MySQL优化技巧

摘自：http://blog.csdn.net/u013087513/article/details/77899412

**MySQL优化三大方向**

① 优化MySQL所在服务器内核(此优化一般由运维人员完成)。

② 对MySQL配置参数进行优化（my.cnf）此优化需要进行压力测试来进行参数调整。

③ 对SQL语句以及表优化。

# MySQL参数优化

1:MySQL 默认的最大连接数为 100，可以在 mysql 客户端使用以下命令查看  
mysql> show variables like 'max\_connections';  
2:查看当前访问Mysql的线程  
mysql> show processlist;  
3:设置最大连接数  
mysql>set globle max\_connections = 5000;  
最大可设置16384,超过没用  
4:查看当前被使用的connections  
mysql>show globle status like 'max\_user\_connections'

# 对MySQL语句性能优化的16条经验

① 为查询缓存优化查询

② EXPLAIN 我们的SELECT查询(可以查看执行的行数)

③ 当只要一行数据时使用LIMIT 1

④ 为搜索字段建立索引

⑤ 在Join表的时候使用相当类型的列，并将其索引

⑥ 千万不要 ORDER BY RAND  ()

⑦ 避免SELECT \*

⑧ 永远为每张表设置一个ID

⑨ 可以使用ENUM 而不要VARCHAR

⑩ 尽可能的使用NOT NULL

⑪ 固定长度的表会更快

⑫ 垂直分割

⑬ 拆分打的DELETE或INSERT语句

⑭ 越小的列会越快

⑮ 选择正确的存储引擎

⑯ 小心 "永久链接"

具体描述如下：

1. 使用查询缓存优化查询

大多数的MySQL服务器都开启了查询缓存。这是提高性能最有效的方法之一，而且这是被MySQL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入一个缓存中，这样后续的相同查询就不用操作而直接访问缓存结果了。

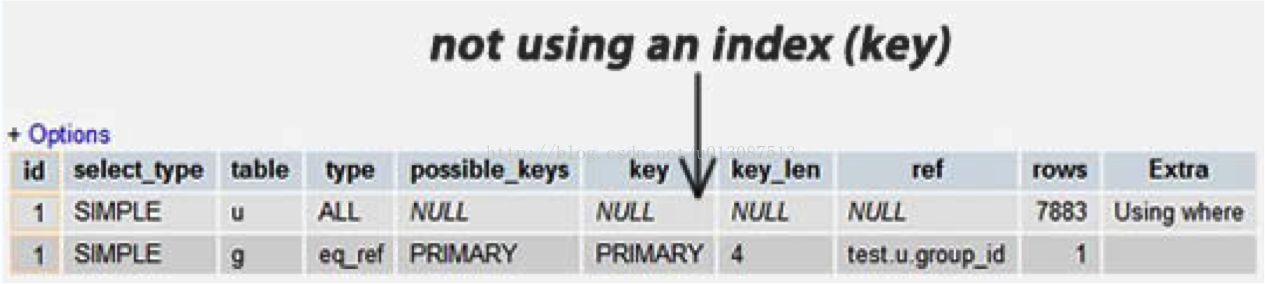
这里最主要的问题是，对于我们程序员来说，这个事情是很容易被忽略的。因为我们某些查询语句会让MySQL不使用缓存，示例如下：

1：SELECT username FROM user WHERE signup\_date >= CURDATE()  
2：SELECT username FROM user WHERE signup\_date >= '2014-06-24’

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。

1. 使用EXPLAIN关键字检测查询

使用EXPLAIN关键字可以使我们知道MySQL是如何处理SQL语句的，这样可以帮助我们分析我们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我们索引主键是如何被利用的，数据表是如何被被搜索或排序的....等等。语法格式是：EXPLAIN +SELECT语句;





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。

1. 当只要一行数据时使用LIMIT 1

加上LIMIT 1可以增加性能。MySQL数据库引擎会在查找到一条数据后停止搜索，而不是继续往后查询下一条符合条件的数据记录。

1. 为搜索字段建立索引

索引不一定就是给主键或者是唯一的字段，如果在表中，有某个字段经常用来做搜索，需要将其建立索引。

索引的有关操作如下：

1. 创建索引

在执行CREATE TABLE语句时可以创建索引，也可以单独用CREATE INDEX或ALTER TABLE来为表增加索引。

* 1. ALTER TABLE

ALTER TABLE 用来创建普通索引、唯一索引、主键索引和全文索引

ALTER TABLE table\_name ADD INDEX index\_name (column\_list);

ALTER TABLE table\_name ADD UNIQUE (column\_list);

ALTER TABLE table\_name ADD PRIMARY KEY (column\_list);

