

华东师范大学软件工程学院实验报告

实验课程：数据库系统及其应用实践

年级：2023 级

实验成绩：

实验名称：Lab-02

姓名：顾翌炜

实验编号：Lab-02

学号：10235101527

实验日期：2025/03/26

指导教师：姚俊杰

组号：01

实验时间：2 课时

1 实验目的

本实验旨在通过 JDBC (Java Database Connectivity) 技术, 实现 Java 应用程序与 MySQL 数据库的交互。通过本实验, 学生将掌握以下内容:

- 了解 JDBC 的基本概念和原理。
- 学习如何使用 JDBC 连接数据库。
- 练习 SQL 语句的编写和在 Java 中执行 SQL 查询。
- 掌握使用 PreparedStatement 进行安全查询。
- 通过编写 Java 代码, 实现对数据库的增、删、改、查等操作。
- 计算学生的 GPA, 并提高对数据库应用的实际能力。

2 实验要求

为了顺利完成本实验, 学生需要满足以下要求:

- 熟悉 Java 语言基础, 掌握面向对象编程思想。
- 了解基本的 SQL 语法, 包括 SELECT、INSERT、UPDATE、DELETE 等操作。
- 安装并配置 MySQL 数据库, 创建实验所需的数据表。
- 安装 JDK 并配置 Java 开发环境, 确保能够编写和运行 Java 程序。
- 下载并正确引用 MySQL JDBC 驱动 (Jar 包)。
- 编写 Java 代码, 实现用户登录、查询学生信息、查询课程信息及计算 GPA 等功能。
- 代码需具有较好的可读性, 采用异常处理机制, 确保系统的稳定性和安全性。

3 实验过程记录

3.1 下载 Java SE

为了使用 JDBC 连接 MySQL 数据库，我们需要先安装 Java SE (Java Platform, Standard Edition)。本实验使用 Java SE 8 及以上版本。以下是下载和安装步骤：

3.1.1 访问 Oracle 官方网站

首先，打开浏览器，进入 Oracle Java 官方下载页面：

<https://www.oracle.com/java/technologies/javase-downloads.html>

3.1.2 选择 Java SE 版本

- 点击 **JDK 下载**，选择适用于操作系统 (Windows/macOS/Linux) 的版本。

3.1.3 下载并安装 JDK

1. 下载 JDK 安装包后，运行安装程序，并按照默认选项完成安装。
2. 安装完成后，确认 JDK 目录，例如：

3.1.4 配置环境变量 (Windows 系统)

1. 右键“此电脑”→“属性”→“高级系统设置”→“环境变量”。
2. 在 **系统变量** 中找到 **Path**，点击“编辑”，添加 JDK 的 **bin** 目录
3. 新建 **JAVA_HOME** 变量，值为 JDK 安装路径

3.1.5 验证安装

打开终端或命令提示符，输入：

```
java -version
```

若正确显示 Java 版本信息，则安装成功。



图 1: 安装 Java SE



图 2: 安装 Java SE

3.2 安装 JDBC 需要的 Jar 包

本实验使用 MySQL 数据库，需要 MySQL JDBC 驱动程序（Jar 包）以支持 Java 访问数据库。

1. 下载老师提供的 JDBC 安装包。
2. 解压后，获取 `mysql-connector-java-8.4.0.jar` 文件。
3. 在 Java 项目中引入该 Jar 包，确保项目能够使用 JDBC 连接数据库。

