设计 学号\_\_\_ 姓名\_\_

2019年11月

班级

请将答案写在答题纸上，写明题号，不必抄题，字迹工整、清晰;

请在答题纸和试题纸上都写上你的班级，学号和姓名，交卷时请将试题纸、答题纸和草纸一并交上来。

一、根据要求回答问题(每小题5分，共40分)

1.写出在 DB2 环境下，连接SAMPLE 数据库的SQL语句和查询emplyee表所有数据的SQL语句。

2.假设有如下代码片段，写出遍历结果集rs所有列的java 代码片段:

Statement stmt = con.createStatement();

ResultSet rs = stmt.executeQuery

("SELECT EMPNO, LASTNAME "+

" FROM TEMPL"+

" WHERE SALARY >40000 ");

1. 有语句 Connection con = null ; con.setAutoCommit(false)。请解释setAutoCommit方法参数为true和false 的区别。

setAutoCommit(true)：启用自动提交模式，每个SQL语句独立提交。

setAutoCommit(false)：禁用自动提交模式，多个SQL语句组合成一个事务，需要手动提交或回滚。

1. 请解释 ResultSet类中wasNulI()方法的作用。

wasNull() 方法的作用

作用：检查上一次从结果集中读取的值是否为 SQL NULL。

返回值：如果上一次读取的值是 SQL NULL，则返回 true；否则返回 false。

用途：主要用于处理那些不能直接表示 null 值的数据类型（如 int、long 等基本数据类型）。

5.假设有如下代码片段，请解释prs.TYPE\_FORWARD\_ONLY和prs.CONCUR\_READ\_ONLY 参数作用。

ResultSet prs = null;

PreparedStatement pstmt = con.prepareStatemente("SELECT EMPNO, LASTNAME"

" FROM TEMPL"+

" WHERE WORKDEPT-? ", prs.TYPE\_FORWARD ONLY,prs.CONCUR READ\_ONLY)

ResultSet.TYPE\_FORWARD\_ONLY：指定结果集只能向前移动，不能反向或随机访问，适用于顺序读取的场景。

ResultSet.CONCUR\_READ\_ONLY：指定结果集是只读的，不能进行更新操作，确保数据的完整性。

7.假设有代码片段如下，补充将数组数据存储到数据库的批处理java 代码。

short [] edLvl ={ 14,18,16);

String [] workDpt={ "D11","DO1","BO1" };

String exSql ="UPDATE EMP SET EDLEVEL =? "+ "WHERE WORKDEPT=? ";

PreparedStatement ps = con.prepareStatement(exSql);

for(int i=0;i<edLvL.length;i++)

{

ps.setShort(1,edLvl[i]);

Ps.setString(2,workDept[i]);

Ps.addBatch();  
}

Int [] rowsCount=ps.executeBatch();

for(int i=0;i<i<edLvL.length;i++)

{

System.out.println(“rowscoutt:”+rowsCount[i]);

}

ps.close();

Con.close();

1. 请解释 ResultSet 类中afterLast()和previous()方法的作用。

afterLast()：将光标移动到结果集的最后一行之后。

previous()：将光标移动到当前行的前一行，并返回一个布尔值表示移动是否成功。

二、根据要求写出 java代码(共40分)

1.假设有如下代码片段，补充读取Blob对象到图片文件的java代码。(10分)

java.sql.Statement st=conn.createStatement();

java.sql.ResultSet rs=st.executeQuery("select picture from administrator.emp\_photo where empno='000150'");

While(rs.next())

{

Blob blob=rs.getBlob(1);

InputStream inputstream=blob.getBinaryStream();

File file=new File(“c:/a.jpg”);

FileOutputStream fo=new FileOutputStream(file);

Int c;

While((c=inputstream.read())!=-1)

