

昵称: joy696163
园龄: 3年9个月
粉丝: 23
关注: 3
[+加关注](#)

<	2016年8月						>
日	一	二	三	四	五	六	
31	1	2	3	4	5	6	
7	8	9	10	11	12	13	
14	15	16	17	18	19	20	
21	22	23	24	25	26	27	
28	29	30	31	1	2	3	
4	5	6	7	8	9	10	

搜索

找找看

谷歌搜索

常用链接

[我的随笔](#)

[我的评论](#)

[我的参与](#)

[最新评论](#)

[我的标签](#)

[更多链接](#)

我的标签

[mysql\(3\)](#)

[php\(3\)](#)

[排序\(3\)](#)

[java\(2\)](#)

[js\(1\)](#)

[json\(1\)](#)

[linux\(1\)](#)

[preg_match、正则表达式\(1\)](#)

[查找\(1\)](#)

[恢复回收站图标\(1\)](#)

mysql 数据表读锁机制详解

为了给高并发情况下的mysql进行更好的优化，有必要了解一下mysql查询更新时的锁表机制。

一、概述

MySQL有三种锁的级别：页级、表级、行级。

MyISAM和MEMORY存储引擎采用的是表级锁（table-level locking）；BDB存储引擎采用的是页面锁（page-level locking），但也支持表级锁；InnoDB存储引擎既支持行级锁（row-level locking），也支持表级锁，但默认情况下是采用行级锁。

MySQL这3种锁的特性可大致归纳如下：

表级锁：开销小，加锁快；不会出现死锁；锁定粒度大，发生锁冲突的概率最高,并发度最低。

行级锁：开销大，加锁慢；会出现死锁；锁定粒度最小，发生锁冲突的概率最低,并发度也最高。

页面锁：开销和加锁时间界于表锁和行锁之间；会出现死锁；锁定粒度界于表锁和行锁之间，并发度一般。

二、MyISAM表锁

MyISAM存储引擎只支持表锁，是现在用得最多的存储引擎。

1、查询表级锁争用情况

可以通过检查table_locks_waited和table_locks_immediate状态变量来分析系统上的表锁定争夺：

mysql> show status like 'table%';

+	-----	+	-----	+
	Variable_name		Value	
+	-----	+	-----	+
	Table_locks_immediate		76939364	
	Table_locks_waited		305089	
+	-----	+	-----	+

2 rows in set (0.00 sec)Table_locks_waited的值比较高，说明存在着较严重的表级锁争用情况。

2、MySQL表级锁的锁模式

MySQL的表级锁有两种模式：表共享读锁（Table Read Lock）和表独占写锁（Table Write Lock）。MyISAM在执行查询语句（SELECT）前，会自动给涉及的所有表加读锁，在执行更新操作（UPDATE、DELETE、INSERT等）前，会自动给涉及的表加写锁。

所以对MyISAM表进行操作，会有以下情况：

a、对MyISAM表的读操作（加读锁），不会阻塞其他进程对同一表的读请求，但会阻塞对同一表的写请求。只有当读锁释放后，才会执行其它进程的写操作。

b、对MyISAM表的写操作（加写锁），会阻塞其他进程对同一表的读和写操作，只有当写锁释放后，才会执行其它进程的读写操作。

下面通过例子来进行验证以上观点。数据表gz_phone里有二百多万数据，字段id,phone,ua,day。现在同时用多个客户端同时对该表进行操作分析。

a、当我用客户端1进行一个比较长时间的读操作时，分别用客户端2进行读和写操作：

client1:

mysql>select count(*) from gz_phone group by ua;

75508 rows in set (3 min 15.87 sec) client2:

select id,phone from gz_phone limit 1000,10;

+	---	+	---	+
	id		phone	
+	---	+	---	+
	1001		2222	
	1002		2222	
	1003		2222	
	1004		2222	
	1005		2222	
	1006		2222	
	1007		2222	
	1008		2222	
	1009		2222	
	1010		2222	
+	---	+	---	+

更多
随笔分类
algorithm(2)
c语言(2)
english passages(1)
html css(1)
interview(1)
java(2)
JavaScript相关(15)
jQuery相关(2)
linux(20)
mysql相关(30)
php实例代码(28)
php相关(52)
安全防护(1)
待读(2)
计算互联其他(13)
建站经验(25)
面试准备(7)
小技巧(31)
原创(28)
职场、人生：不停的学习(3)
随笔档案
2016年7月 (1)
2016年5月 (1)
2016年3月 (2)
2016年2月 (1)
2015年12月 (2)
2015年11月 (6)
2015年9月 (8)
2015年8月 (2)

