[**MySQL性能优化的21个最佳实践**](http://mygirl1314520.iteye.com/blog/829377)

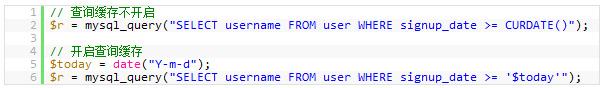
**文章分类:**[**数据库**](http://www.iteye.com/blogs/category/database)

今天，数据库的操作越来越成为整个应用的性能瓶颈了，这点对于Web应用尤其明显。关于数据库的性能，这并不只是DBA才需要担心的事，而这更是我们程序员需要去关注的事情。当我们去设计数据库表结构，对操作数据库时(尤其是查表时的SQL语句)，我们都需要注意数据操作的性能。这里，我们不会讲过多的SQL语句的优化，而只是针对MySQL这一Web应用最多的数据库。希望下面的这些优化技巧对你有用。

**1. 为查询缓存优化你的查询**

　　大多数的MySQL服务器都开启了查询缓存。这是提高性最有效的方法之一，而且这是被MySQL的数据库引擎处理的。当有很多相同的查询被执行了多次的时候，这些查询结果会被放到一个缓存中，这样，后续的相同的查询就不用操作表而直接访问缓存结果了。

　　这里最主要的问题是，对于程序员来说，这个事情是很容易被忽略的。因为，我们某些查询语句会让MySQL不使用缓存。请看下面的示例：



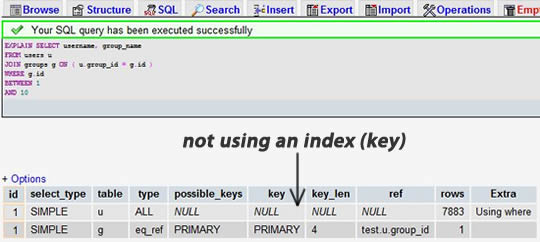
　　上面两条SQL语句的差别就是 CURDATE() ，MySQL的查询缓存对这个函数不起作用。所以，像 NOW() 和 RAND() 或是其它的诸如此类的SQL函数都不会开启查询缓存，因为这些函数的返回是会不定的易变的。所以，你所需要的就是用一个变量来代替MySQL的函数，从而开启缓存。

**2. EXPLAIN 你的 SELECT 查询**

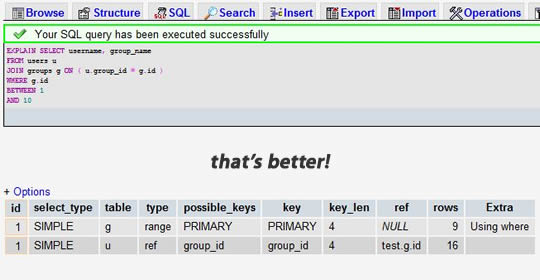
　　使用 EXPLAIN 关键字可以让你知道MySQL是如何处理你的SQL语句的。这可以帮你分析你的查询语句或是表结构的性能瓶颈。

　　EXPLAIN 的查询结果还会告诉你你的索引主键被如何利用的，你的数据表是如何被搜索和排序的……等等，等等。

　　挑一个你的SELECT语句(推荐挑选那个最复杂的，有多表联接的)，把关键字EXPLAIN加到前面。你可以使用phpmyadmin来做这个事。然后，你会看到一张表格。下面的这个示例中，我们忘记加上了group\_id索引，并且有表联接：



　　当我们为 group\_id 字段加上索引后：



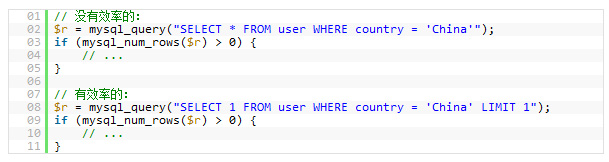
　　我们可以看到，前一个结果显示搜索了 7883 行，而后一个只是搜索了两个表的 9 和 16 行。查看rows列可以让我们找到潜在的性能问题。