ALTER TABLE table\_name ADD FULLTEXT (column\_list);

其中table\_name是要增加索引名的表名，column\_list指出对哪些列列进行索引，多列时各列之间使用半角逗号隔开。索引名index\_name是可选的，如果不指定索引名称，MySQL将根据第一个索引列自动指定索引名称，另外，ALTER TABLE允许在单个语句中更改多个表，因此可以在同时创建多个索引。

* 1. CREATE INDEX

CREATE INDEX可对表增加普通索引或UNIQUE索引以及全文索引，但是不可以对表增加主键索引

CREATE INDEX index\_name ON table\_name (column\_list);

CREATE UNIQUE index\_name ON table\_name (column\_list);

CREATE FULLTEXT index\_name ON table\_name (column\_list);

table\_name、index\_name和column\_list具有与ALTER TABLE语句中相同的含义，索引名必须指定。另外，不能用CREATE INDEX语句创建PRIMARY KEY索引。

1. 索引类型

普通索引INDEX：适用于name、email等一般属性

唯一索引UNIQUE：与普通索引类似，不同的是唯一索引要求索引字段值在表中是唯一的，这一点和主键索引类似，但是不同的是，唯一索引允许有空值。唯一索引一般适用于身份证号码、用户账号等不允许有重复的属性字段上。

主键索引：其实就是主键，一般在建表时就指定了，不需要额外添加。

全文检索：只适用于VARCHAR和Text类型的字段。

注意：全文索引和普通索引是有很大区别的，如果建立的是普通索引，一般会使用like进行模糊查询，只会对查询内容前一部分有效，即只对前面不使用通配符的查询有效，如果前后都有通配符，普通索引将不会起作用。对于全文索引而言在查询时有自己独特的匹配方式，例如我们在对一篇文章的标题和内容进行全文索引时：

ALTER TABLE article ADD FULLTEXT ('title', 'content'); 在进行检索时就需要使用如下的语法进行检索：

SELECT \* FROM article WHERE MATCH('title', 'content') AGAINST ('查询字符串');

在使用全文检索时的注意事项：

MySql自带的全文索引只能用于数据库引擎为MYISAM的数据表，如果是其他数据引擎，则全文索引不会生效。此外，MySql自带的全文索引只能对英文进行全文检索，目前无法对中文进行全文检索。如果需要对包含中文在内的文本数据进行全文检索，我们需要采用Sphinx（斯芬克斯）/Coreseek技术来处理中文。另外使用MySql自带的全文索引时，如果查询字符串的长度过短将无法得到期望的搜索结果。MySql全文索引所能找到的词默认最小长度为4个字符。另外，如果查询的字符串包含停止词，那么该停止词将会被忽略。

1. 组合索引

组合索引又称多列索引，就是建立索引时指定多个字段属性。有点类似于字典目录，比如查询 'guo' 这个拼音的字时，首先查找g字母，然后在g的检索范围内查询第二个字母为u的列表，最后在u的范围内查找最后一个字母为o的字。比如组合索引(a,b,c)，abc都是排好序的，在任意一段a的下面b都是排好序的，任何一段b下面c都是排好序的

组合索引的生效原则是  从前往后依次使用生效，如果中间某个索引没有使用，那么断点前面的索引部分起作用，断点后面的索引没有起作用；

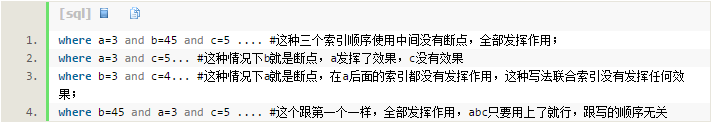
造成断点的原因：

前边的任意一个索引没有参与查询，后边的全部不生效。

前边的任意一个索引字段参与的是范围查询，后面的不会生效。

断点跟索引字字段在SQL语句中的位置前后无关，只与是否存在有关。在网上找到了很好的示例：

比如：



（a,b,c） 三个列上加了联合索引（是联合索引 不是在每个列上单独加索引）而是建立了a,(a,b),(a,b,c)三个索引，另外(a,b,c)多列索引和 (a,c,b)是不一样的。  
具体实例可以说明：



注意：在查询时，MYSQL只能使用一个索引，如果建立的是多个单列的普通索引，在查询时会根据查询的索引字段，从中选择一个限制最严格的单例索引进行查询。别的索引都不会生效。

1. 查看索引

mysql> show index from tblname;

mysql> show keys from tblname;

1. 删除索引

删除索引的mysql格式 :DORP INDEX IndexName ON tab\_name；

注意：不能使用索引的情况

对于普通索引而言 在使用like进行通配符模糊查询时,如果首尾之间都使用了通配符，索引时无效的。

假设查询内容的关键词为'abc'

