**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：

// 查询缓存不开启

$r = mysql\_query("SELECT username FROM user WHERE signup\_date >= CURDATE()");

// 开启查询缓存

$today = date("Y-m-d");

$r = mysql\_query("SELECT username FROM user WHERE signup\_date >= '$today'");

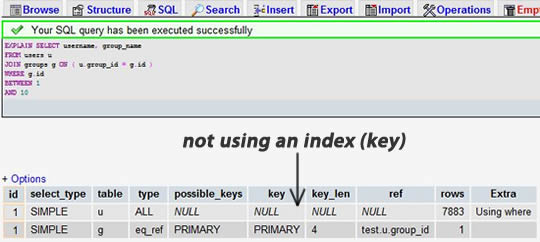
上面两条SQL语句的差别就是 CURDATE() ，MySQL的查询缓存对这个函数不起作用。所以，像 NOW() 和 RAND() 或是其它的诸如此类的SQL函数都不会开启查询缓存，因为这些函数的返回是会不定的易变的。所以，你所需要的就是用一个变量来代替MySQL的函数，从而开启缓存。

#### 2. EXPLAIN 你的 SELECT 查询

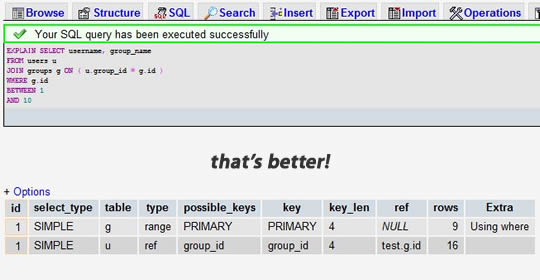
使用 [EXPLAIN](http://dev.mysql.com/doc/refman/5.0/en/explain.html) 关键字可以让你知道MySQL是如何处理你的SQL语句的。这可以帮你分析你的查询语句或是表结构的性能瓶颈。

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）

// 没有效率的：

$r = mysql\_query("SELECT \* FROM user WHERE country = 'China'");

if (mysql\_num\_rows($r) > 0) {

// ...

}

// 有效率的：

$r = mysql\_query("SELECT 1 FROM user WHERE country = 'China' LIMIT 1");

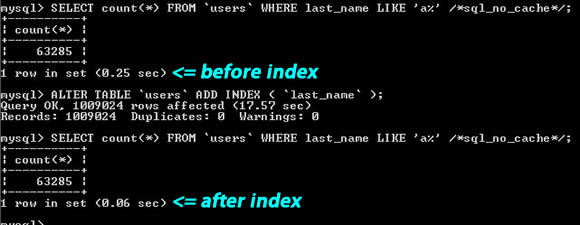
if (mysql\_num\_rows($r) > 0) {

// ...

}

**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](http://dev.mysql.com/doc/refman/5.1/en/fulltext-search.html" \t "_blank) 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）

// 在state中查找company

$r = mysql\_query("SELECT company\_name FROM users

LEFT JOIN companies ON (users.state = companies.state)

WHERE users.id = $user\_id");

// 两个 state 字段应该是被建过索引的，而且应该是相当的类型，相同的字符集。

#### 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

// 千万不要这样做：

$r = mysql\_query("SELECT username FROM user ORDER BY RAND() LIMIT 1");

// 这要会更好：

$r = mysql\_query("SELECT count(\*) FROM user");

$d = mysql\_fetch\_row($r);

$rand = mt\_rand(0,$d[0] - 1);

$r = mysql\_query("SELECT username FROM user LIMIT $rand, 1");

#### 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。

// 不推荐

$r = mysql\_query("SELECT \* FROM user WHERE user\_id = 1");

$d = mysql\_fetch\_assoc($r);

echo "Welcome {$d['username']}";

// 推荐

$r = mysql\_query("SELECT username FROM user WHERE user\_id = 1");

$d = mysql\_fetch\_assoc($r);

echo "Welcome {$d['username']}";

#### 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](http://dev.mysql.com/doc/refman/5.0/en/enum.html) 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()](http://dev.mysql.com/doc/refman/5.0/en/procedure-analyse.html) 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