《数据库概论》实验三 使用高级程序设计语言访问数据库 实验报告

吴政亿 151220129 18805156360@163.com

实验环境

Windows 10 ，mysql-community-5.7.20.0 ，JAVA jdk9.0.1 ， intellij

实验过程

1. **实验截图**

应用JDBC连接到数据库并且执行相关语句：

executeSql("CREATE TABLE Staff\n" +

"(\n" +

" SNAME char(8) NOT NULL,\n" +

" SNO int NOT NULL,\n" +

" AGE int DEFAULT 0, -- 默认值约束\n" +

" SALARY int,\n" +

" DNO int, \n" +

" PRIMARY KEY (SNO)\n" +

");");

executeSql("INSERT INTO Staff\n" +

"VALUES \n" +

"('谢敏辉',131,21,-5,1),\n" +

"('吴政亿',129,20,1000,2),\n" +

"('吴玉明',130,21,100,3);");

executeSql("INSERT INTO Staff(SNAME,SNO,AGE,DNO)\n" +

"VALUES\n" +

"('吴一楠',128,21,3),\n" +

"('许丽君',133,21,2);");

executeSql("DELETE FROM Staff WHERE SNO = 128;");

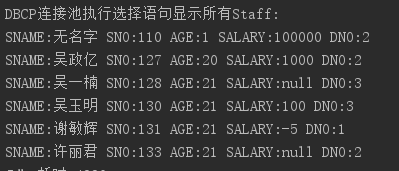
executeSql("INSERT INTO Staff(SNAME,SNO,AGE,DNO) VALUES ('吴一楠',128,21,3);");

executeSql("INSERT INTO Staff VALUES ('无名字',110,1,100000,2);");

executeSql("update Staff set SNO = 127 where SNO=129;");

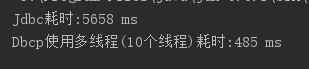






JDBC每次重获连接，执行查询语句，关闭连接耗时：5658ms

DBCP每次从池中获得连接，执行查询语句，关闭连接耗时：485ms



JDBC用一条连接，执行2000条查询语句耗时：999ms

DBCP用一条从池中获得的连接，执行2000条查询语句耗时：660ms



DBCP最大池连接数等于8时（默认值）耗时为：485ms

DBCP最大池连接数等于6时（默认值）耗时为：548ms



DBCP应用多线程（8个线程）的时候耗时为：500ms



1. **代码展示**

**JDBC.java**

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

//import entity.UserInfo;

public class JDBC {

Connection conn;

PreparedStatement ps;

ResultSet rs;

private Statement statement;

/\*\*

\* connection database

\*/

public Connection getConnection(){

String url="jdbc:mysql://localhost:3306/lab3?useSSL=false";

String ssl = "useSSL=false";

String userName="root";

String password="wuzhengyi";

try {

Class.forName("com.mysql.jdbc.Driver");

} catch (ClassNotFoundException e) {

// TODO Auto-generated catch block

System.out.println("找不到驱动！");

e.printStackTrace();

}

try {

conn=DriverManager.getConnection(url,userName, password);

if(conn!=null){

// System.out.println("connection successful");

statement = conn.createStatement(); //create Statement

}

} catch (SQLException e) {

// TODO Auto-generated catch block

System.out.println( "connection fail");

e.printStackTrace();

}

return conn;

}

public void closeConnection(){

try {

conn.close();

}catch(SQLException e) {

e.printStackTrace();

}

}

public ResultSet querySql(String sql) {

ResultSet result = null;

try

{

result = statement.executeQuery(sql);

} catch (SQLException e)

{

// TODO Auto-generated catch block

e.printStackTrace();

}

return result;

}

public static void printStaff(ResultSet result) {

try

{

while(result.next()) {

System.out.println("SNAME:" + result.getString(1)

+ " SNO:" + result.getString(2)

+ " AGE:" + result.getString(3)

+ " SALARY:" + result.getString(4)

+ " DNO:" + result.getString(5)

);

}

} catch (SQLException e)

{

// TODO Auto-generated catch block

e.printStackTrace();

}

}

public void executeSql(String sql) {

try

{

statement.execute(sql);

} catch (SQLException e)

{

// TODO Auto-generated catch block

e.printStackTrace();

}

}

public void testJDBC(){

System.out.println("\n Create Staff Table:");

executeSql("CREATE TABLE Staff\n" +

"(\n" +

" SNAME char(8) NOT NULL,\n" +

" SNO int NOT NULL,\n" +

" AGE int DEFAULT 0, -- 默认值约束\n" +

" SALARY int,\n" +

" DNO int, -- FOREIGN KEY REFERENCES Department(DNO) --外键约束，之后再加\n" +

" PRIMARY KEY (SNO)\n" +

");");

