数据库优化篇

1. SQL优化（索引、表分析和排序）

（1）优化SQL语句一般步骤：

A）通过show status命令了解各种sql的执行频率

B）定位执行效率较低的sql语句

通过慢查询日志定位那些执行效率较低的sql语句

C）通过EXPLAIN分析低效的SQL的执行计划

D）确定问题并采取相应的优化措施

（2）索引问题（索引是数据库优化中最常用也是最重要的手段之一）

可能使用索引的情况（记住：最左前缀匹配原则）

A）对于多列索引，只要查询条件中用到了最左边的列，索引一般就会使用

B）对于使用like的查询，不以%开头才会使用索引

C）如果对大的文本进行搜索，使用全文索引而不用使用like ‘%...%’

D）如果列名是索引，使用column\_name is null将使用索引

存在索引但不使用索引

A）如果mysql估计使用索引比全表扫描更慢，则不使用索引

B）用or分割开的条件

C）如果不是索引列的第一部分

D）如果like是以%开始

E）如果列类型是字符串，那么一定记得在where 条件中把字符常量值用引号引起来，否则的话即便这个列上有索引，MySQL 也不会用到的

（3）常用优化

大批量插入数据

因为InnoDB 类型的表是按照主键的顺序保存的，所以将导入的数据按照主键的顺序排列，可以有效地提高导入数据的效率

优化insert语句

如果同时从同一客户插入很多行，尽量使用多个值表的INSERT 语句

如果从不同客户插入很多行，能通过使用INSERT DELAYED 语句得到更高的速度

优化GROUP BY语句（排序问题引起）

如果查询包括GROUP BY 但用户想要避免排序结果的消耗，则可以指定ORDER BY NULL禁止排序

优化ORDER BY语句（排序与索引）

MySQL 可以使用一个索引来满足ORDER BY 子句，而不需要额外的排序。WHERE 条件和ORDER BY 使用相同的索引，并且ORDER BY 的顺序和索引顺序相同，并且ORDER BY 的字段都是升序或者都是降序

优化嵌套查询（表结构）

子查询可以被更有效率的连接（JOIN）替代

连接（JOIN）之所以更有效率一些，是因为MySQL 不需要在内存中创建临时表来完成这个逻辑上的需要两个步骤的查询工作

优化OR条件

对于含有OR 的查询子句，如果要利用索引，则OR 之间的每个条件列都必须用到索引；如果没有索引，则应该考虑增加索引

2.优化数据库对象

（1）优化表的数据类型

表需要使用何种数据类型，是需要根据应用来判断的。虽然应用设计的时候需要考虑字段的长度留有一定的冗余，但是不推荐让很多字段都留有大量的冗余，这样即浪费磁盘存储空间,同时在应用程序操作时也浪费物理内存。

（2）分表和分库

垂直拆分：即把主码和一些列放到一个表，然后把主码和另外的列放到另一个表中。

如果一个表中某些列常用，而另外一些列不常用，则可以采用垂直拆分，另外垂直拆分可以使得数据行变小，一个数据页就能存放更多的数据，在查询时就会减少I/O 次数。

其缺点是需要管理冗余列，查询所有数据需要联合（JOIN）操作

水平拆分：即根据一列或多列数据的值把数据行放到两个独立的表中

表很大，降低索引层数，降低了查询是需要读数据和索引的页数

表中的数据本来就有独立性

需要把数据存在多个介质上

（3）逆范式化

增加冗余列、增加派生列、重新组表、分割表

因为规范化越高，那么产生的关系就越多，关系过多的直接结果就是导致表之间的连接操作越频繁，而表之间的连接操作是性能较低的操作，直接影响到查询的速度，所以对于查询较多的应用就需要根据实际情况运用逆规范化对数据进行设计，通过逆规范化来提高查询的性能

（4）使用中间表提高统计查询速度

中间表复制源表部分数据，并且与源表相“隔离”，在中间表上做统计查询不会对在线应用产生负面影响

 中间表上可以灵活的添加索引或增加临时用的新字段,从而达到提高统计查询效率和辅助统计查询作

3.锁问题