**3. 当只要一行数据时使用 LIMIT 1**

　　当你查询表的有些时候，你已经知道结果只会有一条结果，但因为你可能需要去fetch游标，或是你也许会去检查返回的记录数。

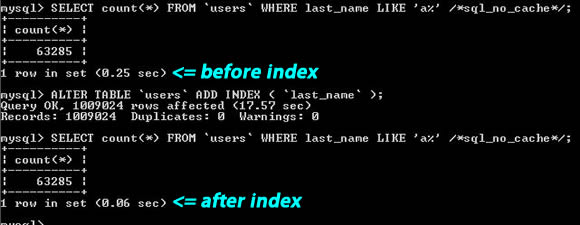
　　在这种情况下，加上 LIMIT 1 可以增加性能。这样一样，MySQL数据库引擎会在找到一条数据后停止搜索，而不是继续往后查少下一条符合记录的数据。

　　下面的示例，只是为了找一下是否有“中国”的用户，很明显，后面的会比前面的更有效率。(请注意，第一条中是Select \*，第二条是Select 1)



**4. 为搜索字段建索引**

　　索引并不一定就是给主键或是唯一的字段。如果在你的表中，有某个字段你总要会经常用来做搜索，那么，请为其建立索引吧。



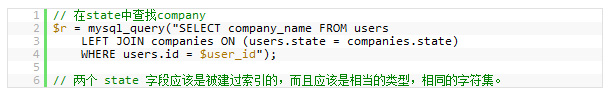
　　从上图你可以看到那个搜索字串 “last\_name LIKE ‘a%’”，一个是建了索引，一个是没有索引，性能差了4倍左右。

　　另外，你应该也需要知道什么样的搜索是不能使用正常的索引的。例如，当你需要在一篇大的文章中搜索一个词时，如： “WHERE post\_content LIKE ‘%apple%’”，索引可能是没有意义的。你可能需要使用MySQL全文索引 或是自己做一个索引(比如说：搜索关键词或是Tag什么的)

**5. 在Join表的时候使用相当类型的例，并将其索引**

　　如果你的应用程序有很多 JOIN 查询，你应该确认两个表中Join的字段是被建过索引的。这样，MySQL内部会启动为你优化Join的SQL语句的机制。

　　而且，这些被用来Join的字段，应该是相同的类型的。例如：如果你要把 DECIMAL 字段和一个 INT 字段Join在一起，MySQL就无法使用它们的索引。对于那些STRING类型，还需要有相同的字符集才行。(两个表的字符集有可能不一样)

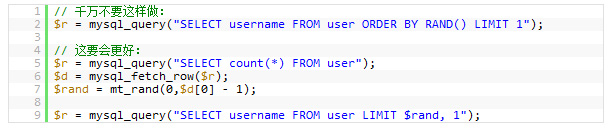


**6. 千万不要 ORDER BY RAND()**

　　想打乱返回的数据行?随机挑一个数据?真不知道谁发明了这种用法，但很多新手很喜欢这样用。但你确不了解这样做有多么可怕的性能问题。

　　如果你真的想把返回的数据行打乱了，你有N种方法可以达到这个目的。这样使用只让你的数据库的性能呈指数级的下降。这里的问题是：MySQL会不得不去执行RAND()函数(很耗CPU时间)，而且这是为了每一行记录去记行，然后再对其排序。就算是你用了Limit 1也无济于事(因为要排序)

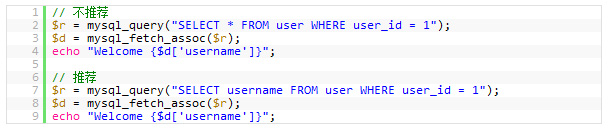
　　下面的示例是随机挑一条记录



**7. 避免 SELECT \***

　　从数据库里读出越多的数据，那么查询就会变得越慢。并且，如果你的数据库服务器和WEB服务器是两台独立的服务器的话，这还会增加网络传输的负载。

　　所以，你应该养成一个需要什么就取什么的好的习惯。



**8. 永远为每张表设置一个ID**

　　我们应该为数据库里的每张表都设置一个ID做为其主键，而且最好的是一个INT型的(推荐使用UNSIGNED)，并设置上自动增加的AUTO\_INCREMENT标志。

　　就算是你 users 表有一个主键叫 “email”的字段，你也别让它成为主键。使用 VARCHAR 类型来当主键会使用得性能下降。另外，在你的程序中，你应该使用表的ID来构造你的数据结构。

　　而且，在MySQL数据引擎下，还有一些操作需要使用主键，在这些情况下，主键的性能和设置变得非常重要，比如，集群，分区……

　　在这里，只有一个情况是例外，那就是“关联表”的“外键”，也就是说，这个表的主键，通过若干个别的表的主键构成。我们把这个情况叫做“外键”。比如：有一个“学生表”有学生的ID，有一个“课程表”有课程ID，那么，“成绩表”就是“关联表”了，其关联了学生表和课程表，在成绩表中，学生ID和课程 ID叫“外键”其共同组成主键。

