汇总表 砍基本表

MySQL服务器的安全基础是：用户应该对他们需要的数据具有适当的访问权，既不能多也不能少。换句话说，用户不能对过多的数据具有过多的访问权。

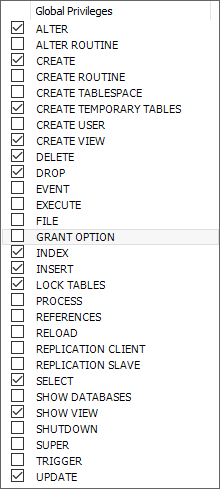


MySQL用户账号和信息存储在名为mysql的MySQL数据库中。一般不需要直接访问mysql数据库和表，但有时需要直接访问。需要直接访问它的时机之一是在需要获得所有用户账号列表时。

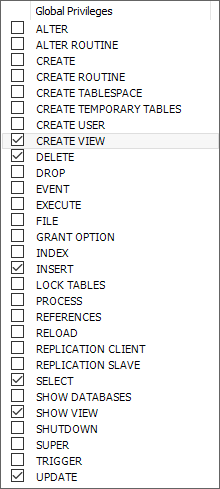
可以在mySQL中建立一个user表，包含所有账户，存储其登录名和密码

设置三类账户，教务端账户为1型，老师端账户为2型，家长端账户为3型

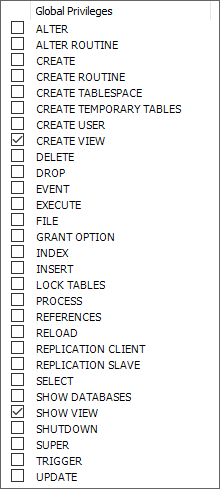
其中1型账户：



2型账户：



3型用户



数据库查询优化：参考https://blog.csdn.net/qq\_42022528/article/details/85131447

1、为什么查询速度会慢

通常来说，查询的生命周期大致可以分为以下顺序：从客户端，到服务器，然后在服务器上进行解析，优化后生成执行计划，执行，并返回结果给客户端。其中执行可以认为是整个生命周期最重要的阶段，这其中包含了大量为了检索数据到存储引擎的调用，以及调用后的数据处理，包括排序和分组。

在完成这些任务的时候，查询需要在不同的地方花费时间,包括网络、CPU计算，生成统计信息和执行计划、锁等待等操作，尤其是向底层存储引擎检索数据的调用操作。优化和查询的目的就是减少和消除这些操作所花费的时间。

查询性能低下的最基本原因是访问的数据太多，大部分性能低下的查询可以通过减少访问的数据量的方式进行优化：

确认程序是否在检索大量超过需要的数据，这通常意味着访问太多行，但有时也可能是访问太多列。

确认MySQL服务器层是否在分析大量超过需求的数据。

2.1 是否向数据库请求了不需要的数据

（1）查询不需要的记录

例如查询select查询大量的结果，然后获取前面N条结果后关闭结果集。分页查询使用逻辑分页也是一样，查询所有数据，只返给页面所需N条记录。最简单有效的方法就是在这样的查询中使用Limit。

（2）多表关联时返回全部列

“select \* from..”总是取出全部列，会让优化器无法完成索引覆盖扫描这列优化，还会为服务器带来额外的I\O，内存和CPU消耗。因此一些DBA严格禁止SELECT \* 的写法。但是在许多实际开发中，查询返回超过需要的数据也不总是坏事，因为这种有点浪费数据库资源的方式可以简化开发，提高相同代码片段的复用性（一组数据结果可供多个接口使用）。获取并缓存所有列的查询，相比多个独立的只获取部分列的查询可能更有好处。

（3）重复查询相同的数据。

在程序中很容易出现这样的逻辑错误——不断执行相同的查询，并返回相同数据。例如，博客中用户评论功能中，需要查询用户头像，那么用户多次评论的时候，可能会反复查询头像数据。比较好的方案是，在初次查询的时候就将头像数据缓存起来。

一般MySQL可以用如下三种方式应用where条件，从好到坏依次是：

1.**在索引中使用where条件来过滤不匹配的记录，这是在存储引擎完成的**。

2.**使用索引覆盖扫描（在extra列中出现了Using Index）来返回记录，直接从索引中过滤不需要的记录并返回命中的结果。这是在MySQL服务器层完成，但无需再回表查询记录**。

3.**从数据表中返回数据，然后过滤不满足条件的记录（在extra列中出现了Using Where）,这在MySQL服务器层完成，MySQL需要先从数据表中读出记录然后过滤**。