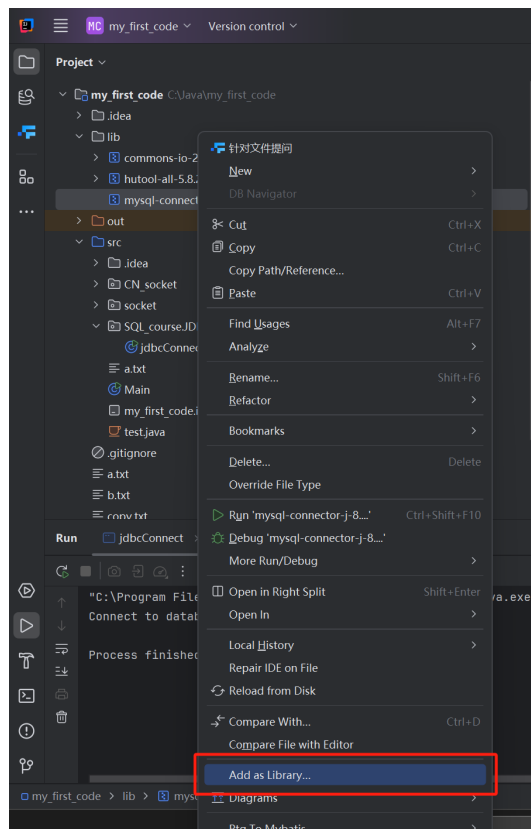


图 3: 添加到 Library

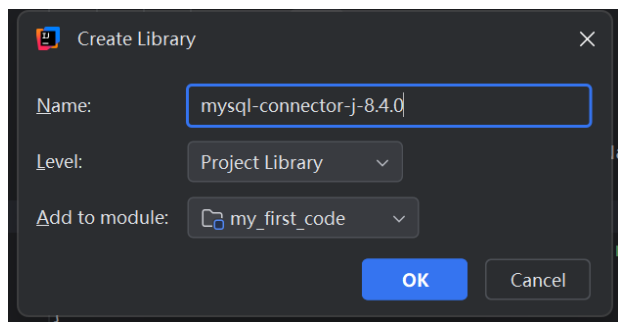


图 4: Create Library

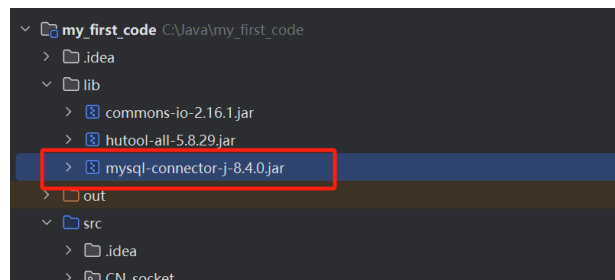


图 5: 添加到 Library 完毕

3.3 使用 JDBC

3.3.1 测试连接

jdbcTemplate

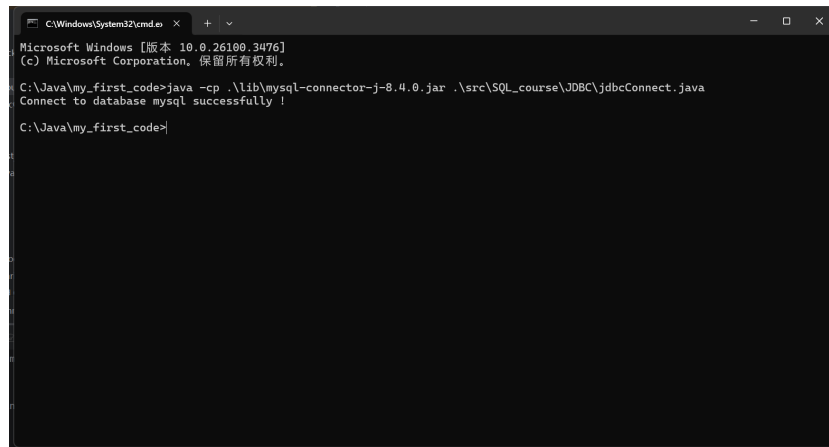
```
1 package SQL_course.JDBC;
```

```
2
3  import java.sql.Connection;
4  import java.sql.DriverManager;
5
6  public class jdbcConnect {
7      public static void main(String args[]) {
8          Connection c = null;
9          try {
10              Class.forName("com.mysql.cj.jdbc.Driver");
11              c = DriverManager.getConnection(
12                  "jdbc:mysql://localhost:3306/lab-2_college",
13                  "root",
14                  "GUyiwei_)*@@");
15          } catch (Exception e) {
16              e.printStackTrace();
17              System.err.println(e.getClass().getName() + ": " + e.getMessage());
18              System.exit(0);
19          }
20          System.out.println("Connect to database mysql successfully!");
21      }
22  }
```

然后在命令行内输入以下指令：

```
java -cp .\lib\mysql-connector-j-8.4.0.jar .\src\SQL_course\JDBC\jdbcConnect.java
```

得到以下结果：



```
C:\Windows\System32\cmd.exe
Microsoft Windows [版本 10.0.26100.3476]
(c) Microsoft Corporation. 保留所有权利。

C:\Java\my_first_code>java -cp .\lib\mysql-connector-j-8.4.0.jar .\src\SQL_course\JDBC\jdbcConnect.java
Connect to database mysql successfully !

C:\Java\my_first_code>
```

图 6: 测试连接

可以看到：Connect to database mysql successfully ! 连接成功

3.3.2 创建表格

jdbcCreate

```
1  package SQL_course.JDBC;
2
3  import java.sql.Connection;
4  import java.sql.DriverManager;
5  import java.sql.Statement;
6
7  public class jdbcCreate {
8      public static void main(String args[]) {
9          Connection c = null;
10         Statement stmt = null;
11
12         try {
13             // 加载 MySQL JDBC 驱动
14             Class.forName("com.mysql.cj.jdbc.Driver");
15
16             // 建立连接
17             c = DriverManager.getConnection(
18                 "jdbc:mysql://localhost:3306/lab-2_college",
19                 "root",
20                 "GUyiwei_)*@@"
21             );
22             System.out.println("Connect to database mysql successfully!");
23
24             // 创建 Statement 对象
25             stmt = c.createStatement();
26
27             // 创建表的 SQL 语句
28             String sql = "CREATE TABLE employee " +
29                 "(id INT, " +
30                 "name VARCHAR(20) NOT NULL, " +
31                 "age INT NOT NULL, " +
32                 "address VARCHAR(50), " +
33                 "salary REAL, " +
34                 "PRIMARY KEY(id))";
35
36             // 执行创建表的 SQL 语句
37             stmt.executeUpdate(sql);
38             stmt.close();
39             c.close();
40
41         } catch (Exception e) {
```

```

42         // 捕获并打印异常
43         System.err.println(e.getClass().getName() + ": " + e.getMessage());
44         System.exit(0);
45     }
46     // 打印成功消息
47     System.out.println("Create table company successfully!");
48 }
49 }

```

然后在命令行内输入以下指令：

```
java -cp .\lib\mysql-connector-j-8.4.0.jar .\src\SQL_course\JDBC\jdbcCreate.java
```

得到以下结果：

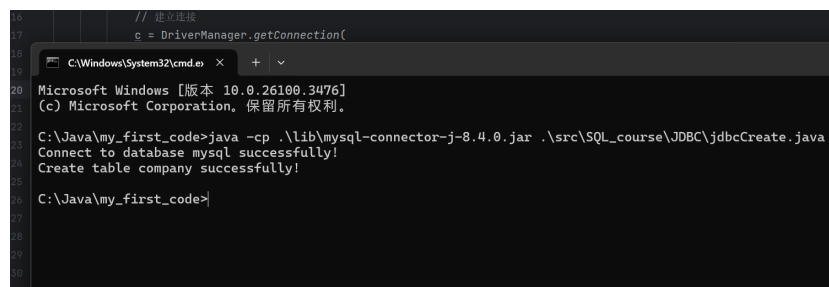


图 7: 创建表格

可以看到：

Connect to database mysql successfully !

