[**为mysql数据库建立索引**](http://www.cnblogs.com/cy163/archive/2008/10/27/1320798.html)

前些时候，一位颇高级的程序员居然问我什么叫做索引，令我感到十分的惊奇，我想这绝不会是沧海一粟，因为有成千上万的开发者（可能大部分是使用MySQL的）都没有受过有关数据库的正规培训，尽管他们都为客户做过一些开发，但却对如何为数据库建立适当的索引所知较少，因此我起了写一篇相关[文章](javascript:void(0))的念头。  
  
  最普通的情况，是为出现在where子句的字段建一个索引。为方便讲述，我们先建立一个如下的表。

|  |
| --- |
| Code代码如下: |
| CREATE TABLE mytable ( 　id serial primary key, 　category\_id int not null default 0, 　user\_id int not null default 0, 　adddate int not null default 0 ); |

  很简单吧，不过对于要说明这个问题，已经足够了。如果你在查询时常用类似以下的语句：  
  
SELECT \* FROM mytable WHERE category\_id=1;   
  
  最直接的应对之道，是为category\_id建立一个简单的索引：  
  
CREATE INDEX mytable\_categoryid   
　ON mytable (category\_id);  
  
  OK，搞定？先别高兴，如果你有不止一个[选择](javascript:void(0))条件呢？例如：  
  
SELECT \* FROM mytable WHERE category\_id=1 AND user\_id=2;  
  
  你的第一反应可能是，再给user\_id建立一个索引。不好，这不是一个最佳的方法。你可以建立多重的索引。  
  
CREATE INDEX mytable\_categoryid\_userid ON mytable (category\_id,user\_id);  
  
  注意到我在命名时的习惯了吗？我使用"表名\_字段1名\_字段2名"的方式。你很快就会知道我为什么这样做了。  
  
  现在你已经为适当的字段建立了索引，不过，还是有点不放心吧，你可能会问，数据库会真正用到这些索引吗？测试一下就OK，对于大多数的数据库来说，这是很容易的，只要使用EXPLAIN命令：  
  
EXPLAIN  
  
　SELECT \* FROM mytable   
　　WHERE category\_id=1 AND user\_id=2;  
  
This is what Postgres 7.1 returns (exactly as I expected)   
  
　NOTICE: QUERY PLAN:  
  
Index Scan using mytable\_categoryid\_userid on   
  mytable (cost=0.00..2.02 rows=1 width=16)  
  
EXPLAIN  
  
  以上是postgres的数据，可以看到该数据库在查询的时候使用了一个索引（一个好开始），而且它使用的是我创建的第二个索引。看到我上面命名的好处了吧，你马上知道它使用适当的索引了。  
  
  接着，来个稍微复杂一点的，如果有个ORDER BY字句呢？不管你信不信，大多数的数据库在使用order by的时候，都将会从索引中受益。  
  
SELECT \* FROM mytable   
  WHERE category\_id=1 AND user\_id=2  
    ORDER BY adddate DESC;  
  
  有点迷惑了吧？很简单，就象为where字句中的字段建立一个索引一样，也为ORDER BY的字句中的字段建立一个索引：  
  
CREATE INDEX mytable\_categoryid\_userid\_adddate  
  ON mytable (category\_id,user\_id,adddate);  
  
  注意: "mytable\_categoryid\_userid\_adddate" 将会被截短为  
  
"mytable\_categoryid\_userid\_addda"  
  
CREATE  
  
  EXPLAIN SELECT \* FROM mytable  
　　WHERE category\_id=1 AND user\_id=2  
　　　ORDER BY adddate DESC;  
  
　NOTICE: QUERY PLAN:  
  
　Sort (cost=2.03..2.03 rows=1 width=16)  
　　-> Index Scan using mytable\_categoryid\_userid\_addda   
　　　　on mytable (cost=0.00..2.02 rows=1 width=16)  
  
EXPLAIN  
  
  看看EXPLAIN的输出，好象有点恐怖啊，数据库多做了一个我们没有要求的排序，这下知道性能如何受损了吧，看来我们对于数据库的自身运作是有点过于乐观了，那么，给数据库多一点提示吧。  
  
  为了跳过排序这一步，我们并不需要其它另外的索引，只要将查询语句稍微改一下。这里用的是postgres，我们将给该数据库一个额外的提示--在ORDER BY语句中，[加入](javascript:void(0))where语句中的字段。这只是一个技术上的处理，并不是必须的，因为实际上在另外两个字段上，并不会有任何的排序操作，不过如果加入，postgres将会知道哪些是它应该做的。  
  
EXPLAIN SELECT \* FROM mytable   
  WHERE category\_id=1 AND user\_id=2  
　　ORDER BY category\_id DESC,user\_id DESC,adddate DESC;  
  
NOTICE: QUERY PLAN:  
  
Index Scan Backward using   
　mytable\_categoryid\_userid\_addda on mytable   
　  (cost=0.00..2.02 rows=1 width=16)  
  
EXPLAIN  
  
  现在使用我们料想的索引了，而且它还挺[聪明](javascript:void(0))，知道可以从索引后面开始读，从而避免了任何的排序。  
  
  以上说得细了一点，不过如果你的数据库非常巨大，并且每日的页面请求达上百万算，我想你会获益良多的。不过，如果你要做更为复杂的查询呢，例如将多张表结合起来查询，特别是where限制字句中的字段是来自不止一个表格时，应该怎样处理呢？我通常都尽量避免这种做法，因为这样数据库要将各个表中的东西都结合起来，然后再排除那些不合适的行，搞不好开销会很大。  
  
  如果不能避免，你应该查看每张要结合起来的表，并且使用以上的策略来建立索引，然后再用EXPLAIN命令验证一下是否使用了你料想中的索引。如果是的话，就OK。不是的话，你可能要建立临时的表来将他们结合在一起，并且使用适当的索引。  
  
  要注意的是，建立太多的索引将会[影响](javascript:void(0))更新和插入的速度，因为它需要同样更新每个索引文件。对于一个经常需要更新和插入的表格，就没有必要为一个很少使用的where字句单独建立索引了，对于比较小的表，排序的开销不会很大，也没有必要建立另外的索引。  
  
  以上介绍的只是一些十分基本的东西，其实里面的学问也不少，单凭EXPLAIN我们是不能判定该方法是否就是最优化的，每个数据库都有自己的一些优化器，虽然可能还不太完善，但是它们都会在查询时对比过哪种方式较快，在某些情况下，建立索引的话也未必会快，例如索引放在一个不连续的存储空间时，这会[增加](javascript:void(0))读磁盘的负担，因此，哪个是最优，应该通过实际的使用环境来检验。  
  
  在刚开始的时候，如果表不大，没有必要作索引，我的意见是在需要的时候才作索引，也可用一些命令来优化表，例如MySQL可用"OPTIMIZE TABLE"。  
  
  综上所述，在如何为数据库建立恰当的索引方面，你应该有一些基本的概念了。

分类: [MySQL](http://www.cnblogs.com/cy163/category/147582.html)