**再送一波干货，测试2000线程并发下同时查询1000万条数据库表及索引优化**

继上篇文章《 [绝对干货，教你4分钟插入1000万条数据到mysql数据库表，快快进来](http://www.cnblogs.com/fanwencong/p/5765136.html) 》发布后在博客园首页展示得到了挺多的阅读量，我这篇文章就是对上篇文章的千万级数据库表在高并发访问下如何进行测试访问

这篇文章的知识点如下:

1.如何自写几十行代码就能模拟测试高并发下访问千万级数据库表

2.比较高并发下（200次/秒,2000次/秒,10000次/秒）数据库的性能

3.比较千万级数据库在查询时加索引与不加索引的巨大差异（说实话，这个测试结果让我自己本人也很惊讶）

针对上篇文章插入的1000万条数据到数据库后，我们进行了高并发下测试（模拟教师输入姓名和密码在1秒内登录数据库），线程类代码如下

package insert;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

public class ThreadToMysql extends Thread {

public String teacherName;

public String password;

public ThreadToMysql(String teacherName, String password) {//构造函数传入要查询登录的老师姓名和密码

this.teacherName=teacherName;

this.password=password;

}

public void run() {

String url = "jdbc:mysql://127.0.0.1/teacher";

String name = "com.mysql.jdbc.Driver";

String user = "root";

String password = "123456";

Connection conn = null;

try {

Class.forName(name);

conn = DriverManager.getConnection(url, user, password);//获取连接

conn.setAutoCommit(false);//关闭自动提交，不然conn.commit()运行到这句会报错

} catch (ClassNotFoundException e1) {

e1.printStackTrace();

} catch (SQLException e) {

e.printStackTrace();

}

if (conn!=null) {

Long startTime=System.currentTimeMillis();//开始时间

String sql="select id from t\_teacher where t\_name='"+teacherName+"' and t\_password='"+password+"'";//SQL语句

String id=null;

try {

Statement stmt=conn.createStatement();

ResultSet rs=stmt.executeQuery(sql);//获取结果集

if (rs.next()) {

id=rs.getString("id");

}

conn.commit();

stmt.close();

conn.close();

} catch (SQLException e) {

e.printStackTrace();

}

Long end=System.currentTimeMillis();

System.out.println(currentThread().getName()+" 查询结果:"+id+" 开始时间:"+startTime+" 结束时间:"+end+" 用时:"+(end-startTime)+"ms");

} else {

System.out.println(currentThread().getName()+"数据库连接失败:");

}

}

}

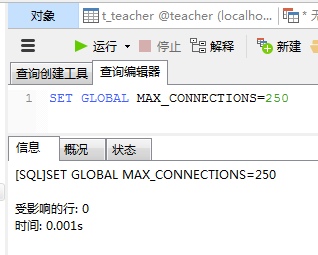
测试类代码如下：

|  |
| --- |
| package insert;  public class TestThreadToMysql {  public static void main(String[] args) {  for (int i = 1; i <=2000; i++) {  String teacherName=String.valueOf(i);  new ThreadToMysql(teacherName, "123456").start();    } |

## 一.在没有加索引的情况下测试：

把数据库的最大连接数设置为250：

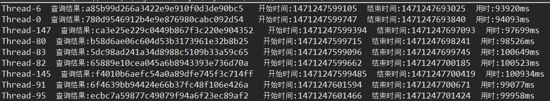
|  |
| --- |
| set global max\_connections =250 |



测试代码：

|  |
| --- |
| package insert;  public class TestThreadToMysql {  public static void main(String[] args) {  for (int i = 1; i <=200; i++) {  String teacherName=String.valueOf(i);  new ThreadToMysql(teacherName, "123456").start();  }  }  } |