Create table company successfully!

创建成功

3.3.3 插入数据

jdbcInsert

```

1     package SQL_course.JDBC;
2
3     import java.sql.Connection;
4     import java.sql.DriverManager;
5     import java.sql.Statement;
6
7     public class jdbcInsert {
8         public static void main(String args[]) {
9             Connection c = null;
10            Statement stmt = null;

```

```
11
12     try {
13         // 加载 MySQL JDBC 驱动
14         Class.forName("com.mysql.cj.jdbc.Driver");
15
16         // 建立连接
17         c = DriverManager.getConnection(
18             "jdbc:mysql://localhost:3306/lab-2_college",
19             "root",
20             "GUyiwei_)*@@"
21         );
22         System.out.println("Connected to the database successfully!");
23
24         // 创建 Statement 对象
25         stmt = c.createStatement();
26
27         // 插入数据的 SQL 语句
28         String sql1 = "INSERT INTO employee VALUES (1, 'Gong', 48, '2075",
29             "Kongjiang Road', 20000.00);";
30         String sql2 = "INSERT INTO employee VALUES (2, 'Luan', 25, '3663",
31             "Zhongshan Road(N)', 15000.00);";
32         String sql3 = "INSERT INTO employee VALUES (3, 'Hu', 23, '3663",
33             "Zhongshan Road(N)', 15000.00);";
34         String sql4 = "INSERT INTO employee VALUES (4, 'Jin', 24, '3663",
35             "Zhongshan Road(N)', 15000.00);";
36         String sql5 = "INSERT INTO employee VALUES (5, 'Yi', 24, '3663",
37             "Zhongshan Road(N)', 15000.00);";
38
39         // 执行插入数据的 SQL 语句
40         stmt.executeUpdate(sql1);
41         stmt.executeUpdate(sql2);
42         stmt.executeUpdate(sql3);
43         stmt.executeUpdate(sql4);
44         stmt.executeUpdate(sql5);
45
46         stmt.close();
47         c.close();
48
49         System.out.println("Records inserted successfully!");
50
51     } catch (Exception e) {
52         // 捕获并打印异常
53         System.err.println(e.getClass().getName() + ": " + e.getMessage());
54     }
```



```

49         System.exit(0);
50     }
51 }
52 }

```

然后在命令行内输入以下指令：

```
java -cp .\lib\mysql-connector-j-8.4.0.jar .\src\SQL_course\JDBC\jdbcInsert.java
```

得到以下结果：

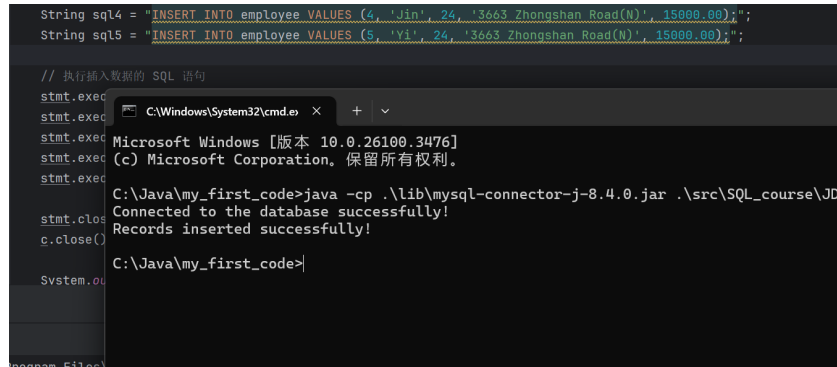


图 8: 插入数据

可以看到：

Connect to database mysql successfully !

Records inserted successfully!

创建成功

我们也可以在数据库里看到新插入的数据，上传成功：

employee					
id	name	age	address	salary	
1	Gong	48	2075 Kongjiang Road	20000	
2	Luan	25	3663 Zhongshan Road(N)	15000	
3	Hu	23	3663 Zhongshan Road(N)	15000	
4	Jin	24	3663 Zhongshan Road(N)	15000	
5	Yi	24	3663 Zhongshan Road(N)	15000	