SELECT \* FROM tab\_name WHERE index\_column LIKE  'abc%';  #索引是有效的

SELECT \* FROM tab\_name WHERE index\_column LIKE  '%abc';  #索引是无效的

SELECT \* FROM tab\_name WHERE index\_column LIKE  '%cba';  #索引是有效的

SELECT \* FROM tab\_name WHERE index\_column LIKE  '%abc%';  #索引是无效的

当检索的字段内容比较大而且检索内容前后部分都不确定的情况下，可以改为全文索引，并使用特定的检索方式。

1. 在join表的时候使用相当类型的列，并将其索引

如果在程序中有很多JOIN查询，应该保证两个表中join的字段时被建立过索引的。这样MySQL颞部会启动优化JOIN的SQL语句的机制。注意：这些被用来JOIN的字段，应该是相同类型的。例如：如果要把 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）

例如：

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。

1. 切记不要使用ORDER BY RAND()

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）

1. 避免使用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，如果我们的数据库服务器和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，我们应该养成一个需要什么就取什么的好的习惯。  
Hibernate性能方面就会差，它不用\*，但它将整个表的所有字段全查出来   
优点：开发速度快

1. 永远为每张表设置一个ID主键

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。   
就算是我们 users 表有一个主键叫 “email”的字段，我们也别让它成为主键。使用 VARCHAR 类型来当主键会使用得性能下降。另外，在我们的程序中，我们应该使用表的ID来构造我们的数据结构。   
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。

1. 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。   
如果我们有一个字段，比如“性别”，“国家”，“民族”，“状态”或“部门”，我们知道这些字段的取值是有限而且固定的，那么，我们应该使用 ENUM 而不是 VARCHAR。

1. 尽可能的不要赋值为NULL

如果不是特殊情况，尽可能的不要使用NULL。在MYSQL中对于INT类型而言，EMPTY是0，而NULL是空值。而在Oracle中 NULL和EMPTY的字符串是一样的。NULL也需要占用存储空间，并且会使我们的程序判断时更加复杂。现实情况是很复杂的，依然会有些情况下，我们需要使用NULL值。

1. 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。只要我们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我们用不用，他都是要分配那么多的空间。另外在取出值的时候要使用trim去除空格

扩展内容：

第一你要理解清楚char和varchar的區別。  
1．CHAR的长度是固定的，而VARCHAR2的长度是可以变化的， 比如，存储字符串“abc"，对于CHAR (10)，表示你存储的字符将占10个字节(包括7个空字符)，而同样的VARCHAR2 (10)则只占用3个字节的长度，10只是最大值，当你存储的字符小于10时，按实际长度存储。   
2．CHAR的效率比VARCHAR2的效率稍高。   
3．目前VARCHAR是VARCHAR2的同义词。工业标准的VARCHAR类型可以存储空字符串，但是oracle不这样做，尽管它保留以后这样做的权利。Oracle自己开发了一个数据类型VARCHAR2，这个类型不是一个标准的VARCHAR，它将在数据库中varchar列可以存储空字符串的特性改为存储NULL值。如果你想有向后兼容的能力，Oracle建议使用VARCHAR2而不是VARCHAR。   
何时该用CHAR，何时该用varchar2？   
CHAR与VARCHAR2是一对矛盾的统一体，两者是互补的关系.   
VARCHAR2比CHAR节省空间，在效率上比CHAR会稍微差一些，即要想获得效率，就必须牺牲一定的空间，这也就是我们在数据库设计上常说的‘以空间换效率’。   
VARCHAR2虽然比CHAR节省空间，但是如果一个VARCHAR2列经常被修改，而且每次被修改的数据的长度不同，这会引起‘行迁移’(Row Migration)现象，而这造成多余的I/O，是数据库设计和调整中要尽力避免的，在这种情况下用CHAR代替VARCHAR2会更好一些。

1. 垂直分割

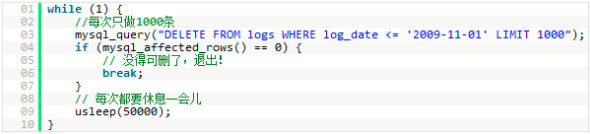
“垂直分割”是一种把数据库中的表按列变成几张表的方法，这样可以降低表的复杂度和字段的数目，从而达到优化的目的。

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。

1. 拆分大的DELETE或INSERT

如果我们需要在一个在线的网站上去执行一个大的 DELETE 或 INSERT 查询，我们需要非常小心，要避免我们的操作让我们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。如果我们把我们的表锁上一段时间，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，可能不仅仅会让我们的WEB服务Crash，还可能会让我们的整台服务器马上掛了。所以在使用时使用LIMIT 控制数量操作记录的数量。