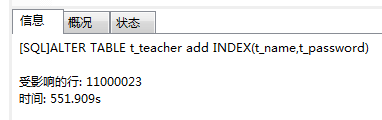
测试结果：



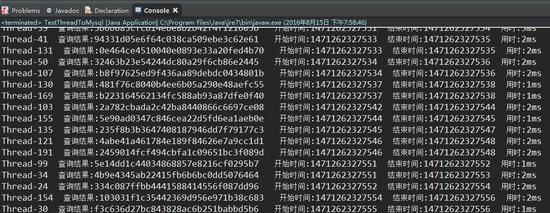
100多秒啊。。。我的天，这用户体验也没准了O(∩\_∩)O哈哈~

## 二.加索引后再次进行高并发下测试：

数据库加索引SQL语句如下：这里我有一个疑问，上个星期我加索引等了半个小时我都没加完索引我就停止了，今天下午居然只用了551秒就加完了索引。。。搞不懂

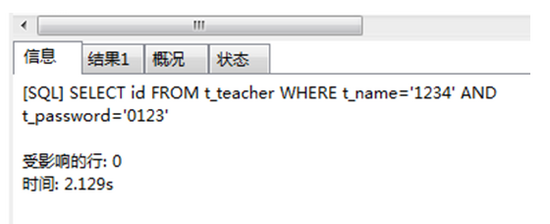


clean下项目代码后再次运行（尽量经常clean下项目去掉缓存，不然结果会有出入）：

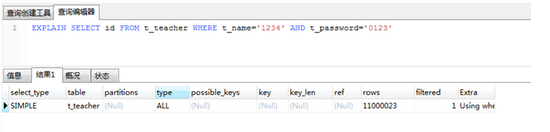


看到这个结果有没有被惊呆啊？哈哈加了索引由100多秒提升到1~2毫秒，查询速度提示1万多倍，查询性能得到大幅度变态级提升~~~

没加索引之前我查询单个记录都要2秒多

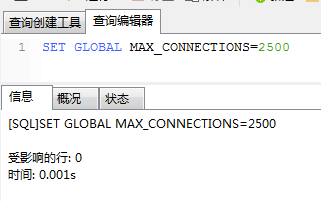


用explain查看语句可以知道要扫描全表，性能当然大幅度下降



下面我们来挑战2000线程同时并发访问查询数据库。看看结果:

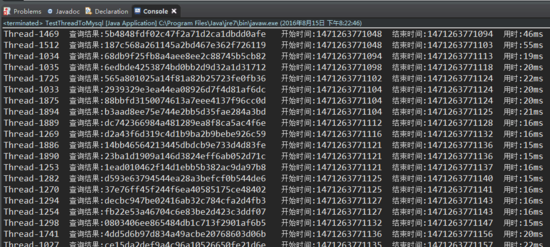
### 把数据库最大连接数设置为2500



### 测试代码改为2000

|  |
| --- |
| package insert;  public class TestThreadToMysql {  public static void main(String[] args) {  for (int i = 1; i <=2000; i++) {  String teacherName=String.valueOf(i);  new ThreadToMysql(teacherName, "123456").start();  }  }  } |

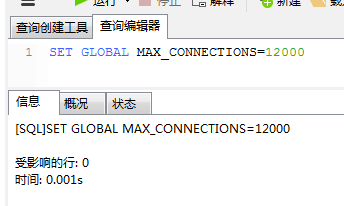
### 结果截图：



性能没问题，平均几十毫秒，很满意

### 下面我们来挑战一下1万个线程同时高并发访问，大家可以先想想结果会怎么样，哈哈

### 设置数据库最大连接数12000

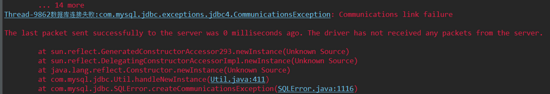


测试代码改为10000（再次提示。clean一下项目去掉缓存，这样结果更准确）

|  |
| --- |
| package insert;  public class TestThreadToMysql {  public static void main(String[] args) {  for (int i = 1; i <=10000; i++) {  String teacherName=String.valueOf(i);  new ThreadToMysql(teacherName, "123456").start();  }  }  } |