图 9: 插入结果

3.3.4 查询数据

jdbcSelect

```
1 package SQL_course.JDBC;
```

```
2
3  import java.sql.Connection;
4  import java.sql.DriverManager;
5  import java.sql.ResultSet;
6  import java.sql.Statement;
7
8  public class jdbcSelect {
9      public static void main(String args[]) {
10          Connection c = null;
11          Statement stmt = null;
12
13          try {
14              // 加载 MySQL JDBC 驱动
15              Class.forName("com.mysql.cj.jdbc.Driver");
16
17              // 建立连接
18              c = DriverManager.getConnection(
19                  "jdbc:mysql://localhost:3306/lab-2_college",
20                  "root",
21                  "GUyiwai_)*@@"
22              );
23              System.out.println("Connect to database mysql successfully!");
24
25              // 创建 Statement 对象
26              stmt = c.createStatement();
27
28              // 查询的 SQL 语句
29              String sql = "SELECT * FROM employee";
30
31              // 执行查询的 SQL 语句
32              ResultSet rs = stmt.executeQuery(sql);
33
34              // 处理结果集
35              while (rs.next()) {
36                  int id = rs.getInt("id");
37                  String name = rs.getString("name");
38                  int age = rs.getInt("age");
39                  String address = rs.getString("address");
40                  double salary = rs.getDouble("salary");
41
42                  // 打印结果
43                  System.out.println("ID: " + id);
44                  System.out.println("Name: " + name);
45                  System.out.println("Age: " + age);
```

```

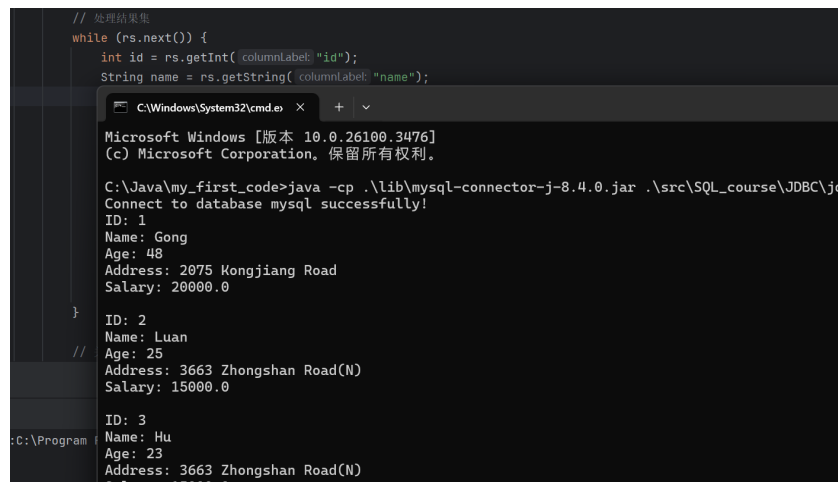
46         System.out.println("Address:␣" + address);
47         System.out.println("Salary:␣" + salary);
48         System.out.println();
49     }
50
51     // 关闭资源
52     rs.close();
53     stmt.close();
54     c.close();
55
56     } catch (Exception e) {
57         // 捕获并打印异常
58         System.err.println(e.getClass().getName() + ":␣" + e.getMessage
59             ());
60         System.exit(0);
61     }
62     // 打印成功消息
63     System.out.println("Select␣operation␣successfully!");
64 }

```

然后在命令行内输入以下指令:

```
java -cp .\lib\mysql-connector-j-8.4.0.jar .\src\SQL_course\JDBC\jdbcSelect.java
```

得到以下结果:



```

// 处理结果集
while (rs.next()) {
    int id = rs.getInt( columnLabel: "id");
    String name = rs.getString( columnLabel: "name");
    ID: 1
    Name: Gong
    Age: 48
    Address: 2075 Kongjiang Road
    Salary: 20000.0
}
//
ID: 2
Name: Luan
Age: 25
Address: 3663 Zhongshan Road(N)
Salary: 15000.0
ID: 3
Name: Hu
Age: 23
Address: 3663 Zhongshan Road(N)
Salary: 15000.0

```

图 10: 查询数据

可以看到:

Connect to database mysql successfully !

ID:1

Name: Gong

Age: 48

Address: 2075 Kongjiang Road

Salary:20000.0

ID:2

Name: Luan

Age: 25

Address:3663 Zhongshan Road(N)

Salary: 15000.0

...

查询成功

3.3.5 更新数据

jdbcUpdate

```
1  package SQL_course.JDBC;
2
3  import java.sql.Connection;
4  import java.sql.DriverManager;
5  import java.sql.ResultSet;
6  import java.sql.Statement;
7
8  public class jdbcUpdate {
9      public static void main(String args[]) {
10         Connection c = null;
11         Statement stmt = null;
12
13         try {
14             // 加载 MySQL JDBC 驱动
15             Class.forName("com.mysql.cj.jdbc.Driver");
16
17             // 建立连接
18             c = DriverManager.getConnection(
19                 "jdbc:mysql://localhost:3306/lab-2_college",
20                 "root",
21                 "GUyiwei_)*@@"
22             );
23             System.out.println("Connect to database mysql successfully!");
24         }
```

```
25      // 创建 Statement 对象
26      stmt = c.createStatement();
27
28      // 更新的 SQL 语句
29      String updateSql = "UPDATE employee SET salary=50000.00 WHERE
        id=1";
30
31      // 执行更新的 SQL 语句
32      stmt.executeUpdate(updateSql);
33      System.out.println("Update operation successfully!");
34
35      // 查询的 SQL 语句
36      String selectSql = "SELECT * FROM employee";
37
38      // 执行查询的 SQL 语句
39      ResultSet rs = stmt.executeQuery(selectSql);
40
41      // 处理结果集
42      while (rs.next()) {
43          int id = rs.getInt("id");
44          String name = rs.getString("name");
45          int age = rs.getInt("age");
46          String address = rs.getString("address");
47          double salary = rs.getDouble("salary");
48
49          // 打印结果
50          System.out.println("ID:" + id);
51          System.out.println("Name:" + name);
52          System.out.println("Age:" + age);
53          System.out.println("Address:" + address);
54          System.out.println("Salary:" + salary);
55          System.out.println();
56      }
57
58      // 关闭资源
59      rs.close();
60      stmt.close();
61      c.close();
62
63      } catch (Exception e) {
64          // 捕获并打印异常
65          System.err.println(e.getClass().getName() + ": " + e.getMessage
            ());
66          System.exit(0);
```

```

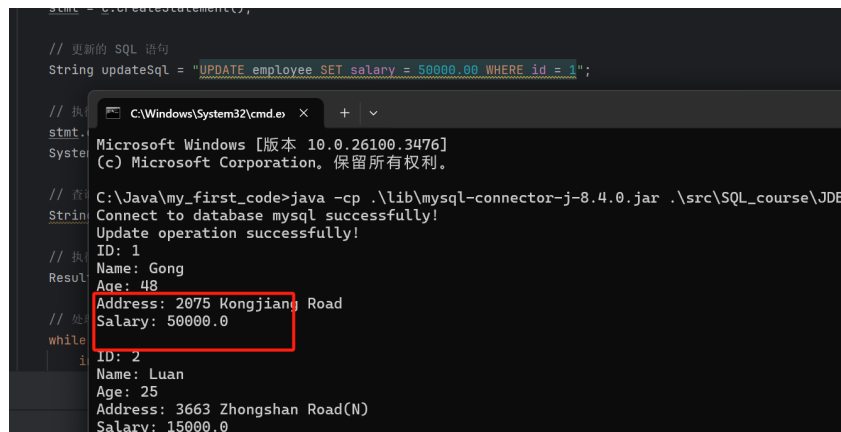
67         }
68         // 打印成功消息
69         System.out.println("Select operation successfully!");
70     }
71 }