10 rows in set (0.01 sec)

```
mysql> update gz_phone set phone='1111111111' where id=1001;
Query OK, 0 rows affected (2 min 57.88 sec)
Rows matched: 1 Changed: 0 Warnings: 0
```

说明当数据表有一个读锁时，其它进程的查询操作可以马上执行，但更新操作需等待读锁释放后才会执行。

b、当用客户端1进行一个较长时间的更新操作时，用客户端2,3分别进行读写操作：

client1:

```
mysql> update gz_phone set phone='1111111111';
Query OK, 1671823 rows affected (3 min 4.03 sec)
Rows matched: 2212070 Changed: 1671823 Warnings: 0 client2:
mysql> select id,phone,ua,day from gz_phone limit 10;
+---+-----+-----+-----+
| id | phone | ua | day |
+---+-----+-----+-----+
| 1 | 2222 | SonyEricssonK310c | 2007-12-19 |
| 2 | 2222 | SonyEricssonK750c | 2007-12-19 |
| 3 | 2222 | MAUI WAP Browser | 2007-12-19 |
| 4 | 2222 | Nokia3108 | 2007-12-19 |
| 5 | 2222 | LENOVO-I750 | 2007-12-19 |
| 6 | 2222 | BIRD_D636 | 2007-12-19 |
| 7 | 2222 | SonyEricssonS500c | 2007-12-19 |
| 8 | 2222 | SAMSUNG-SGH-E258 | 2007-12-19 |
| 9 | 2222 | NokiaN73-1 | 2007-12-19 |
| 10 | 2222 | Nokia2610 | 2007-12-19 |
+---+-----+-----+-----+
10 rows in set (2 min 58.56 sec) client3:
mysql> update gz_phone set phone='55555' where id=1;
Query OK, 1 row affected (3 min 50.16 sec)
Rows matched: 1 Changed: 1 Warnings: 0
```

说明当数据表有一个写锁时，其它进程的读写操作都需等待读锁释放后才会执行。

3、并发插入

原则上数据表有一个读锁时，其它进程无法对此表进行更新操作，但在一定条件下，MyISAM表也支持查询和插入操作的并发进行。

MyISAM存储引擎有一个系统变量concurrent_insert，专门用以控制其并发插入的行为，其值分别可以为0、1或2。

a、当concurrent_insert设置为0时，不允许并发插入。

b、当concurrent_insert设置为1时，如果MyISAM表中没有空洞（即表的中间没有被删除的行），MyISAM允许在一个进程读表的同时，另一个进程从表尾插入记录。这也是MySQL的默认设置。

c、当concurrent_insert设置为2时，无论MyISAM表中有没有空洞，都允许在表尾并发插入记录。

4、MyISAM的锁调度

由于MySQL认为写请求一般比读请求要重要，所以如果有读写请求同时进行的话，MySQL将会优先执行写操作。这样MyISAM表在进行大量的更新操作时（特别是更新的字段中存在索引的情况下），会造成查询操作很难获得读锁，从而导致查询阻塞。

我们可以通过一些设置来调节MyISAM的调度行为：

a、通过指定启动参数low-priority-updates，使MyISAM引擎默认给予读请求以优先的权利。

b、通过执行命令SET LOW_PRIORITY_UPDATES=1，使该连接发出的更新请求优先级降低。

c、通过指定INSERT、UPDATE、DELETE语句的LOW_PRIORITY属性，降低该语句的优先级。

上面3种方法都是要么更新优先，要么查询优先的方法。这里要说明的就是，不要盲目的给mysql设置为读优先，因为一些需要长时间运行的查询操作，也会使写进程“饿死”。只有根据你的实际情况，来决定设置哪种操作优先。这些方法还是没有从根本上同时解决查询和更新的问题。

在一个有大数据量高并发表的mysql里，我们还可采用另一种策略来进行优化，那就是通过mysql主从（读写）分离来实现负载均衡，这样可避免优先哪一种操作从而可能导致另一种操作的堵塞。下面将用一个篇幅来说明mysql的读写分离技术。

1. MySQL锁表请求有两种方式：read锁和write锁

语法 lock tables t read/write 两者的共同点是当执行锁表后除当前进程外其他进程都无法访问该表除非发生下面三种情况之一：1.该进程执行解锁语句unlock tables 2.该进程执行其他锁表请求 3.该进程退出或断开与MySQL数据库连接；两者不同点是执行read锁的锁表进程只可对表查询不能修改数据，执行write锁的进程可以有增删改查所有权限可以理解为后者包含前者事实上也是后者的优先级比前者要高 通常我都是执行write锁的，下面举例也都以write为例