**9. 使用 ENUM 而不是 VARCHAR**

　　ENUM 类型是非常快和紧凑的。在实际上，其保存的是 TINYINT，但其外表上显示为字符串。这样一来，用这个字段来做一些选项列表变得相当的完美。

　　如果你有一个字段，比如“性别”，“国家”，“民族”，“状态”或“部门”，你知道这些字段的取值是有限而且固定的，那么，你应该使用 ENUM 而不是 VARCHAR。

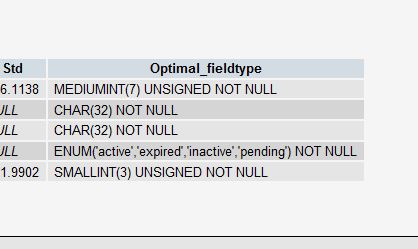
　　MySQL也有一个“建议”(见第十条)告诉你怎么去重新组织你的表结构。当你有一个 VARCHAR 字段时，这个建议会告诉你把其改成 ENUM 类型。使用 PROCEDURE ANALYSE() 你可以得到相关的建议。

**10. 从 PROCEDURE ANALYSE() 取得建议**

　　PROCEDURE ANALYSE() 会让 MySQL 帮你去分析你的字段和其实际的数据，并会给你一些有用的建议。只有表中有实际的数据，这些建议才会变得有用，因为要做一些大的决定是需要有数据作为基础的。

　　例如，如果你创建了一个 INT 字段作为你的主键，然而并没有太多的数据，那么，PROCEDURE ANALYSE()会建议你把这个字段的类型改成 MEDIUMINT 。或是你使用了一个 VARCHAR 字段，因为数据不多，你可能会得到一个让你把它改成 ENUM 的建议。这些建议，都是可能因为数据不够多，所以决策做得就不够准。

　　在phpmyadmin里，你可以在查看表时，点击 “Propose table structure” 来查看这些建议



　　一定要注意，这些只是建议，只有当你的表里的数据越来越多时，这些建议才会变得准确。一定要记住，你才是最终做决定的人。

**11. 尽可能的使用 NOT NULL**

　　除非你有一个很特别的原因去使用 NULL 值，你应该总是让你的字段保持 NOT NULL。这看起来好像有点争议，请往下看。

　　首先，问问你自己“Empty”和“NULL”有多大的区别(如果是INT，那就是0和NULL)?如果你觉得它们之间没有什么区别，那么你就不要使用NULL。(你知道吗?在 Oracle 里，NULL 和 Empty 的字符串是一样的!)

　　不要以为 NULL 不需要空间，其需要额外的空间，并且，在你进行比较的时候，你的程序会更复杂。 当然，这里并不是说你就不能使用NULL了，现实情况是很复杂的，依然会有些情况下，你需要使用NULL值。

**12. Prepared Statements**

　　Prepared Statements很像存储过程，是一种运行在后台的SQL语句集合，我们可以从使用 prepared statements 获得很多好处，无论是性能问题还是安全问题。

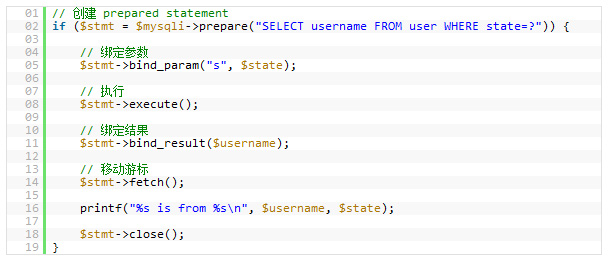
　　Prepared Statements 可以检查一些你绑定好的变量，这样可以保护你的程序不会受到“SQL注入式”攻击。当然，你也可以手动地检查你的这些变量，然而，手动的检查容易出问题，而且很经常会被程序员忘了。当我们使用一些framework或是ORM的时候，这样的问题会好一些。

　　在性能方面，当一个相同的查询被使用多次的时候，这会为你带来可观的性能优势。你可以给这些Prepared Statements定义一些参数，而MySQL只会解析一次。

　　虽然最新版本的MySQL在传输Prepared Statements是使用二进制形势，所以这会使得网络传输非常有效率。

　　当然，也有一些情况下，我们需要避免使用Prepared Statements，因为其不支持查询缓存。但据说版本5.1后支持了。

　　在PHP中要使用prepared statements，你可以查看其使用手册：mysqli 扩展 或是使用数据库抽象层，如： PDO.