```

然后在命令行内输入以下指令：

```
java -cp .\lib\mysql-connector-j-8.4.0.jar .\src\SQL_course\JDBC\jdbcUpdate.java
```

得到以下结果：



```

// 更新的 SQL 语句
String updateSql = "UPDATE employee SET salary = 50000.00 WHERE id = 1";

// 执行
stmt.executeUpdate(updateSql);

// 打印
System.out.println("Update operation successfully!");

// 查询
String querySql = "SELECT * FROM employee";
ResultSet rs = stmt.executeQuery(querySql);

// 遍历结果集
while (rs.next()) {
    int ID = rs.getInt("ID");
    String Name = rs.getString("Name");
    int Age = rs.getInt("Age");
    String Address = rs.getString("Address");
    double Salary = rs.getDouble("Salary");

    System.out.println("ID: " + ID);
    System.out.println("Name: " + Name);
    System.out.println("Age: " + Age);
    System.out.println("Address: " + Address);
    System.out.println("Salary: " + Salary);
}

```

图 11: 更新数据

可以看到：

Connect to database mysql successfully !

且 Gong 的 Salary 被更新到 50000.0

更新成功

3.3.6 删除数据

jdbcDelete

```

1     package SQL_course.JDBC;
2
3     import java.sql.Connection;
4     import java.sql.DriverManager;
5     import java.sql.ResultSet;
6     import java.sql.Statement;
7
8     public class jdbcDelete {
9         public static void main(String args[]) {
10             Connection c = null;

```

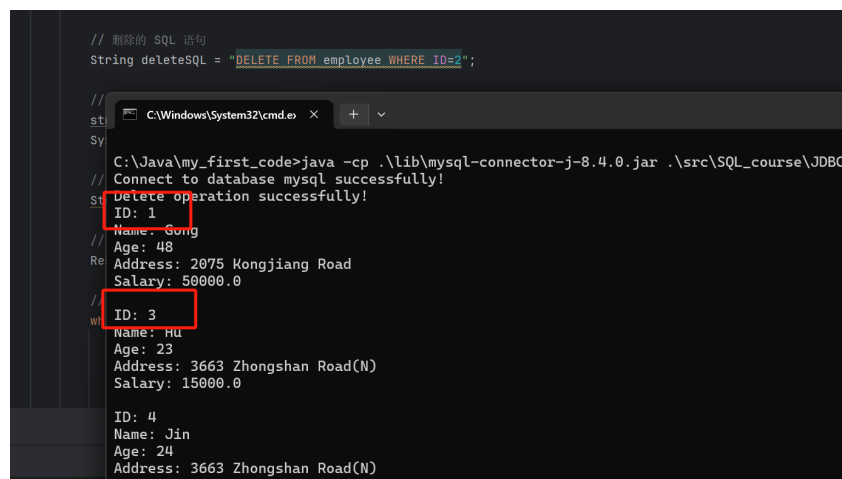
```
11         Statement stmt = null;
12
13     try {
14         // 加载 MySQL JDBC 驱动
15         Class.forName("com.mysql.cj.jdbc.Driver");
16
17         // 建立连接
18         c = DriverManager.getConnection(
19             "jdbc:mysql://localhost:3306/lab-2_college",
20             "root",
21             "GUyiwei_)*@@"
22         );
23         System.out.println("Connect to database mysql successfully!");
24
25         // 创建 Statement 对象
26         stmt = c.createStatement();
27
28         // 删除的 SQL 语句
29         String deleteSQL = "DELETE FROM employee WHERE ID=2";
30
31         // 执行删除的 SQL 语句
32         stmt.executeUpdate(deleteSQL);
33         System.out.println("Delete operation successfully!");
34
35         // 查询的 SQL 语句
36         String selectSQL = "SELECT * FROM employee";
37
38         // 执行查询的 SQL 语句
39         ResultSet rs = stmt.executeQuery(selectSQL);
40
41         // 处理结果集
42         while (rs.next()) {
43             int id = rs.getInt("id");
44             String name = rs.getString("name");
45             int age = rs.getInt("age");
46             String address = rs.getString("address");
47             double salary = rs.getDouble("salary");
48
49             // 打印结果
50             System.out.println("ID: " + id);
51             System.out.println("Name: " + name);
52             System.out.println("Age: " + age);
53             System.out.println("Address: " + address);
54             System.out.println("Salary: " + salary);
```

```
55         System.out.println();
56     }
57
58     // 关闭资源
59     rs.close();
60     stmt.close();
61     c.close();
62
63     } catch (Exception e) {
64         // 捕获并打印异常
65         System.err.println(e.getClass().getName() + ": " + e.getMessage());
66         System.exit(0);
67     }
68 }
69 }
```