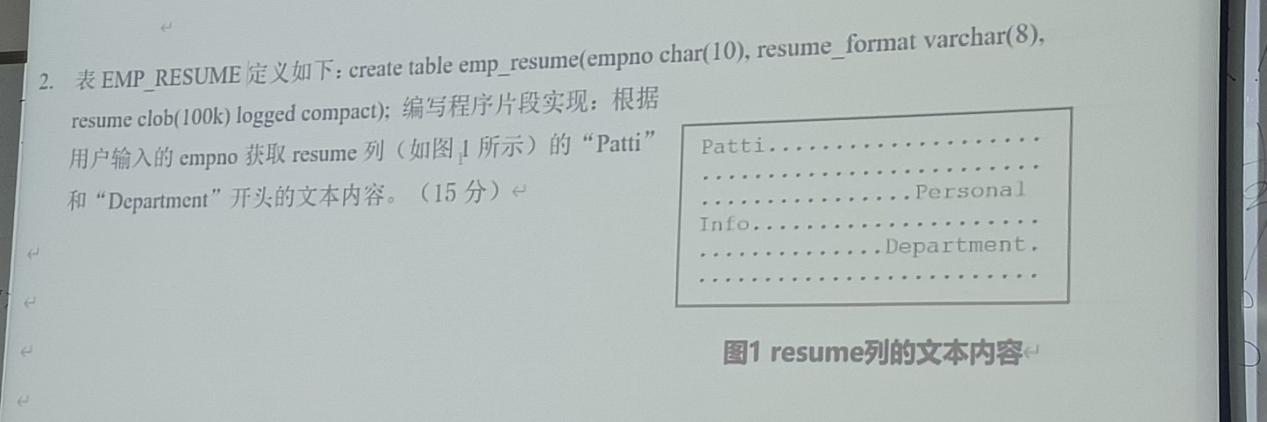
{

fo.write(c);

}

fo.close();

}



Scanner scanner =new Scanner();//输入

String empno=scanner.nextLine();

String sql1=”select posstr(resume,’Personal’) from emp\_resume”+”where empno=?”;

Preparedstatement pstmt1=prepareStatement(sql1);

pstmt1.setString(1,empno);

ResultSet rs1=pstmt1.executeQuery();

while(rs1.next())

{

int startper=rs1.getInt(1);

}

String sql2=”select posstr(resume,’Department’) from emp\_resume”+”where empno=?”;

Preparedstatement pstmt2=prepareStatement(sql2);

Pstmt2.setString(1,empno);

ResultSet rs2=pstmt2.executeQuery();

while(rs2.next())

{

int startDept=rs2.getInt(1);

}

int startper1=startper-1;

String sql3=”select sunstr(resume,1,startper1)||substr(resume,startDept)”+”from emp\_resume”

+”where empno=?”;

Preparedstatement pstmt3=prepareStatement(sql3);

Pstmt3.setString(1,empno);

ResultSet rs3=pstmt3.executeQuery();

While(rs3.next())

{

Clob clob=rs3.getClob(1);

String resume=clob.getSubString((long)1,(int)clob.length());

System.out.println(“RESUME:”+resume);  
}

rs1.close();

rs2.close();

rs3.close();

pstmt1.close();

pstmt2.close();

pstmt3.close();

con.close();

3.(15分)

编写程序片段实现“Update e mp set salary=salary\*1.1 where workdept=?”，要求

(1)至少实现如下的容错处理:数据类型不兼容(SQLSTATE=42818)、输入的部门编号不存在(SQLSTATE=02000)

(2)分析执行该语句过程中还有可能出现哪些异常?

三、编程题(20分)

模拟某宝的电商平台，编写程序实现某商品的信息查询和购买功能，其中商品、顾客和订单表结

构分别如下:

Products (pid,p\_name,price,stock); Orders (oid,cid,pid, date,amount):

Customers (cid,c\_name,birthdate,address);

要求用户输入商品名称(p\_name)，可实现① 模糊查询;②可按照价格(price)进行升序或降序排列:

顾客(cid)购买商品时，如果购买数量(amount)超过实际库存(stock)时，应提示“商品库存不足”;

如果购买成功，应生成一条订单(orders)信息，同时商品库存相应减少。

3

import java.sql.\*;

import java.util.Scanner;

public class ECommercePlatform {

static {

try {

Class.forName("com.mysql.cj.jdbc.Driver"); // 根据你的数据库类型选择合适的驱动

} catch (ClassNotFoundException e) {

e.printStackTrace();