2. 进程执行lock tables t write锁表后，如果需要访问到表t1，MySQL会报错ERROR 1100: Table 't1' was not locked with LOCK TABLES

解决办法：进程一次对多表锁定，语法：lock tables t write,t1 write,... 解锁方法见1,unlock tables 只需执行一次即可

mysql锁和死锁

MyISAM和MEMORY存储引擎采用的是表级锁table-level locking

BDB存储引擎采用的是页面锁page-level locking,但也支持表级锁

2015年7月 (3)
2015年6月 (1)
2015年4月 (2)
2014年12月 (7)
2014年8月 (2)
2014年7月 (2)
2014年6月 (2)
2014年5月 (3)
2014年4月 (11)
2014年3月 (5)
2014年2月 (10)
2014年1月 (3)
2013年12月 (4)
2013年11月 (11)
2013年10月 (10)
2013年9月 (5)
2013年8月 (4)
2013年7月 (7)
2013年6月 (8)
2013年5月 (12)
2013年4月 (3)
2013年3月 (13)
2013年2月 (12)
2013年1月 (10)
2012年12月 (5)
2012年11月 (9)

文章分类
algorithm

最新评论
1. Re:24种设计模式介绍与6大设计原则（PHP版）之代理模式
你真赢了!!!

InnoDB存储引擎既支持行级锁row-level locking,也支持表级锁,但默认情况下是采用行级锁
表级锁 开销小,加锁快;不会出现死锁;锁定粒度大,发生锁冲突的概率最高,并发度最低
行级锁 开销大,加锁慢;会出现死锁;锁定粒度最小,发生锁冲突的概率最低,并发度也最高
页面锁 开销和加锁时间界于表锁和行锁之间;会出现死锁;锁定粒度界于表锁和行锁之间,并发度一般
仅从锁的角度来说:
表级锁更适合于以查询为主,只有少量按索引条件更新数据的应用,如Web应用
行级锁则更适合于有大量按索引条件并发更新少量不同数据,同时又有并发查询的应用,如一些在线事务处理系统

死锁
所谓死锁<DeadLock>:是指两个或两个以上的进程在执行过程中,
因争夺资源而造成的一种互相等待的现象,若无外力作用,它们都将无法推进下去。
此时称系统处于死锁状态或系统产生了死锁,这些永远在互相等待的进程称为死锁进程。
表级锁不会产生死锁.所以解决死锁主要还是真对于最常用的InnoDB。
在遇到问题时
先执行show processlist找到死锁线程号.然后Kill processNo
当然主要解决还是需要去看一下具体的操作.可能产生死锁
Show innodb status检查引擎状态 ,可以看到哪些语句产生死锁
然后就是解决了。
怎么解决还是要看具体什么问题。
MyISAM使用的是 flock 类的函数，直接就是对整个文件进行锁定（叫做文件锁定），InnoDB使用的是 fcntl 类的函数，可以对文件中局部数据进行锁定（叫做行锁定），所以区别就是在这里。
另外MyISAM的数据表是按照单个文件存储的，可以针对单个表文件进行锁定，但是InnoDB是一整个文件，把索引、数据、结构全部保存在 ibdata 文件里，所以必须用行锁定。

1、对于MySQL来说，有三种锁的级别：页级、表级、行级。
页级的典型代表引擎为BDB。
表级的典型代表引擎为MyISAM, MEMORY以及很久以前的ISAM。
行级的典型代表引擎为INNODB。
2、我们实际应用中用的最多的就是行锁了。

行级锁的优点如下：
1）、当很多连接分别进行不同的查询时减小LOCK状态。
2）、如果出现异常，可以减少数据的丢失。因为一次可以只回滚一行或者几行少量的数据。
行级锁的缺点如下：
1）、比页级锁和表级锁要占用更多的内存。
2）、进行查询时比页级锁和表级锁需要的I/O要多，所以我们经常把行级锁用在写操作而不是读操作。
3）、容易出现死锁。
3、MySQL用写队列和读队列来实现对数据库的写和读操作。

对于写锁定如下：
1）、如果表没有加锁，那么对其加写锁定。
2）、否则，那么把请求放入写锁队列中。
对于读锁定如下：
1）、如果表没有加写锁，那么加一个读锁。
2）、否则，那么把请求放到读锁队列中。
当然我们可以分别用low_priority 以及high_priority在写和读操作上来改变这些行为。