1. 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，把我们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。如果我们不需要记录时间，使用 DATE 要比 DATETIME 好得多。

1. 选择正确的存储引擎

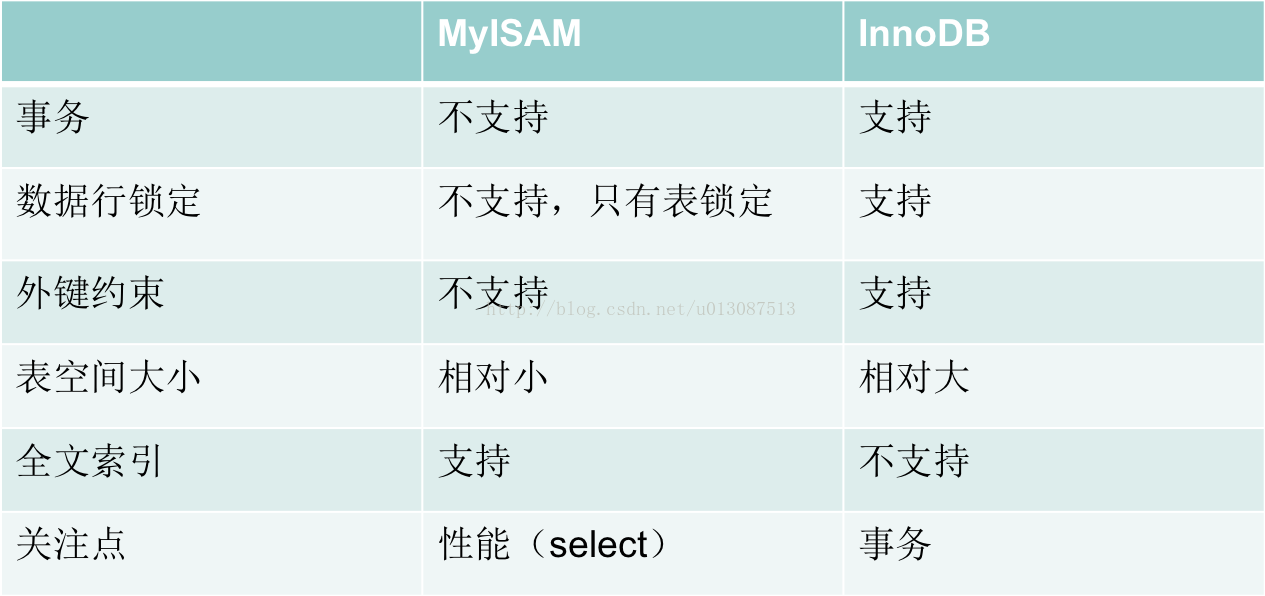
在MYSQL中有两个存储引擎MyISAM和InnoDB,每个引擎都有利有弊。

MyISAM适合于一些需要大量查询的应用，但是对于大量写操作的支持不是很好。甚至一个update语句就会进行锁表操作，这时读取这张表的所有进程都无法进行操作直至写操作完成。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。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。

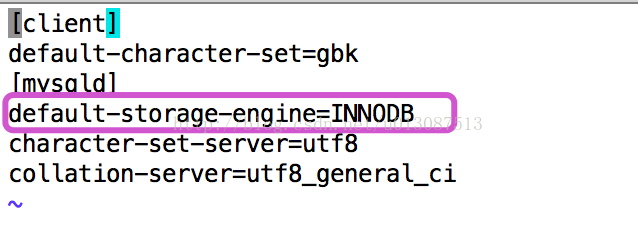
MyISAM是MYSQL5.5版本以前默认的存储引擎，基于传统的ISAM类型，支持B-Tree，全文检索，但是不是事务安全的，而且不支持外键。不具有原子性。支持锁表。

InnoDB是事务型引擎，支持ACID事务(实现4种事务隔离机制)、回滚、崩溃恢复能力、行锁。以及提供与Oracle一致的不加锁的读取方式。InnoDB存储它的表和索引在一个表空间中，表空间可以包含多个文件。

MyISAM和InnoDB比较，如下图所示：



对于Linux版本的MYSQL  配置文件在 /etc/my.cnf中



在5.5之后默认的存储引擎是INNODB

可以单独进行修改也可以在创建表时修改：

ALTER TABLE tab\_name ENGINE INNODB；

1. 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。

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。在我们决定要使用“永久链接”之前，我们需要好好地考虑一下我们的整个系统的架构。