分库、分表和分区

单表行数超过500万行或者单表容量超过 2GB，才推荐进行分库分表

分表与分区的区别

1. 数据处理上  
   a），分表后，数据都是存放在分表里，总表只是一个外壳，存取数据发生在一个一个的分表里面。select \* from alluser where id='12'表面上看，是对表alluser进行操作的，其实不是的。是对alluser里面的分表进行了操作。  
   b），分区呢，不存在分表的概念，分区只不过把存放数据的文件分成了许多小块，分区后的表呢，还是一张表。数据处理还是由自己来完成。   
   2，提高性能上   
   a）， 分表后，单表的并发能力提高了，磁盘I/O性能也提高了。并发能力为什么提高了呢，因为查寻一次所花的时间变短了，如果出现高并发的话，总表可以根据不同 的查询，将并发压力分到不同的小表里面。磁盘I/O性能怎么搞高了呢，本来一个非常大的.MYD文件现在也分摊到各个小表的.MYD中去了。   
   b），mysql提出了分区的概念，我觉得就想突破磁盘I/O瓶颈，想提高磁盘的读写能力，来增加mysql性能。   
   在这一点上，分区和分表的测重点不同，分表重点是存取数据时，如何提高mysql并发能力上；而分区呢，如何突破磁盘的读写能力，从而达到提高mysql性能的目的。   
   4），实现的难易度上   
   a），分表的方法有很多，用merge来分表，是最简单的一种方式。这种方式根分区难易度差不多，并且对程序代码来说可以做到透明的。如果是用其他分表方式就比分区麻烦了。   
   b），分区实现是比较简单的，建立分区表，根建平常的表没什么区别，并且对开代码端来说是透明的。   
     
   mysql分表和分区联系呢  
   1，都能提高mysql的性高，在高并发状态下都有一个良好的表面。   
   2，分表和分区不矛盾，可以相互配合的，对于那些大访问量，并且表数据比较多的表，我们可以采取分表和分区结合的方式（如果merge这种分表方式，不能和分区配合的话，可以用其他的分表试），访问量不大，但是表数据很多的表，我们可以采取分区的方式等。

分库分表区别：

1 基本思想之什么是分库分表？  
从字面上简单理解，就是把原本存储于一个库的数据分块存储到多个库上，把原本存储于一个表的数据分块存储到多个表上。  
2 基本思想之为什么要分库分表？  
     数据库中的数据量不一定是可控的，在未进行分库分表的情况下，随着时间和业务的发展，库中的表会越来越多，表中的数据量也会越来越大，相应地，数据操作，增删改查的开销也会越来越大；另外，一台服务器的资源（CPU、磁盘、内存、IO等）是有限的，最终数据库所能承载的数据量、数据处理能力都将遭遇瓶颈,。  
3 分库分表的实施策略。  
     如果你的单机性能很低了，那可以尝试分库。分库，业务透明，在物理实现上分成多个服务器，不同的分库在不同服务器上。分区可以把表分到不同的硬盘上，但不能分配到不同服务器上。一台机器的性能是有限制的，用分库可以解决单台服务器性能不够，或者成本过高问题。  
当分区之后，表还是很大，处理不过来，这时候可以用分库。  
orderid,userid,ordertime,.....  
userid%4=0，用分库1  
userid%4=1，用分库2  
userid%4=2, 用分库3  
userid%4=3，用分库4  
上面这个就是一个简单的分库路由，根据userid选择分库，即不同的服务器  
分库分表有垂直切分和水平切分两种。  
     3.1 何谓垂直切分，即将表按照功能模块、关系密切程度划分出来，部署到不同的库上。例如，我们会建立定义数据库workDB、商品数据库payDB、用户数据库userDB、日志数据库logDB等，分别用于存储项目数据定义表、商品定义表、用户数据表、日志数据表等。

如userid,name,addr一个表，为了防止表过大，分成2个表。

userid,name

userid,addr

       3.2 何谓水平切分，当一个表中的数据量过大时，我们可以把该表的数据按照某种规则，例如userID散列、按性别、按省，进行划分，然后存储到多个结构相同的表，和不同的库上。例如，我们的userDB中的用户数据表中，每一个表的数据量都很大，就可以把userDB切分为结构相同的多个userDB：part0DB、part1DB等，再将userDB上的用户数据表userTable，切分为很多userTable：userTable0、userTable1等，然后将这些表按照一定的规则存储到多个userDB上。  
      3.3 应该使用哪一种方式来实施数据库分库分表，这要看数据库中数据量的瓶颈所在，并综合项目的业务类型进行考虑。  
如果数据库是因为表太多而造成海量数据，并且项目的各项业务逻辑划分清晰、低耦合，那么规则简单明了、容易实施的垂直切分必是首选。  
而如果数据库中的表并不多，但单表的数据量很大、或数据热度很高，这种情况之下就应该选择水平切分，水平切分比垂直切分要复杂一些，它将原本逻辑上属于一体的数据进行了物理分割，除了在分割时要对分割的粒度做好评估，考虑数据平均和负载平均，后期也将对项目人员及应用程序产生额外的数据管理负担。  
在现实项目中，往往是这两种情况兼而有之，这就需要做出权衡，甚至既需要垂直切分，又需要水平切分。我们的游戏项目便综合使用了垂直与水平切分，我们首先对数据库进行垂直切分，然后，再针对一部分表，通常是用户数据表，进行水平切分。

4 分库分表存在的问题。  
  
4.1 事务问题。  
 在执行分库分表之后，由于数据存储到了不同的库上，数据库事务管理出现了困难。如果依赖数据库本身的分布式事务管理功能去执行事务，将付出高昂的性能代价；如果由应用程序去协助控制，形成程序逻辑上的事务，又会造成编程方面的负担。

4.2 跨库跨表的join问题。  
 在执行了分库分表之后，难以避免会将原本逻辑关联性很强的数据划分到不同的表、不同的库上，这时，表的关联操作将受到限制，我们无法join位于不同分库的表，也无法join分表粒度不同的表，结果原本一次查询能够完成的业务，可能需要多次查询才能完成。

4.3 额外的数据管理负担和数据运算压力。  
额外的数据管理负担，最显而易见的就是数据的定位问题和数据的增删改查的重复执行问题，这些都可以通过应用程序解决，但必然引起额外的逻辑运算，例如，对于一个记录用户成绩的用户数据表userTable，业务要求查出成绩最好的100位，在进行分表之前，只需一个order by语句就可以搞定，但是在进行分表之后，将需要n个order by语句，分别查出每一个分表的前100名用户数据，然后再对这些数据进行合并计算，才能得出结果。