String sql = "SELECT \* FROM Staff;";

ResultSet result = querySql(sql);//在控制台顯示出查找方法

printStaff(result);

System.out.println("\n Insert to Staff Table:");

executeSql("INSERT INTO Staff\n" +

"VALUES \n" +

"('谢敏辉',131,21,-5,1),\n" +

"('吴政亿',129,20,1000,2),\n" +

"('吴玉明',130,21,100,3);");

sql = "SELECT \* FROM Staff;";

result = querySql(sql);//在控制台顯示出查找方法

printStaff(result);

System.out.println("\n向Staff表中插入数据:");

executeSql("INSERT INTO Staff(SNAME,SNO,AGE,DNO)\n" +

"VALUES\n" +

"('吴一楠',128,21,3),\n" +

"('许丽君',133,21,2);");

sql = "SELECT \* FROM Staff;";

result = querySql(sql);//在控制台顯示出查找方法

printStaff(result);

System.out.println("\n显示Staff表所有数据:");

sql = "SELECT \* FROM Staff;";

result = querySql(sql);//在控制台顯示出查找方法

printStaff(result);

System.out.println("\n删除SNO为128的吴一楠:");

executeSql("DELETE FROM Staff WHERE SNO = 128;");

result = querySql(sql);

printStaff(result);

System.out.println("\n添加('吴一楠',128,21,3):");

executeSql("INSERT INTO Staff(SNAME,SNO,AGE,DNO) VALUES ('吴一楠',128,21,3);");

result = querySql(sql);

printStaff(result);

System.out.println("\n添加('无名字',110,1,100000,2):");

executeSql("INSERT INTO Staff VALUES ('无名字',110,1,100000,2);");

result = querySql(sql);

printStaff(result);

System.out.println("\n将SNO=129的改为127:");

executeSql("update Staff set SNO = 127 where SNO=129;");

result = querySql(sql);

printStaff(result);

System.out.println();

}

// public static void main(String[] args) {

// JDBC j=new JDBC();

// j.getConnection();

// j.teztJDBC();

//

// try{

// DBCP d = new DBCP();

// d.getParameter();

// d.testDBCP();

// }

// catch (SQLException e)

// {

// e.printStackTrace();

// }

// }

}

**DBCP.java**

import java.sql.Connection;

import java.sql.SQLException;

import org.apache.commons.dbcp2.BasicDataSource;

import java.sql.Statement;

import java.sql.ResultSet;

public class DBCP {

private Connection conn = null;

protected Statement stmt;

private BasicDataSource pool;

public DBCP(){

initDBCP();

}

public void getParameter(){

System.out.println("最大空闲时间: " + pool.getMaxIdle());// 最大空闲时间。如果一个用户获取一个连接，不用，多长时间会被强行收回

System.out.println("持等待回收时间: " + pool.getMaxWaitMillis());// 在抛出异常之前,池等待连接被回收的最长时间（当没有可用连接时）。设置为-1表示无限等待。

System.out.println("初始化Connection数: " + pool.getInitialSize());// 初始化时有几个Connection

System.out.println("最大Connection数: " + pool.getMaxTotal());// 最多能有多少个Connection

System.out.println("----------------");

}

public ResultSet testDBCP(){

ResultSet resultSet = null;

synchronized (stmt){

try {

while(stmt.isClosed())

getConnection();

resultSet = stmt.executeQuery("select \* from staff;");//执行查询

}

catch (SQLException e ){

e.printStackTrace();

}

}

return resultSet;

}

// 纯Java方式设置参数，使用dbcp池

public void initDBCP() {

pool = new BasicDataSource();// 连接池

pool.setUsername("root");

pool.setPassword("wuzhengyi");

pool.setDriverClassName("com.mysql.jdbc.Driver");

pool.setUrl("jdbc:mysql://localhost:3306/lab3?useSSL=false");

//可以我们自己设置池的相关参数，如最大连接数

// pool.setMaxTotal(6);

getConnection();

}

public void getConnection(){

try {

// 从它的池中获取连接

conn = pool.getConnection();

//语句

if(!conn.isClosed())

stmt = conn.createStatement();

}

catch (SQLException e){

e.printStackTrace();

}

}

public synchronized void endDBCP(){

try {

if(!conn.isClosed())

conn.close();//关闭链接

}

catch (SQLException e){

e.printStackTrace();