4、下面我来一个简单的例子解释上面的说法。

我们来运行一个时间很长的查询
1）、客户端1：
mysql> select count(*) from content group by content;
...
客户端2：
mysql> update content set content = 'I love you' where id = 444;
Query OK, 1 row affected (30.68 sec)
Rows matched: 1 Changed: 1 Warnings: 0
用了半分钟。
2）、我们现在终止客户端1。
此时客户端2：
mysql> update content set content = 'I hate you' where id = 444;
Query OK, 1 row affected (0.02 sec)
Rows matched: 1 Changed: 1 Warnings: 0
仅仅用了20毫秒。

--dcj890828
2. Re:Apache与Nginx的优缺点比较
13、Apache在处理动态有优势，Nginx并发性比较好，CPU内存占用低，如果rewrite频繁，那还是Apache吧rewrite Apache 是每次都要读取.htccess文件，而且每个文件.....
--礼物de绷带
3. Re:composer安装指定版本的laravel
现在都5.1版本了
--苏生不惑
4. Re:Apache与Nginx的优缺点比较
从哪拈来的？组织很混乱，就不能有点原创
--torresliang
5. Re:【整理】Virtualbox中的网络类型（NAT，桥接等），网卡，IP地址等方面的设置
图片挂了
--CoderDream

阅读排行榜
1. Apache与Nginx的优缺点比较(52941)
2. mysql添加索引命令(13725)
3. mysql 数据表读锁机制详解(11224)
4. 如何修改vsftpd的默认根目录/var/ftp/pub到另一个目录？(10416)
5. 不同浏览器存放cookie的路径是不一样的(9503)

评论排行榜
1. Apache与Nginx的优缺点比较(2)
2. 【整理】Virtualbox中的网络类型（NAT，桥接等），网卡，IP地址等方面的设置(1)
3. composer安装指定版本的laravel(1)
4. print(\$arr,true)的参数true表示

这个例子很好的说明了读写队列的运行。

对于1中的客户端1，此时表没有加锁，当然也没有加写锁了，那么此时客户端1对表加了一个读锁。

对于1中的客户端2，此时因为表有一个读锁，所以把UPDATE请求放到写锁定队列中。

当读锁释放的时候，也就是SHOW PROCESSLIST中STATUS 为COPY TO TMP TABLE的时候，UPDATE操作开始执行。

5、可以在REPLICATION中对MASTER 和SLAVE运用不同的锁定使系统达到最佳的性能。（当然这个前提是SQL语句都是最优的。）

通过锁机制，可以实现多线程同时对某个表进行操作。如下图所示，在某个时刻，用户甲、用户乙、用户丙可能会同时或者先后(前面一个作业还没有完成) 对数据表A进行查询或者更新的操作。当某个线程涉及到更新操作时，就需要获得独占的访问权。在更新的过程中，所有其它想要访问这个表的线程必须要等到其更新完成为止。此时就会导致锁竞争的问题。从而导致用户等待时间的延长。在这篇文章中，笔者将跟大家讨论，采取哪些措施可以有效的避免锁竞争，减少 MySQL用户的等待时间。

降低锁竞争 减少MySQL用户等待时间

背景模拟：

为了更加清楚的说明这个问题，笔者先模拟一个日常的案例。通过案例大家来阅读下面的内容，可能条理会更加的清晰。现在MySQL数据库遇到如上图所示这种情况。

首先，用户甲对数据表A发出了一个查询请求。

然后，用户乙又对数据表A发出了一个更新请求。此时用户乙的请求只有在用户甲的作业完成之后才能够得到执行。

最后，用户丙又对数据表A发出了一个查询请求。在MySQL数据库中，更新语句的优先级要比查询语句的优先级高，为此用户丙的查询语句只有在用户乙的更新作业完成之后才能够执行。而用户乙的更新作业又必须在用户甲的查询语句完成之后才能够执行。此时就存在比较严重的锁竞争问题。

现在数据库工程师所要做的就是数据库设计与优化过程中，采取哪些措施来降低这种锁竞争的不利情况？

措施一：利用Lock Tables来提高更新速度

对于更新作业来说，在一个锁定中进行许多更新要比所有锁定的更新要来得快。为此如果一个表更新频率比较高，如超市的收银系统，那么可以通过使用Lock Tables选项来提高更新速度。更新的速度提高了，那么与Select查询作业的冲突就会明显减少，锁竞争的现象也能够得到明显的抑制。

