同一个表按不同的条件分散到多个数据库或多个表中，每个表中只包含一部分数据，从而使得单个表的数据量变小，达到分布式的效果。

水平切分的优点：

* 不存在单库数据量过大、高并发的性能瓶颈，提升系统稳定性和负载能力
* 应用端改造较小，不需要拆分业务模块

缺点：

* 跨分片的事务一致性难以保证
* 跨库的join关联查询性能较差
* 数据多次扩展难度和维护量极大

几种典型的数据分片规则为：

按照用户 ID 求模，将数据分散到不同的数据库；

按照日期，将不同月甚至日的数据分散到不同的库中；

按照某个特定的字段求摸，或者根据特定范围段分散到不同的库中。

## 3.分库分表带来的问题

### 1、事务一致性问题

分布式事务

分布式事务能最大限度保证了数据库操作的原子性。但在提交事务时需要协调多个节点，推后了提交事务的时间点，延长了事务的执行时间。导致事务在访问共享资源时发生冲突或死锁的概率增高。随着数据库节点的增多，这种趋势会越来越严重，从而成为系统在数据库层面上水平扩展的枷锁。

最终一致性

对于那些性能要求很高，但对一致性要求不高的系统，往往不苛求系统的实时一致性，只要在允许的时间段内达到最终一致性即可，可采用事务补偿的方式。与事务在执行中发生错误后立即回滚的方式不同，事务补偿是一种事后检查补救的措施，一些常见的实现方法有：对数据进行对账检查，基于日志进行对比，定期同标准数据来源进行同步等等。事务补偿还要结合业务系统来考虑。

### 2、跨节点关联查询 join 问题

只要是进行切分，跨节点Join的问题是不可避免的。但是良好的设计和切分却可以减少此类情况的发生。解决这一问题的普遍做法是分两次查询实现。在第一次查询的结果集中找出关联数据的id,根据这些id发起第二次请求得到关联数据。

### 3.跨节点的count,order by,group by以及聚合函数问题

这些是一类问题，因为它们都需要基于全部数据集合进行计算。多数的代理都不会自动处理合并工作。解决方案：与解决跨节点join问题的类似，分别在各个节点上得到结果后在应用程序端进行合并。和join不同的是每个结点的查询可以并行执行，因此很多时候它的速度要比单一大表快很多。但如果结果集很大，对应用程序内存的消耗是一个问题。

### 4.数据迁移，容量规划，扩容等问题

如果采用数值范围分片，只需要添加节点就可以进行扩容了，不需要对分片数据迁移。如果采用的是数值取模分片，则考虑后期的扩容问题就相对比较麻烦。

### 5. ID问题

一旦数据库被切分到多个物理结点上，我们将不能再依赖数据库自身的主键生成机制。一方面，某个分区数据库自生成的ID无法保证在全局上是唯一的；另一方面，应用程序在插入数据之前需要先获得ID,以便进行SQL路由.

[Twitter的分布式自增ID算法Snowflake](http://blog.sina.com.cn/s/blog_6b7c2e660102vbi2.html)。

### 6. 跨分片的排序分页

一般来讲，分页时需要按照指定字段进行排序。当排序字段就是分片字段的时候，我们通过分片规则可以比较容易定位到指定的分片，而当排序字段非分片字段的时候，情况就会变得比较复杂了。为了最终结果的准确性，我们需要在不同的分片节点中将数据进行排序并返回，并将不同分片返回的结果集进行汇总和再次排序，最后再返回给用户。

## 4.什么时候考虑切分

1.能不切分尽量不要切分；

2. 数据量过大，正常运维影响业务访问；

3.随着业务发展，需要对某些字段垂直拆分；

4. 数据量快速增长；

5、安全性和可用性。