}

}

}

**Cmp.java**

import java.sql.ResultSet;  
import java.util.concurrent.ExecutorService;  
import java.util.concurrent.Executors;  
import java.util.concurrent.TimeUnit;  
  
class cmpJDBC extends JDBC{  
 //覆盖基类的测试方法,此处连接、查询、关闭Cmp.NR\_TEST次。  
 @Override  
 public void testJDBC(){  
 for(int i = 0; i< Cmp.*NR\_TEST*; i++) {  
 getConnection();  
 querySql("select \* from staff;");//执行查询  
// executeSql("INSERT INTO Department\n" +  
// "VALUES \n" +  
// "('清洁工',1,131);");  
 closeConnection();  
  
 }  
// //用一个JDBC连接执行2000语句  
// getConnection();  
// for(int i = 0; i< Cmp.NR\_TEST; i++) {  
// querySql("select \* from staff;");//执行查询  
// }  
// closeConnection();  
 }  
}  
  
  
class cmpDBCP extends DBCP{  
 //覆盖基类的测试方法,此处使用10个线程的线程池进行连接、查询、关闭共Cmp.NR\_TEST次。  
 @Override  
 public ResultSet testDBCP() {  
 ExecutorService executorService = Executors.*newFixedThreadPool*(Cmp.*NR\_THREADS*);//固定的十线程池  
 for (int i = 0; i < Cmp.*NR\_TEST*; i++) {  
 executorService.execute(() -> {  
 getConnection();  
 super.testDBCP();  
 endDBCP();  
 });  
 }  
  
// //用一个连接读写DBCP  
// getConnection();  
// for (int i = 0; i < Cmp.NR\_TEST; i++) {  
// executorService.execute(() -> {  
// super.testDBCP();  
// });  
// }  
// endDBCP();  
  
 executorService.shutdown();//不允许再往线程池中添加任务  
  
 try {//等待直到所有任务完成  
 executorService.awaitTermination(Long.*MAX\_VALUE*, TimeUnit.*MINUTES*);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 return null;  
 }  
}  
  
public class Cmp {  
 static final int *NR\_THREADS*=8;//多线程线程数  
 static final int *NR\_TEST*=2000;//  
 private MyTimer timer = new MyTimer();  
 public static void main(String[] args) {  
  
 JDBC jdbc = new JDBC();  
 jdbc.getConnection();  
 jdbc.testJDBC();  
  
 DBCP dbcp = new DBCP();  
 ResultSet resultSet = dbcp.testDBCP();  
 System.*out*.println("DBCP连接池执行选择语句显示所有Staff:");  
 JDBC.*printStaff*(resultSet);  
  
 Cmp cmpTwoMethod=new Cmp();  
// cmpTwoMethod.testJdbcTwoThousandTimes();  
 cmpTwoMethod.testJdbcTwoThousandTimes();  
 cmpTwoMethod.testDbcpTwoThousandTimes();  
// cmpTwoMethod.testDbcpTwoThousandTimes();  
// cmpTwoMethod.testDbcpTwoThousandTimes();  
 }  
  
 public void testJdbcTwoThousandTimes(){  
 timer.start();  
 cmpJDBC jdbc = new cmpJDBC();  
 jdbc.testJDBC();  
 timer.end();  
 System.*out*.print("Jdbc耗时:");  
 System.*out*.println(timer.getTime() +" ms");  
 }  
 //DBCP：使用多线程的2000次sql查询  
 public void testDbcpTwoThousandTimes() {  
 timer.start();  
 cmpDBCP dbcp = new cmpDBCP();  
 dbcp.testDBCP();  
 timer.end();  
 System.*out*.print("Dbcp使用多线程("+*NR\_THREADS*+"个线程)耗时:");  
 System.*out*.println(timer.getTime() +" ms");  
 dbcp.getParameter();  
 }  
}  
  
class MyTimer{  
 long startTime;  
 long endTime;  
 public void start(){  
 startTime = System.*currentTimeMillis*(); //获取开始时间  
 }  
 public void end() {  
 endTime = System.*currentTimeMillis*(); //获取结束时间]  
 }  
  
 public long getTime(){  
 return endTime - startTime;  
 }  
}

实验中遇到的困难及解决办法

Java数据库在连接mysql时应用多线程经常抛异常报错，之后发现是多线程的同步问题。应用同步锁后就解决了。

参考文献及致谢

https://www.cnblogs.com/sunseine/p/5947448.html