措施二：将某个表分为几个表来降低锁竞争

如一个大型的购物超市，如沃尔玛，其销售纪录表每天的更新操作非常的多。此时如果用户在更新的同时，另外有用户需要对其进行查询，显然锁竞争的现象会比较严重。针对这种情况，其实可以人为的将某张表分为几个表。如可以为每一台收银机专门设置一张数据表。如此的话，各台收银机之间用户的操作都是在自己的表中完成，相互之间不会产生干扰。在数据统计分析时，可以通过视图将他们整合成一张表。

措施三：调整某个作业的优先级

默认情况下，在MySQL数据库中，更新操作比Select查询有更高的优先级。如上图所示，如果用户乙先发出了一个查询申请，然后用户丙再发出一个更新请求。当用户甲的查询作业完成之后，系统会先执行谁的请求呢？注意，默认情况下系统并不遵循先来后到的规则，即不会先执行用户乙的查询请求，而是执行用户丙的更新进程。这主要是因为，更新进程比查询进程具有更高的优先级。

但是在有些特定的情况下，可能这种优先级不符合企业的需求。此时数据库管理员需要根据实际情况来调整语句的优先级。如果确实需要的话，那么可以通过以下三种方式来实现。

一是通过LOW_PRIORITY属性。这个属性可以将某个特定的语句的优先级降低。如可以调低某个特定的更新语句或者插入语句的优先级。不过需要注意的是，这个属性只有对特定的语句有用。即其作用域只针对某个特定的语句，而不会对全局造成影响。

二是通过HIGH_PRIORITY属性。与通过LOW_PRIORITY属性对应，有一个HIGH_PRIORITY属性。顾名思义，这个属性可以用来提高某个特定的Select查询语句的优先级。如上面这个案例，在用户丙的查询语句中加入HIGH_PRIORITY属性的话，那么用户甲查询完毕之后，会立即执行用户丙的查询语句。等到用户丙执行完毕之后，才会执行用户乙的更新操作。可见，此时查询语句的优先级得到了提升。这里需要注意，跟上面这个属性一样，这个作用域也只限于特定的查询语句。而不会对没有加这个

将\$arr的值返会，而不是打印(1)

5. 24种设计模式介绍与6大设计原则（PHP版）之代理模式(1)

推荐排行榜

1. Apache与Nginx的优缺点比较(2)

2. mysql 数据表读锁机制详解(1)

3. 随心所欲玩复制 详解robocopy(1)

4. 不同浏览器存放cookie的路径是不一样的(1)

5. 新浪网易IP地区信息查询API接口调用方法(1)

参数的其他查询语句产生影响。也就是说，其他查询语句如果没有加这个属性，那么其优先级仍然低于更新进程。

三是通过Set LOW_PRIORIT_UPDATES=1选项。以上两个属性都是针对特定的语句，而不会造成全局的影响。如果现在数据库管理员需要对某个连接来调整优先级，该如何实现呢?如上例，现在用户需要将用户丙连接的查询语句的优先级别提高，而不是每次查询时都需要使用上面的属性。此时就需要使用Set LOW_PRIORIT_UPDATES=1选项。通过这个选项可以制定具体连接中的所有更新进程都是用比较低的优先级。注意这个选项只针对特定的连接有用。对于其他的连接，就不适用。

四是采用Low_Priority_updates选项。上面谈到的属性，前面两个针对特定的语句，后面一个是针对特定的连接，都不会对整个数据库产生影响。如果现在需要在整个数据库范围之内，降低更新语句的优先级，是否可以实现?如上面这个案例，在不使用其他参数的情况下，就让用户丙的查询语句比用户乙的更新具有更先执行?如果用户有这种需求的话，可以使用 Low_Priority_updates选项来启动数据库。采用这个选项启动数据库时，系统会给数据库中所有的更新语句比较低的优先级。此时用户丙的查询语句就会比用户用户乙的更新请求更早的执行。而对于查询作业来说，不存在锁定的情况。为此用户甲的查询请求与用户丙的查询请求可以同时进行。为此通过调整语句执行的优先级，可以有效的降低锁竞争的情况。

可见，可以利用属性或者选项来调整某条语句的优先级。如现在有一个应用，主要供用户来进行查询。更新的操作一般都是有管理员来完成，并且对于用户来说更新的数据并不敏感。此时基于用户优先的原则，可以考虑将查询的优先级别提高。如此的话，对于用户来说，其遇到锁竞争的情况就会比较少，从而可以缩短用户的等待时间。在调整用户优先级时，需要考虑其调整的范围。即只是调整特定的语句、还是调整特定的连接，又或者对整个数据库生效。