所以好的索引可以让查询使用适合的访问类型，尽可能地只扫描需要的数据行。如果发现查询需要扫描大量的数据但只返回少数的行，那么通常可以尝试以下的方式去优化它：

1.使用索引覆盖扫描，把所有需要用的列都放在索引中（比如多字段的聚合索引），这样存储引擎无须回表获取对应的行就能返回结果。

2.改变库表结构，例如使用单独的汇总表。

3.重写这个复杂的查询，让MySQL优化器能够以更优化的方式执行这个查询。

参考：<https://blog.csdn.net/yangyuge1987/article/details/79401545>

范式和反范式

在初学数据库设计时，我们往往要遵循数据库的范式要求，尤其是前三个范式。严格遵循范式设计的表通常更小、数据冗余少，做更新操作简单快捷，但是，唯一的缺点就是在做查询时需要表关联，关联查询会不仅会带来高的代价，而且还可能造成索引策略失效，导致更低效率的查询。**绝对的范式化是实验室中的产物，在实际的应用中要混用范式化和反范式化，根据具有情况，往往会带来较高的查询效率**。

缓存表和汇总表

有时提升性能最好的方法是将衍生的冗余数据保存到**缓存表或汇总表**，然后执行查询这些表即可得出所需要的数据。这里缓存表表示存储那些从其它表获取（但是每次获取的速度很慢）数据的表，而汇总表保存的是使用group by 语句聚合数据的表（数据不是逻辑上冗余），也可以把这些表称为累积表。比如我们要统计某网站24小时之内发送的信息数量，那么从相关表进行统计就会很慢，如果我们每一个小时就把进行统计一次，并把统计结果存放到一张表中，当我们需要统计24小时之内发送的信息数量时，只需要执行简单地查询就能得到需要的结果，这样就极大地提高的查询效率，但是缺点是数据不是100%精确。

对于衍生数据保存的另外一种方式是使用物化视图，物化视图可以增量地重新计算其内容，不需要通过查询原始表来更新其数据。

计数器表在web应用中很常见，例如记录每个用户的发送信息的数量、下载文件的数量，利用计数器表可以很简单的获取个人的记录数量，比直接从相关数据表中统计的效率高很多，比如一张计数器表，记录网站的点击数量，若只有一行记录，那么在条记录上只能存在一个排它锁，更新查询操作等只能串行进行，影响并发，若将记录保存在多行上，每次随机选择一行进行更新，那么并发性能就会大幅提高。多行记录与单行记录的表设计，要参照具体的应用。

参考：https://www.cnblogs.com/bypp/p/7755307.html

MySQL的索引分类

索引分类

1.普通索引index :加速查找

2.唯一索引

主键索引：primary key ：加速查找+约束（不为空且唯一）

唯一索引：unique：加速查找+约束 （唯一）

3.联合索引

-primary key(id,name):联合主键索引

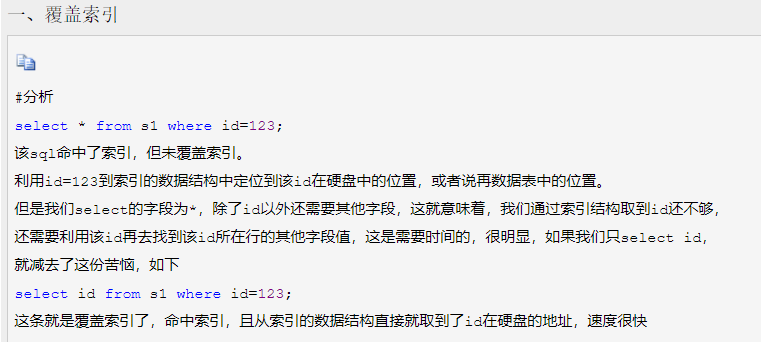
-unique(id,name):联合唯一索引

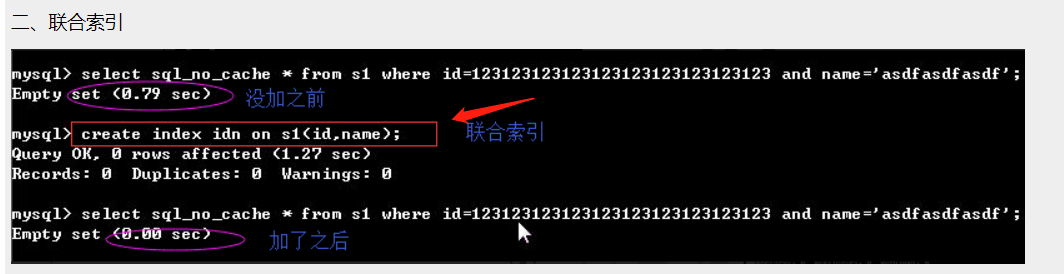
-index(id,name):联合普通索引

4.全文索引fulltext :用于搜索很长一篇文章的时候，效果最好。

5.空间索引spatial :了解就好，几乎不用

使用例子：











有关索引：https://zhuanlan.zhihu.com/p/335734145