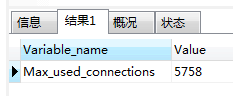
结果如下（运行后发现电脑有点卡）：





结果出现两种报错，1.连接请求被拒绝 2.连接失效 3.不过也有一部分成功连接上并且正确运行

然后我在数据库查看最大连接响应数：



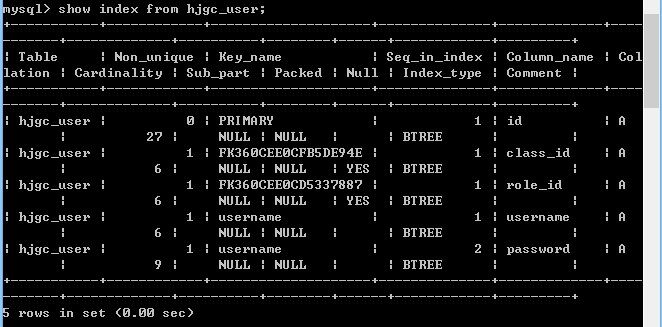
可以看出来就算你的数据库设置为再高你的数据库服务器也响应不过来。。。。顶多响应5758个

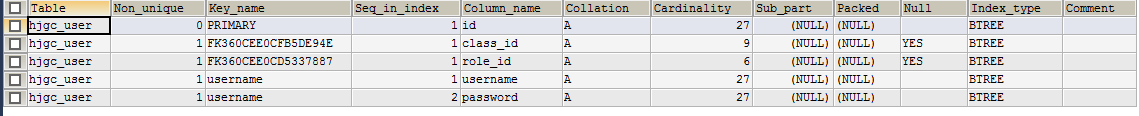
小小总结，1.可以自己测试高并发下挑战数据库性能，2. 对索引在查询性能上的强大有一个大概认识  很适合初学者学习了解

# 查看Mysql的索引信息

## 第一种方法

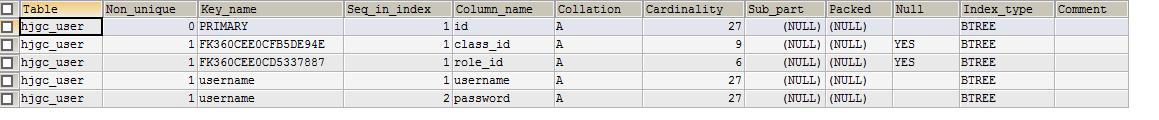
|  |
| --- |
| SHOW INDEX FROM hjgc\_user; |





## 第二种方法

|  |
| --- |
| SHOW KEYS FROM hjgc\_user; |



## 字段明细

· Table

表的名称。

· Non\_unique

如果索引不能包括重复词，则为0。如果可以，则为1。

· Key\_name

索引的名称。

· Seq\_in\_index

索引中的列序列号，从1开始。

· Column\_name

列名称。

· Collation

列以什么方式存储在索引中。在MySQL中，有值‘A’（升序）或NULL（无分类）。

· Cardinality

索引中唯一值的数目的估计值。通过运行ANALYZE TABLE或myisamchk -a可以更新。基数根据被存储为整数的统计数据来计数，所以即使对于小型表，该值也没有必要是精确的。基数越大，当进行联合时，MySQL使用该索引的机 会就越大。

· Sub\_part

如果列只是被部分地编入索引，则为被编入索引的字符的数目。如果整列被编入索引，则为NULL。

· Packed

指示关键字如何被压缩。如果没有被压缩，则为NULL。

· Null

如果列含有NULL，则含有YES。如果没有，则该列含有NO。

· Index\_type

用过的索引方法（BTREE, FULLTEXT, HASH, RTREE）。

· Comment

# 查看连接数

|  |
| --- |
| show processlist; |