措施四：对于混合操作的情况，可以采用特定的选项

有时候会遇到混合操作的作业，如即有更新操作又有插入操作又有查询操作时，要根据特定的情况，采用特定的选项。如现在需要对数据表同时进行插入和删除的作业，此时如果能够使用Insert Delayed选项，将会给用户带来很大的帮助。再如同一个数据表执行Select和Delete语句会有锁竞争的情况。此时数据库管理员也可以根据实际情况来选择使用Delete Limint选项来解决所遇到速度问题。

通常情况下，锁竞争与死锁不同，并不会对数据库的运行带来很大的影响。只是可能会延长用户的等待时间。如果用户并发访问的机率并不是很高，此时锁竞争的现象就会很少。那么采用上面的这些措施并不会带来多大的收益。相反，如果用户对某个表的并发访问比较多，特别是不同的用户会对表执行查询、更新、删除、插入等混合作业，那么采取上面这些措施可以在很大程度上降低锁冲突，减少用户的等待时间。

感觉我们技术部的总结会，每回都 能学到很多新知识，希望以后多有分享技术心得的环节。其实分享往往就寥寥几句，有的时候听完了，可能会 不知道 原理、实现的过程，甚至只能听明白一小部分。但这 都没关系，重要的是，知道有那么回事，用到的时候会少走很多弯路。关于上次总结会 显功提出的“双表并发”，猜想原理如下：

在分发注册密码的web交互程序中，普通的，不考虑并发的设计如下：

设计一个表：一个字段是id，一个字段是需要分发的密码，另一个是标志位，标志该密码是否已经分发，初始是0；

程序设计：从表里找到一个标志位为0的密码，设置该标志位1，然后发给用户。

但这个方法问题是当用户的访问是高并发的时候，多个用户会得到相同的密码，

原因是(猜想仅供参考)：

mysql的数据库操作方式是类似操作系统的读写锁，就是允许多个读锁同时操作，此时是不允许写的，当读锁放开的时候允许写，同理当写锁起作用的时候，读锁是阻塞的。所以，当用户高并发的的时候，多个读锁可以一起读，第一个读锁释放后，它要将标志位置为1，但由于有其它读锁在读，所以第一个操作的写锁阻塞在这里，不能够将刚读到的这一行的标志字段，及时设置为1。并发的其他读锁读到的标志位还是0，当所有的并发读锁都释放后，所有操作的写锁开始起作用，多个并发的写操作阻塞执行，依次将该位置为1。这样多个并发的操作读的都是一条数据。

解决这个问题的方法是，仍旧利用mysql的读写锁的机制，对于这种机制，写锁一定互斥的，虽然允许同时多个读操作，但永远只允许一个写操作。刚才的问题是多个读数据的操作并发执行造成的，要避免这个，需要对读取的时候也加锁，不允许并发读取。我不知道mysql有没有这方面的设置直接来实现，但可以通过如下取巧的方式解决。

由于在写入的时候锁是互斥的，所以再建立一个表，只保存一个字段即可，就是一个自增的id，当

有操作需要申请密码的时候，先在这个表里插入一条空数据，这样返回一个mysql分配的自增的id，用这个id去第一个表里取相应id的密码就可以了。

不会出现 多个用户得到同样密码的 解释是，此时多个并发的操作肯定可以得到不同的id，因为在插入的时候写锁是互斥的，并发的多个操作要想写数据库，就会阻塞排队，第一个操作写入后，释放了该锁，获得 mysql分配的id，其后的操作需要执行insert操作，mysql就会将这个操作顺序插入数据库的不同行，返回不同的id，此时虽然操作是并发的，同时到达的，但对于mysql来说，是一条一条执行插入语句，所以当然操作的是不同的行，返回不同的id，这样在第一个表里找到的就是不同的密码，用户分配到的也是不同的密码。

当这个分配的id大于密码表里的id总数时候，表示密码全部发送完。

这个原理可以用于其它需要防止并发读脏数据的程序，mysql的读写锁可以帮助我们来做其它的互斥程序。那天听到吴浩说memcached也有类似的读写锁，不知道如果用那个是不是可以提高并发效率，要比操作mysql快。

LOCK TABLES

tbl_name [AS alias] {READ [LOCAL] | [LOW_PRIORITY] WRITE}

[, tbl_name [AS alias] {READ [LOCAL] | [LOW_PRIORITY] WRITE}] ...

