第八章　第八次作业

1. (单选题, 7.6 分) MySQL Connector/J 是一个 JDBC () 驱动

   A. Type 1　　B. Type 2　　C. Type 3　　D. [Type 4]

2. (单选题, 7.8 分) 使用 Class 类的 forName() 加载驱动程序需要捕获 () 异常。

   A. SQLException　　B. [ClassNotFoundException]

   C. IOException　　D. DBException

3. (单选题, 7.8 分) 使用 Connection 接口的 () 方法可以建立一个 PreparedStatement 对象

   A. createPrepareStatement()　　B. createPreparedStatement()

   C. [PrepareStatement()]　　D. PreparedStatement()

4. (单选题, 7.8 分) 在 JDBC 编程中执行下列 SQL 语句"SELECT id,name FROM employee", 获取结果集 rs 的第 1 列数据的代码是

   A. [rs.getString(1)]　　B. rs.getString(2)

   C. rs.getString(name)　　D. rs.getString(id)

5. (单选题, 7.8 分) 如果要创建带参数的 SQL 查询语句，应该使用 ( ) 对象

   A. Statement　　B. [PreparedStatement]

   C. PrepareStatment　　D. CallableStatement

6. (单选题, 7.8 分) 下面描述中不属于连接池的功能的是

   A. 可以缓解连接频繁的关闭和创建所造成的系统性能下降

   B. 可以限制客户端的连接数量

   C. [可以大幅度提高查询语句的执行效率]

   D. 可以提高系统的伸缩性

7. (单选题, 7.8 分) 下面关于 PreparedStatement 的说法错误的是 ()

   A. PreparedStatement 继承 Statement

   B. PreparedStatement 可以防止 SQL 注入，更加安全

   C. [PreparedStatement 不能用于批量更新操作]

   D. PreparedStatement 可以存储预编译的 Statement，从而提升执行效率

8. (单选题, 7.8 分) 在使用 JDBC 连接数据库时，以下哪个选项描述了正确的连接流程?

   A. 建立数据库连接 -> 创建 statement 对象 -> 执行 SQL 语句 -> 处理结果集 -> 加载 JDBC 驱动程序

   B. 加载 JDBC 驱动程序 -> 建立数据库连接 -> 创建 statement 对象 -> 执行 SQL 语句 -> 处理结果集

   C. 创建 statement 对象 -> 建立数据库连接 -> 执行 SQL 语句 -> 处理结果集 -> 加载 JDBC 驱动程序

   D. 加载 JDBC 驱动程序 -> 建立数据库连接 -> 处理结果集 -> 创建 statement 对象 -> 执行 SQL 语句

9. (多选题, 7.8 分) 下面哪个 JDBC 方法可能会改变数据库中的已有数据?

   A. executeQuery()　　B. executeUpdate()

   C. executeInsert()　　D. executeBatch()

10. (简答题, 7.8 分) 加载 JDBC 驱动，使用 Java 连接 university 数据库，查询 instructor 表中 salary 大于 70000 的教师信息，按照 dept_name 的升序排列后用 System.out.println() 输出，各列数据之间用 | 分隔。

    ID|name|dept_name|salary

    将编写的 Java 类代码贴在下方输入框中，并给出输出结果。

```java
import java.sql.*;

public class Question10 {

    // MySQL 8.0 以上版本 - JDBC 驱动名及数据库 URL
    static final String JDBC_DRIVER = "com.mysql.cj.jdbc.Driver";
    static final String DB_URL = "jdbc:mysql://公网
    ↪  IP:3306/user000db?useSSL=false&allowPublicKeyRetrieval=true&serverTimezone=UTC";


    // 数据库的用户名与密码，需要根据自己的设置
    static final String USER = "DB_USER000";
    static final String PASS = "DB_USER000@123";

    public static void main(String[] args) {
        Connection conn = null;
        Statement stmt = null;
        try{
            // 注册 JDBC 驱动
            Class.forName(JDBC_DRIVER);
```

```
        // 打开链接
        System.out.println(" 连接数据库...");
        conn = DriverManager.getConnection(DB_URL,USER,PASS);

        // 执行查询
        System.out.println(" 实例化 Statement 对象...");
        stmt = conn.createStatement();
        String sql;
        sql = "SELECT ID, name, dept_name, salary from instructor where
        ↪  salary > 70000 order by dept_name";
        ResultSet rs = stmt.executeQuery(sql);

        // 展开结果集数据库
        while(rs.next()){
            // 通过字段检索
            String id  = rs.getString("ID");
            String name = rs.getString("name");
            String dept_name = rs.getString("dept_name");
            Double salary = rs.getDouble("salary");

            // 输出数据
            System.out.print(id);
            System.out.print("|" + name + "|" + dept_name + "|" +
            ↪  salary);
            System.out.print("\n");
        }
        // 完成后关闭
        rs.close();
        stmt.close();
        conn.close();
    }catch(SQLException se){
        // 处理 JDBC 错误
        se.printStackTrace();
    }catch(Exception e){
        // 处理 Class.forName 错误
        e.printStackTrace();
    }finally{
        // 关闭资源
        try{
            if(stmt!=null) stmt.close();
        }catch(SQLException se2){
        }// 什么都不做
```

```
            try{
                if(conn!=null) conn.close();
            }catch(SQLException se){
                se.printStackTrace();
            }
        }
        System.out.println("Goodbye!");
    }
}
```