然后在命令行内输入以下指令：

```
java -cp .\lib\mysql-connector-j-8.4.0.jar .\src\SQL_course\JDBC\jdbcDelete.java
```

得到以下结果：



```
// 删除的 SQL 语句
String deleteSQL = "DELETE FROM employee WHERE ID=2";

//
st
Sy
C:\Java\my_first_code>java -cp .\lib\mysql-connector-j-8.4.0.jar .\src\SQL_course\JDBC
//
Connect to database mysql successfully!
Delete operation successfully!
ID: 1
Name: Gong
Age: 48
Re
Address: 2075 Kongjiang Road
Salary: 50000.0
//
ID: 3
Name: Hu
Age: 23
Address: 3663 Zhongshan Road(N)
Salary: 15000.0
//
ID: 4
Name: Jin
Age: 24
Address: 3663 Zhongshan Road(N)
Salary: 15000.0
```

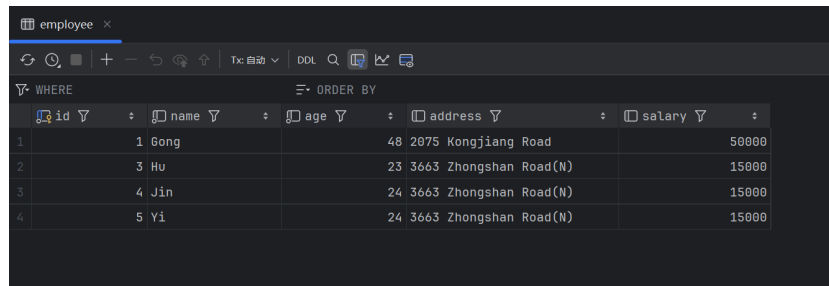
图 12: 删除数据

可以看到：

Connect to database mysql successfully !

且 ID 为 2 的数据已经删除

再来数据库内查看：



id	name	age	address	salary
1	Gong	48	2075 Kongjiang Road	15000
3	Hu	23	3663 Zhongshan Road(N)	15000
4	Jin	24	3663 Zhongshan Road(N)	15000
5	Yi	24	3663 Zhongshan Road(N)	15000

图 13: 删除结果

删除成功

3.3.7 综合处理

jdbcBatch

```
1 package SQL_course.JDBC;
2
3 import java.sql.Connection;
4 import java.sql.DriverManager;
5 import java.sql.PreparedStatement;
6 import java.sql.ResultSet;
7 import java.sql.SQLException;
8 import java.math.BigDecimal;
9
10 public class jdbcBatch {
11     public static void main(String args[]) {
12         Connection c = null;
13         PreparedStatement pstmt = null;
14         ResultSet rs = null;
15
16         try {
17             // 加载 MySQL JDBC 驱动
18             Class.forName("com.mysql.cj.jdbc.Driver");
19
20             // 建立连接
21             c = DriverManager.getConnection(
22                 "jdbc:mysql://localhost:3306/lab-2_college",
23                 "root",
24                 "GUyiwei_)*@@");
25         } catch (Exception e) {
26             System.out.println("Connect to database mysql successfully!");
27         }
28         // 创建 GPA 表的 SQL 语句
```