UNLOCK TABLES

LOCK TABLES可以锁定用于当前线程的表。如果表被其它线程锁定，则造成堵塞，直到可以获取所有锁定为止。UNLOCK TABLES可以释放被当前线程保持的任何锁定。当线程发布另一个LOCK TABLES时，或与服务器的连接被关闭时，所有由当前线程锁定的表被隐含地解锁。

表锁定只用于防止其它客户端进行不正当地读取和写入。保持锁定（即使是读取锁定）的客户端可以进行表层级的操作，比如DROP TABLE。

注意，下面是对事务表使用LOCK TABLES的说明：

- 在尝试锁定表之前，LOCK TABLES不是事务安全型的，会隐含地提交所有活性事务。同时，开始一项事务（例如，使用START TRANSACTION），会隐含地执行UNLOCK TABLES
- 对事务表（如InnoDB）使用LOCK TABLES的正确方法是，设置AUTOCOMMIT=0并且不能调用UNLOCK TABLES，直到您明确地提交事务为止。当您调用LOCK TABLES时，InnoDB会内部地取其自己的表锁定，MySQL取其自己的表锁定。InnoDB在下一个提交时释放其表锁定，但是，对于MySQL，要释放表锁定，您必须调用UNLOCK TABLES。您不应该让AUTOCOMMIT=1，因为那样的话，InnoDB会在调用LOCK TABLES之后立刻释放表锁定，并且很容易形成死锁定。注意，如果AUTOCOMMIT=1，我们根本不能获取InnoDB表锁定，这样就可以帮助旧的应用软件避免不必要的死锁定。
- ROLLBACK不会释放MySQL的非事务表锁定。

要使用LOCK TABLES，您必须拥有相关表的LOCK TABLES权限和SELECT权限。

使用LOCK TABLES的主要原因是仿效事务，或在更新表时加快速度。这将在后面进行更详细的解释。

如果一个线程获得对一个表地READ锁定，该线程（和所有其它线程）只能从该表中读取。如果一个线程获得对一个表的WRITE锁定，只有保持锁定的线程可以对表进行写入。其它的线程被阻止，直到锁定被释放时为止。

READ LOCAL和READ之间的区别是，READ LOCAL允许在锁定被保持时，执行非冲突性INSERT语句（同时插入）。但是，如果您正打算在MySQL外面操作数据库文件，同时您保持锁定，则不能使用READ LOCAL。对于InnoDB表，READ LOCAL与READ相同。

当您使用LOCK TABLES时，您必须锁定您打算在查询中使用的所有的表。虽然使用LOCK TABLES语句获得的锁定仍然有效，但是您不能访问没有被此语句锁定的任何的表。同时，您不能在一次查询中多次使用一个已锁定的表——使用别名代替，在此情况下，您必须分别获得对每个别名的锁定。

```
mysql> LOCK TABLE t WRITE, t AS t1 WRITE;
```

```
mysql> INSERT INTO t SELECT * FROM t;
```

ERROR 1100: Table 't' was not locked with LOCK TABLES

```
mysql> INSERT INTO t SELECT * FROM t AS t1;
```

如果您的查询使用一个别名引用一个表，那么您必须使用同样的别名锁定该表。如果没有指定别名，则不会锁定该表。

```
mysql> LOCK TABLE t READ;
```

```
mysql> SELECT * FROM t AS myalias;
```

ERROR 1100: Table 'myalias' was not locked with LOCK TABLES

相反的，如果您使用一个别名锁定一个表，您必须使用该别名在您的查询中引用该表。

```
mysql> LOCK TABLE t AS myalias READ;
```

```
mysql> SELECT * FROM t;
```

ERROR 1100: Table 't' was not locked with LOCK TABLES

```
mysql> SELECT * FROM t AS myalias;
```

WRITE锁定通常比**READ**锁定拥有更高的优先权，以确保更新被尽快地处理。这意味着，如果一个线程获得了一个**READ**锁定，则另一个线程会申请一个**WRITE**锁定，后续的**READ**锁定申请会等待，直到**WRITE**线程获得锁定并释放锁定。您可以使用**LOW_PRIORITY WRITE**锁定来允许其它线程在该线程正在等待**WRITE**锁定时获得**READ**锁定。只有当您确定最终将有一个时机，此时没有线程拥有**READ**锁定时，您才应该使用**LOW_PRIORITY WRITE**锁定。

LOCK TABLES按照如下方式执行：

1. 按照内部定义的顺序，对所有要被锁定的表进行分类。从用户的角度，此顺序是未经定义的。
2. 如果使用一个读取和一个写入锁定对一个表进行锁定，则把写入锁定放在读取锁定之前。
3. 一次锁定一个表，直到线程得到所有锁定为止。

