Atitit mysql分区分表分库方法总结

目录

[1. mysql分表和分区有什么联系呢 2](#_Toc15507)

[2. 分表方法 2](#_Toc6931)

[2.1. share jdbc 2](#_Toc28248)

[2.2. mysql集群启到了分表的作用， 读写分离 2](#_Toc1706)

[2.3. Mysql分区 3](#_Toc11266)

[2.4. nosql也是一个替代方案. 3](#_Toc17159)

[2.5. mysql+nosql 读写操作mysql, 分页查询走ES等等. 4](#_Toc13057)

[2.6. 使用场景Merge表有点类似于 4](#_Toc9573)

[2.7. 视图 union+单表写入 4](#_Toc8943)

[2.8. Sp 4](#_Toc6431)

[3. 分表后的问题 join 4](#_Toc25432)

[3.1. 同库分表使用view union all 4](#_Toc21183)

[3.2. JoinRowSet接口： 4](#_Toc23815)

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。**

1. ，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分表和分区有什么联系呢

**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。

# 分表方法

## share jdbc

## mysql集群启到了分表的作用， 读写分离

，做mysql集群，例如：利用mysql cluster ，mysql proxy，mysql replication，drdb等等

有人会问mysql集群，根分表有什么关系吗？虽然它不是实际意义上的分表，但是它启到了分表的作用，做集群的意义是什么呢？为一个数据库减轻负担，说白了就是减少sql排队队列中的sql的数量，举个例子：有10个sql请求，如果放在一个数据库服务器的排队队列中，他要等很长时间，如果把这10个sql请求，分配到5个数据库服务器的排队队列

## Mysql分区

基本来说, 分区和分表带来的性能提升是一样的, 由于分区实际上就可以认为是mysql底层来帮我们实现分表的逻辑了, 所以相对来说分表会比分区带来更高的编码复杂度(分区就根本不用考虑多表分页查询的问题了). 从这个角度来说, 一般的业务直接分区就可以了.

当然, 选择分区还是分表还是需要做一点权衡的:

1. 表中的数据只有部分热点数据经常访问, 其他的不常访问的话, 适合用分区表

2. 分区表相对容易维护, 可以针对单独一个分区进行检查,优化, 批量删除大量数据时, 分区表会比一般的表更快

3. 分区表可以分布在不同的物理设备上, 从而可以高效地利用多个硬盘

4. 如果查询条件不包含partition key的话, 分区表不一定有分表效率高

5. 如果分区表中绝对的热点数据, 每一条数据都有可能被访问到, 也不太适合分区

6. 如果数据量超大, 由于mysql只能分1024个分区, 如果1024个分区的数据都是千万以上, 那肯定是也不适合分区的了复制代码

综上所述, 如果分区表就足够满足我们的话, 那其实就没有必要进行分表了增加编程的复杂度了.

另外, 如果不想将数据表进行拆分, 而表的数据量又的确很大的话,

## nosql也是一个替代方案.

特别是那些不需要强事务的表操作, 就很适合放在nosql, 从而可以避免编程的复杂度, 同时性能上也没有过多的损耗.

nosql的方案也有很多:

1. mongoDb

2. hbase

3. tidb

4. elasticSearch 复制代码

当然也可以使用

## mysql+nosql 读写操作mysql, 分页查询走ES等等.

Mysql从库canal复制到es

## 使用场景Merge表有点类似于

。使用Merge存储引擎实现MySQL分表，这种方法比较适合那些没有事先考虑分表，随着数据的增多，已经 .

## 视图 union+单表写入

## Sp

# 分表后的问题 join

## 同库分表使用view union all

## ****JoinRowSet接口：****

　　这个接口可以提供我们在无连接的状态下直接对结果集进行Join。下面的代码提供了JoinRowSet的实现：

CachedRowSet crs1=new CaehedRowSetImpl();  
　　crs1.setUrl(“jdbc:mydql://localhost:3306/test”);  
　　crs1.setUsername(“root”);  
　　crs1.setPassword(“”);  
　　crs1.setCommand(“select \* from table1”);  
　　crs1.execute();  
　　CachedRowSet crs2=new CaehedRowSetImpl();  
　　crs2.setUrl(“jdbc:mydql://localhost:3306/test”);  
　　crs2.setUsername(“root”);  
　　crs2.setPassword(“”);  
　　crs2.setCommand(“select \* from table2”);  
　　crs2.execute();  
　　JoinRowSet jrs=new JoinRowSetImpl();  
　　jrs.addRowSet(crs1,”id”);  
　　jrs.addRowSet(crs2,”id”);  
　　while(jrs.next())  
　　System.out.println(jrs.getInt(“id”)+”/t/t”+jrs.getString(“name”)+”/t/t”+jrs.getString(“info”);

　　这段代码的作用和执行select \* from table1 inner join table2 on table1.id=table2.id语句得到的结果集是一样的。但是我个人认为与其这样复杂地使用JoinRowSet，不如直接使用这条Join语句来得到CachedRowSet。  
  
　　默认的Join是inner join的，接口还支持cross join,full join,left outer join和right outer join，我们通过setJoinType()方法来修改连接类型，当然这还是需要数据库的支持。还有一个值得注意的地方就是，在这个例子里我连接的列在两个表里面都叫id，那么我们取数据的时候就使用id这个名字，那如果两列的名字不一样呢？系统就会为这个连接列取一个默认的名字叫做”MergedCol”。