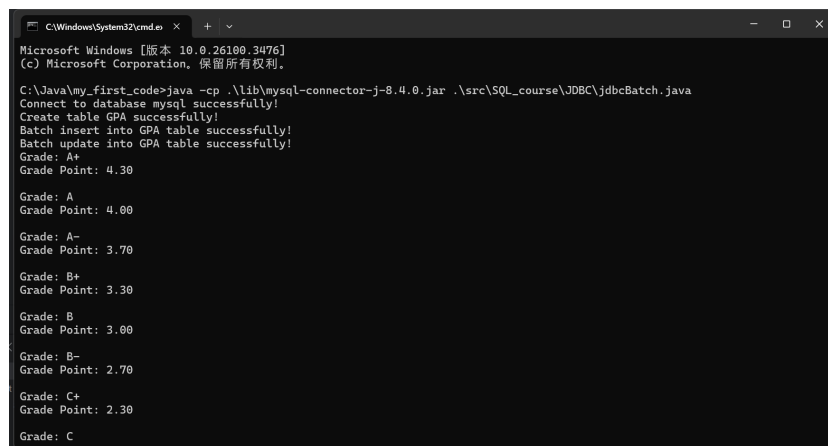
```
29      String createTableSQL = "CREATE TABLE GPA (grade CHAR(2),
30          grade_point DECIMAL(3,2))";
31      PreparedStatement createTableStmt = c.prepareStatement(
32          createTableSQL);
33      createTableStmt.executeUpdate();
34      createTableStmt.close();
35      System.out.println("Create table GPA successfully!");
36
37      // 批量插入数据的 SQL 语句
38      String insertSQL = "INSERT INTO GPA (grade, grade_point) VALUES
39          (?, ?)";
40      pstmt = c.prepareStatement(insertSQL);
41
42      String[] strArray = {"A+", "A", "A-", "B+", "B", "B-", "C+", "C",
43          "C-", "D", "D-", "F"};
44      double[] doubleArray = {4.3, 4.0, 3.7, 3.3, 3.0, 2.7, 2.3, 2.0,
45          1.5, 1.3, 1.0, 0};
46
47      for (int i = 0; i < strArray.length; i++) {
48          pstmt.setString(1, strArray[i]);
49          pstmt.setBigDecimal(2, BigDecimal.valueOf(doubleArray[i]));
50          pstmt.addBatch();
51      }
52
53      pstmt.executeBatch();
54      pstmt.close();
55      System.out.println("Batch insert into GPA table successfully!");
56
57      // 更新数据的 SQL 语句
58      String updateSQL = "UPDATE GPA SET grade_point = ? WHERE grade =
59          ?";
60      pstmt = c.prepareStatement(updateSQL);
61
62      pstmt.executeBatch();
63      pstmt.close();
64      System.out.println("Batch update into GPA table successfully!");
65
66      // 查询的 SQL 语句
67      String selectSQL = "SELECT * FROM GPA";
68      pstmt = c.prepareStatement(selectSQL);
69      rs = pstmt.executeQuery();
70
71      // 处理结果集
72      while (rs.next()) {
```

```
67         String grade = rs.getString("grade");
68         BigDecimal gradePoint = rs.getBigDecimal("grade_point");
69
70         // 打印结果
71         System.out.println("Grade:␣" + grade);
72         System.out.println("Grade␣Point:␣" + gradePoint);
73         System.out.println();
74     }
75
76     // 关闭资源
77     rs.close();
78     pstmt.close();
79     c.close();
80     System.out.println("Select␣operation␣successfully!");
81
82     } catch (Exception e) {
83         // 捕获并打印异常
84         System.err.println(e.getClass().getName() + ":␣" + e.getMessage
85             ());
86         System.exit(0);
87     }
88 }
```

然后在命令行内输入以下指令:

```
java -cp .\lib\mysql-connector-j-8.4.0.jar .\src\SQL_course\JDBC\jdbcBatch.java
```

得到以下结果:



```
Microsoft Windows [版本 10.0.26100.3476]
(c) Microsoft Corporation. 保留所有权利。

C:\Java\my_first_code>java -cp .\lib\mysql-connector-j-8.4.0.jar .\src\SQL_course\JDBC\jdbcBatch.java
Connect to database mysql successfully!
Create table GPA successfully!
Batch insert into GPA table successfully!
Batch update into GPA table successfully!
Grade: A+
Grade Point: 4.30

Grade: A
Grade Point: 4.00

Grade: A-
Grade Point: 3.70

Grade: B+
Grade Point: 3.30

Grade: B
Grade Point: 3.00

Grade: B-
Grade Point: 2.70

Grade: C+
Grade Point: 2.30

Grade: C
```

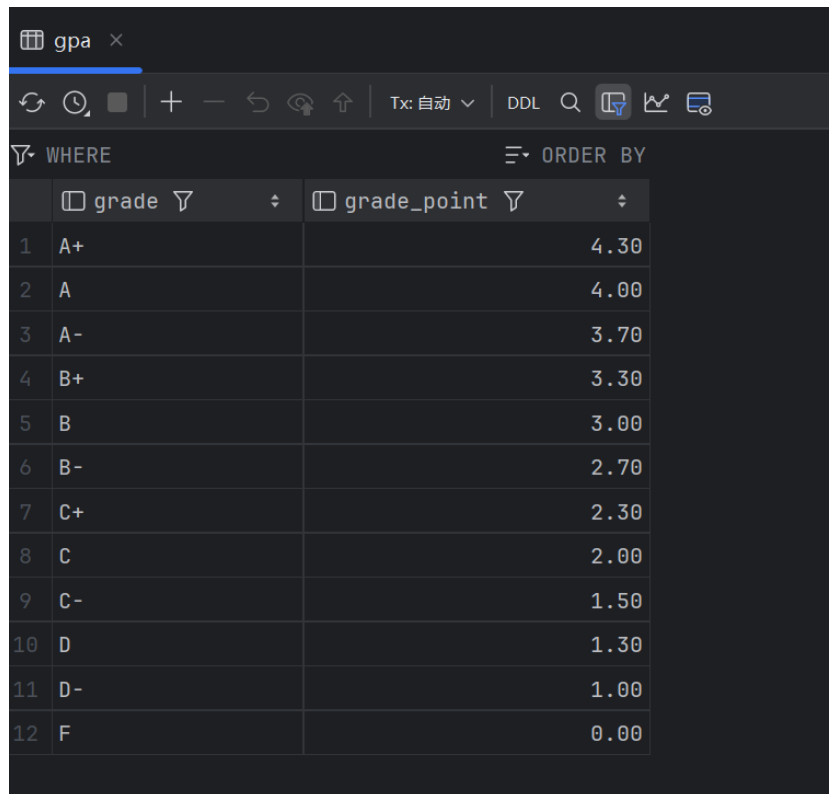
图 14: 综合处理

可以看到:

Connect to database mysql successfully !