以下是输出结果：

```
连接数据库...
 实例化 Statement 对象...
76766|Crick|Biology|72000.0
45565|Katz|Comp. Sci.|75000.0
83821|Brandt|Comp. Sci.|92000.0
98345|Kim|Elec. Eng.|80000.0
12121|Wu|Finance|90000.0
76543|Singh|Finance|80000.0
22222|Einstein|Physics|95000.0
33456|Gold|Physics|87000.0
Goodbye!
```

11. (简答题, 7.8 分) 查阅资料学习 JDBC CallableStatements 执行存储过程的方法：https://dev.mysql.com/doc/connector-j/8.0/en/connector-j-usagenotes-statements-callable.html

(1) 在 Univerisity 中编写一个带输入参数和输出参数的存储过程 demoSp，功能自定，给出存储过程定义并说明自定义的功能。

(2) 编写 Java 代码调用存储过程，指定输入参数，获取输出参数后输出。给出调用存储过程的关键代码并给出输出结果。

存储过程的作用为将输入的变量 +1。

```
import java.sql.*;

public class Question11 {

    // MySQL 8.0 以上版本 - JDBC 驱动名及数据库 URL
    static final String JDBC_DRIVER = "com.mysql.cj.jdbc.Driver";
    static final String DB_URL = "jdbc:mysql://公网
    ↪ IP:3306/user000db?useSSL=false&allowPublicKeyRetrieval=true&serverTimezone=UTC";
```

```
// 数据库的用户名与密码，需要根据自己的设置
static final String USER = "DB_USER000";
static final String PASS = "DB_USER000@123";

public static void main(String[] args) {
    Connection conn = null;
    Statement stmt = null;
    try{
        // 注册 JDBC 驱动
        Class.forName(JDBC_DRIVER);

        // 打开链接
        System.out.println(" 连接数据库...");
        conn = DriverManager.getConnection(DB_URL,USER,PASS);

        // 执行查询
        System.out.println(" 实例化 Statement 对象...");
        stmt = conn.createStatement();

        Boolean rs = stmt.execute("drop procedure if exists demoSp;");
        rs = stmt.execute("create procedure demoSp(inout test int) begin
        ↪  set test = test + 1; end");

        CallableStatement cs = conn.prepareCall("{call demoSp(?)}");
        System.out.println(" 调用存储过程，作用为将输入的变量 +1: ");
        int res = 0;
        System.out.println(" 输入变量: ");
        System.out.println(res);
        cs.setInt(1, res);
        cs.registerOutParameter(1, Types.INTEGER);
        cs.execute();
        res = cs.getInt(1);
        System.out.println(" 调用结果: ");
        System.out.println(res);

        // 完成后关闭
        cs.close();
        stmt.close();
        conn.close();
    }catch(SQLException se){
```

```
            // 处理 JDBC 错误
            se.printStackTrace();
        }catch(Exception e){
            // 处理 Class.forName 错误
            e.printStackTrace();
        }finally{
            // 关闭资源
            try{
                if(stmt!=null) stmt.close();
            }catch(SQLException se2){
            }// 什么都不做
            try{
                if(conn!=null) conn.close();
            }catch(SQLException se){
                se.printStackTrace();
            }
        }
        System.out.println("Goodbye!");
    }
}
```

以下是输出结果:

```
连接数据库...
实例化 Statement 对象...
调用存储过程，作用为将输入的变量 +1:
输入变量:
0
调用结果:
1
Goodbye!
```

12. (简答题, 7.8 分) 考虑两个都具有属性 X 的实体集 A 和 B (其中另外的名称与此问题无关)。

a. 如果两个 X 完全不相关，那么应该如何改进设计呢?

应该把两个 X 属性改名，以便区分这两个不相关的属性。

b. 如果这两个 X 代表相同的性质，并且它同时适用于 A 和 B, 那么应该如何改进设计呢? 考虑三种子情况:

· X 是 A 的主码，但不是 B 的主码。

可以将 X 作为 B 的外码，参考 A 的主码。

```
alter table B add foreign key (X) references A(X);
```

· X 是 A 和 B 的主码。

在满足某些范式的情况下，可以把 A 和 B 合并起来，仍使用 X 作为主码。

· X 不是 A 或 B 的主码。

将 A 和 B 概化出一个实体集，将 X 作为此实体集的属性，并将 X 从 A 和 B 中去除。

13. (简答题, 7.8 分) 考虑实体集 A 和 B 之间的多对一联系 R。假设由 R 生成的关系和由 A 生成的关系合并了。在 SQL 中，参与外码约束的属性可以为空。请解释如何使用 SQL 中的 not null 约束来强制实施 A 在 R 中的全部参与约束。

假设联系 R 的属性名为 X，那么可以添加外键约束和非空约束如下:

```
alter table A add foreign key (X) references B(X);
alter table A modify X not null;
alter table B modify X not null;
```