}

}

public static void main(String[] args) {

Connection conn = null;

PreparedStatement pstmt = null;

ResultSet rs = null;

Scanner scanner = new Scanner(System.in);

try {

// 数据库连接信息

String url = "jdbc:mysql://localhost:3306/ecommerce";

String user = "your\_username";

String password = "your\_password";

// 建立数据库连接

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

// 模糊查询商品信息

System.out.print("请输入商品名称 (p\_name) 进行模糊查询: ");

String p\_name = scanner.nextLine();

System.out.print("请选择排序方式 (1: 升序, 2: 降序): ");

int sortOrder = scanner.nextInt();

scanner.nextLine(); // 清除缓冲区

String orderBy = (sortOrder == 1) ? "ASC" : "DESC";

// 查询商品信息的SQL语句

String sql = "SELECT \* FROM Products WHERE p\_name LIKE ? ORDER BY price " + orderBy;

pstmt = conn.prepareStatement(sql);

pstmt.setString(1, "%" + p\_name + "%");

// 执行查询

rs = pstmt.executeQuery();

System.out.println("查询结果:");

System.out.println("PID | P\_NAME | PRICE | STOCK");

while (rs.next()) {

int pid = rs.getInt("pid");

String productName = rs.getString("p\_name");

double price = rs.getDouble("price");

int stock = rs.getInt("stock");

System.out.printf("%-4d | %-10s | %-6.2f | %-5d\n", pid, productName, price, stock);

}

// 购买商品

System.out.print("请输入要购买的商品ID (pid): ");

int pid = scanner.nextInt();

scanner.nextLine(); // 清除缓冲区

System.out.print("请输入购买数量 (amount): ");

int amount = scanner.nextInt();

scanner.nextLine(); // 清除缓冲区

// 检查库存

String checkStockSql = "SELECT stock FROM Products WHERE pid = ?";

pstmt = conn.prepareStatement(checkStockSql);

pstmt.setInt(1, pid);

rs = pstmt.executeQuery();

if (rs.next()) {

int currentStock = rs.getInt("stock");

if (currentStock < amount) {

System.out.println("商品库存不足，当前库存: " + currentStock);

} else {

// 更新库存

String updateStockSql = "UPDATE Products SET stock = stock - ? WHERE pid = ?";

pstmt = conn.prepareStatement(updateStockSql);

pstmt.setInt(1, amount);

pstmt.setInt(2, pid);

int rowsUpdated = pstmt.executeUpdate();

if (rowsUpdated > 0) {

// 生成订单

System.out.print("请输入顾客ID (cid): ");

int cid = scanner.nextInt();

scanner.nextLine(); // 清除缓冲区

String insertOrderSql = "INSERT INTO Orders (oid, cid, pid, date, amount) VALUES (?, ?, ?, NOW(), ?)";

pstmt = conn.prepareStatement(insertOrderSql, Statement.RETURN\_GENERATED\_KEYS);

pstmt.setInt(1, generateOrderId()); // 生成订单ID

pstmt.setInt(2, cid);

pstmt.setInt(3, pid);

pstmt.setInt(4, amount);

int rowsInserted = pstmt.executeUpdate();

if (rowsInserted > 0) {

System.out.println("购买成功，订单已生成。");

} else {

System.out.println("订单生成失败。");

}

} else {

System.out.println("库存更新失败。");

}

}

} else {

System.out.println("没有找到商品ID为 " + pid + " 的商品。");

}

} catch (SQLException e) {

e.printStackTrace();

} finally {

// 关闭资源

try {

if (rs != null) rs.close();

if (pstmt != null) pstmt.close();

if (conn != null) conn.close();

if (scanner != null) scanner.close();

} catch (SQLException e) {

e.printStackTrace();

}

}

}

public static int generateOrderId() {

// 生成订单ID的逻辑，这里简单返回一个随机数

return (int) (Math.random() \* 100000);

}

}