**13. 无缓冲的查询**

　　正常的情况下，当你在当你在你的脚本中执行一个SQL语句的时候，你的程序会停在那里直到没这个SQL语句返回，然后你的程序再往下继续执行。你可以使用无缓冲查询来改变这个行为。

　　mysql\_unbuffered\_query() 发送一个SQL语句到MySQL而并不像mysql\_query()一样去自动fethch和缓存结果。这会相当节约很多可观的内存，尤其是那些会产生大量结果的查询语句，并且，你不需要等到所有的结果都返回，只需要第一行数据返回的时候，你就可以开始马上开始工作于查询结果了。

　　然而，这会有一些限制。因为你要么把所有行都读走，或是你要在进行下一次的查询前调用 mysql\_free\_result() 清除结果。而且， mysql\_num\_rows() 或 mysql\_data\_seek() 将无法使用。所以，是否使用无缓冲的查询你需要仔细考虑。

**14. 把IP地址存成 UNSIGNED INT**

　　很多程序员都会创建一个 VARCHAR(15) 字段来存放字符串形式的IP而不是整形的IP。如果你用整形来存放，只需要4个字节，并且你可以有定长的字段。而且，这会为你带来查询上的优势，尤其是当你需要使用这样的WHERE条件：IP between ip1 and ip2。

　　我们必需要使用UNSIGNED INT，因为 IP地址会使用整个32位的无符号整形。

　　而你的查询，你可以使用 INET\_ATON() 来把一个字符串IP转成一个整形，并使用 INET\_NTOA() 把一个整形转成一个字符串IP。在PHP中，也有这样的函数 ip2long() 和 long2ip()。

http://www.searchdatabase.com.cn/upload/article/2010/2010-07-22-11-11-59.jpg

**15. 固定长度的表会更快**

　　如果表中的所有字段都是“固定长度”的，整个表会被认为是 “static” 或 “fixed-length”。 例如，表中没有如下类型的字段： VARCHAR，TEXT，BLOB。只要你包括了其中一个这些字段，那么这个表就不是“固定长度静态表”了，这样，MySQL 引擎会用另一种方法来处理。

　　固定长度的表会提高性能，因为MySQL搜寻得会更快一些，因为这些固定的长度是很容易计算下一个数据的偏移量的，所以读取的自然也会很快。而如果字段不是定长的，那么，每一次要找下一条的话，需要程序找到主键。

　　并且，固定长度的表也更容易被缓存和重建。不过，唯一的副作用是，固定长度的字段会浪费一些空间，因为定长的字段无论你用不用，他都是要分配那么多的空间。

　　使用“垂直分割”技术(见下一条)，你可以分割你的表成为两个一个是定长的，一个则是不定长的。

**16. 垂直分割**

　　“垂直分割”是一种把数据库中的表按列变成几张表的方法，这样可以降低表的复杂度和字段的数目，从而达到优化的目的。(以前，在银行做过项目，见过一张表有100多个字段，很恐怖)

　　示例一：在Users表中有一个字段是家庭地址，这个字段是可选字段，相比起，而且你在数据库操作的时候除了个人信息外，你并不需要经常读取或是改写这个字段。那么，为什么不把他放到另外一张表中呢? 这样会让你的表有更好的性能，大家想想是不是，大量的时候，我对于用户表来说，只有用户ID，用户名，口令，用户角色等会被经常使用。小一点的表总是会有好的性能。

　　示例二： 你有一个叫 “last\_login” 的字段，它会在每次用户登录时被更新。但是，每次更新时会导致该表的查询缓存被清空。所以，你可以把这个字段放到另一个表中，这样就不会影响你对用户 ID，用户名，用户角色的不停地读取了，因为查询缓存会帮你增加很多性能。