该规则确保表锁定不会出现死锁定。但是，对于该规则，您需要注意其它的事情：

如果您正在对一个表使用一个**LOW_PRIORITY WRITE**锁定，这并不意味着，**MySQL**等待特定的锁定，直到没有申请**READ**锁定的线程时为止。当线程已经获得**WRITE**锁定，并正在等待得到锁定表清单中的用于下一个表的锁定时，所有其它线程会等待**WRITE**锁定被释放。如果这成为对于应用程序的严重的问题，则您应该考虑把部分表转化为事务安全型表。

您可以安全地使用**KILL**来结束一个正在等待表锁定的线程。

注意，您不能使用**INSERT DELAYED**锁定任何您正在使用的表，因为，在这种情况下，**INSERT**由另一个线程执行。

通常，您不需要锁定表，因为所有的单个**UPDATE**语句都是原子性的；没有其它的线程可以干扰任何其它当前正在执行的**SQL**语句。但是，在几种情况下，锁定表会有好处：

- 如果您正在对一组**MyISAM**表运行许多操作，锁定您正在使用的表，可以快很多。锁定**MyISAM**表可以加快插入、更新或删除的速度。不利方面是，没有线程可以更新一个用**READ**锁定的表（包括保持锁定的表），也没有线程可以访问用**WRITE**锁定的表（除了保持锁定的表以外）。

有些**MyISAM**操作在**LOCK TABLES**之下更快的原因是，**MySQL**不会清空用于已锁定表的关键缓存，直到**UNLOCK TABLE**被调用为止。通常，关键缓存在每个**SQL**语句之后被清空。

- 如果您正在使用**MySQL**中的一个不支持事务的存储引擎，则如果您想要确定在**SELECT**和**UPDATE**之间没有其它线程，您必须使用**LOCK TABLES**。本处所示的例子要求**LOCK TABLES**，以便安全地执行：

```
mysql> LOCK TABLES trans READ, customer WRITE;
```

```
mysql> SELECT SUM(value) FROM trans WHERE customer_id=some_id;

mysql> UPDATE customer

-> SET total_value=sum_from_previous_statement

-> WHERE customer_id=some_id;

mysql> UNLOCK TABLES;
```

如果没有LOCK TABLES，有可能另一个线程会在执行SELECT和UPDATE语句之间在trans表中插入一个新行。

通过使用相对更新（UPDATE customer SET value=value+new_value）或LAST_INSERT_ID()函数，您可以在许多情况下避免使用LOCK TABLES。

通过使用用户层级的顾问式锁定函数GET_LOCK()和RELEASE_LOCK()，您也可以在有些情况下避免锁定表。这些锁定被保存在服务器中的一个混编表中，使用pthread_mutex_lock() 和 pthread_mutex_unlock()，以加快速度。


要了解更多有关锁定规则的说明

您可以使用FLUSH TABLES WITH READ LOCK语句锁定位于所有带有读取锁定的数据库中的所有表。如果您有一个可以及时拍摄快照的文件系统，比如Veritas，这是获得备份的一个非常方便的方式。

注释：如果您对一个已锁定的表使用ALTER TABLE，该表可能会解锁。

分类: [mysql相关](#)



 [joy696163](#)
关注 - 3
粉丝 - 23
[+加关注](#)

1 0

(请您对文章做出评价)

« 上一篇: [大学毕业第一份工作重要吗? 这篇文章读透你就明白了](#)
» 下一篇: [IP地址转、整数互相转换](#)

posted @ 2013-07-06 00:21 joy696163 阅读(11224) 评论(0) 编辑 收藏

[刷新评论](#) [刷新页面](#) [返回顶部](#)

注册用户登录后才能发表评论，请 [登录](#) 或 [注册](#)，[访问](#)网站首页。

最新IT新闻：

- IBM研发的“芯片”可以过滤血液，预测癌症
 - 日本展出有“自主神经网络”的人形机器人
 - 新型钻石涂层屏幕比康宁大猩猩5薄800倍 强度优于蓝宝石
 - Galaxy Note 7真机曝光：美购买者将可能免费获得Gear Fit 2
 - Android现可通知审查用户账号登录活动
- » 更多新闻...

最新知识库文章：

- 可是姑娘，你为什么要编程呢？
 - 知其所以然（以算法学习为例）
 - 如何给变量取个简短且无歧义的名字
 - 编程的智慧
 - 写给初学前端工程师的一封信
- » 更多知识库文章...