　　另外，你需要注意的是，这些被分出去的字段所形成的表，你不会经常性地去Join他们，不然的话，这样的性能会比不分割时还要差，而且，会是极数级的下降。

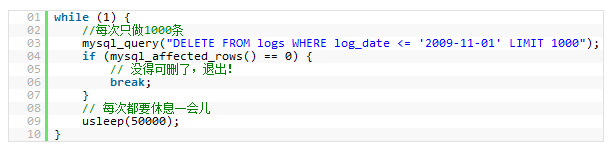
**17. 拆分大的 DELETE 或 INSERT 语句**

　　如果你需要在一个在线的网站上去执行一个大的 DELETE 或 INSERT 查询，你需要非常小心，要避免你的操作让你的整个网站停止相应。因为这两个操作是会锁表的，表一锁住了，别的操作都进不来了。

　　Apache 会有很多的子进程或线程。所以，其工作起来相当有效率，而我们的服务器也不希望有太多的子进程，线程和数据库链接，这是极大的占服务器资源的事情，尤其是内存。

　　如果你把你的表锁上一段时间，比如30秒钟，那么对于一个有很高访问量的站点来说，这30秒所积累的访问进程/线程，数据库链接，打开的文件数，可能不仅仅会让你泊WEB服务Crash，还可能会让你的整台服务器马上掛了。

　　所以，如果你有一个大的处理，你定你一定把其拆分，使用 LIMIT 条件是一个好的方法。下面是一个示例：



**18. 越小的列会越快**

　　对于大多数的数据库引擎来说，硬盘操作可能是最重大的瓶颈。所以，把你的数据变得紧凑会对这种情况非常有帮助，因为这减少了对硬盘的访问。

　　参看 MySQL 的文档 Storage Requirements 查看所有的数据类型。

　　如果一个表只会有几列罢了(比如说字典表，配置表)，那么，我们就没有理由使用 INT 来做主键，使用 MEDIUMINT, SMALLINT 或是更小的 TINYINT 会更经济一些。如果你不需要记录时间，使用 DATE 要比 DATETIME 好得多。

　　当然，你也需要留够足够的扩展空间，不然，你日后来干这个事，你会死的很难看，参看Slashdot的例子(2009年11月06日)，一个简单的ALTER TABLE语句花了3个多小时，因为里面有一千六百万条数据。

**19. 选择正确的存储引擎**

　　在 MySQL 中有两个存储引擎 MyISAM 和 InnoDB，每个引擎都有利有弊。酷壳以前文章《MySQL: InnoDB 还是 MyISAM?》讨论和这个事情。

　　MyISAM 适合于一些需要大量查询的应用，但其对于有大量写操作并不是很好。甚至你只是需要update一个字段，整个表都会被锁起来，而别的进程，就算是读进程都无法操作直到读操作完成。另外，MyISAM 对于 SELECT COUNT(\*) 这类的计算是超快无比的。

　　InnoDB 的趋势会是一个非常复杂的存储引擎，对于一些小的应用，它会比 MyISAM 还慢。他是它支持“行锁” ，于是在写操作比较多的时候，会更优秀。并且，他还支持更多的高级应用，比如：事务。

　　下面是MySQL的手册

　　target=”\_blank”MyISAM Storage Engine

　　InnoDB Storage Engine

**20. 使用一个对象关系映射器(Object Relational Mapper)**

　　使用 ORM (Object Relational Mapper)，你能够获得可靠的性能增涨。一个ORM可以做的所有事情，也能被手动的编写出来。但是，这需要一个高级专家。

　　ORM 的最重要的是“Lazy Loading”，也就是说，只有在需要的去取值的时候才会去真正的去做。但你也需要小心这种机制的副作用，因为这很有可能会因为要去创建很多很多小的查询反而会降低性能。

　　ORM 还可以把你的SQL语句打包成一个事务，这会比单独执行他们快得多得多。

　　目前，个人最喜欢的PHP的ORM是：Doctrine。

**21. 小心“永久链接”**

　　“永久链接”的目的是用来减少重新创建MySQL链接的次数。当一个链接被创建了，它会永远处在连接的状态，就算是数据库操作已经结束了。而且，自从我们的Apache开始重用它的子进程后——也就是说，下一次的HTTP请求会重用Apache的子进程，并重用相同的 MySQL 链接。

　　PHP手册：mysql\_pconnect()

　　在理论上来说，这听起来非常的不错。但是从个人经验(也是大多数人的)上来说，这个功能制造出来的麻烦事更多。因为，你只有有限的链接数，内存问题，文件句柄数，等等。

　　而且，Apache 运行在极端并行的环境中，会创建很多很多的了进程。这就是为什么这种“永久链接”的机制工作地不好的原因。在你决定要使用“永久链接”之前，你需要好好地考虑一下你的整个系统的架构。