且 GPA 相关的数据已经输入成功

再来数据库内查看：



	grade	grade_point
1	A+	4.30
2	A	4.00
3	A-	3.70
4	B+	3.30
5	B	3.00
6	B-	2.70
7	C+	2.30
8	C	2.00
9	C-	1.50
10	D	1.30
11	D-	1.00
12	F	0.00

图 15: 综合结果

处理成功

3.4 JDBC 小项目作业

备注：

以下的实验运行截图中，若无特殊标记，都默认查询 Manber(ID: 1000) 这位同学的相关课程与绩点信息。

3.4.1 数据库连接与用户登录

1. 连接 SQL（实验报告 1）中使用的 college 数据库。
2. 用户输入登录 ID 和密码，建立数据库连接。
3. 若连接失败，提示错误信息并允许用户重试。

连接数据库

```

1      while (true) {
2          try {
3              System.out.print("输入数据库用户名:");
4              String user = scanner.nextLine();
5              System.out.print("输入数据库密码:");
6              String pass = scanner.nextLine();
7              conn = DriverManager.getConnection(URL, user, pass);
8              System.out.println("数据库连接成功!");
9              break;
10         } catch (SQLException e) {
11             System.out.println("数据库连接失败, 请重试!");
12         }
13     }

```

以上代码用于建立数据库连接, 并提示用户输入用户名和密码进行登录
运行结果如下所示:

```

"C:\Program Files\Microsoft\jdk-17.0.8.101-hotspot\
输入数据库用户名: root
输入数据库密码: GUyiwei_)*@@
数据库连接成功!
请输入查询的学生姓名子串, 输入quit退出: |

```

输入正确的账号密码

图 16: 正确登录

```

"C:\Program Files\Microsoft\jdk-17.0.8.101-hotspot\bin\
输入数据库用户名: root
输入数据库密码: 1234
数据库连接失败, 请重试!
输入数据库用户名:

```

这是错误的密码

图 17: 错误登录

3.4.2 学生信息查询

1. 连接成功后, 用户输入一个字符串, 查询名字中含有该子串的所有学生信息 (ID、姓名、院系、总学分)。

输入学生姓名

```

1      try {
2          while (true) {
3              System.out.print("请输入查询的学生姓名子串, 输入quit退出:");
4              String nameSubstr = scanner.nextLine();
5              if (nameSubstr.equals("quit")) {
6                  break;
7              }
8              searchStudentsByName(conn, nameSubstr); // 调用函数
9          }
10     } catch (SQLException e) {
11         System.out.println("查询失败: " + e.getMessage());

```

12 }

运行结果如下所示：

```
请输入查询的学生姓名子串，输入quit退出：man
ID: 1000, Name: Manber, Dept_name: Civil Eng., Tot_credits: 39
ID: 11202, Name: Heckman, Dept_name: Math, Tot_credits: 120
ID: 16405, Name: Rahman, Dept_name: Languages, Tot_credits: 5
ID: 17769, Name: Pearlman, Dept_name: Biology, Tot_credits: 45
ID: 18752, Name: Schulman, Dept_name: Civil Eng., Tot_credits: 102
ID: 22532, Name: Silverman, Dept_name: History, Tot_credits: 120
ID: 34158, Name: Mantzo, Dept_name: Astronomy, Tot_credits: 127
```

图 18: 查询子串-man

```
请输入查询的学生姓名子串，输入quit退出：zel
ID: 10033, Name: Zelty, Dept_name: Mech. Eng., Tot_credits: 60
ID: 27956, Name: Watzel, Dept_name: Psychology, Tot_credits: 53
ID: 35881, Name: A-zel, Dept_name: Cybernetics, Tot_credits: 99
ID: 45359, Name: Zelek, Dept_name: Marketing, Tot_credits: 79
ID: 72485, Name: Wetzel, Dept_name: Finance, Tot_credits: 33
ID: 81175, Name: Zelek, Dept_name: Biology, Tot_credits: 0
ID: 92949, Name: Retzel, Dept_name: Pol. Sci., Tot_credits: 125
```

图 19: 查询子串-zel

2. 若查询结果为空，提示用户无相关学生并允许重新输入。

查询名字中含有该子串的所有学生信息

```
1      private static void searchStudentsByName(Connection conn, String
2          nameSubstr) throws SQLException {
3          String query = "SELECT ID, name, dept_name, tot_cred FROM
4              student WHERE name LIKE ?";
5          try (PreparedStatement stmt = conn.prepareStatement(query)) {
6              stmt.setString(1, "%" + nameSubstr + "%");
7              ResultSet rs = stmt.executeQuery();
8              boolean found = false;
9              while (rs.next()) {
10                  found = true;
11                  System.out.println("ID: " + rs.getInt("ID") + ", Name: "
12                      + rs.getString("name") + ", Dept_name: " + rs.
13                          getString("dept_name") + ", Tot_credits: " + rs.
14                              getInt("tot_cred"));
15              }
16              if (!found) {
17                  System.out.println("未找到相关学生。");
18              } else {
19                  searchStudentById(conn);
20              }
21          }
22      }
```

```

请输入查询的学生姓名子串，输入quit退出： 10
未找到相关学生。
请输入查询的学生姓名子串，输入quit退出： |

```

图 20: 查询子串错误-无相关学生

3. 用户输入一个整数（0~99999），查找 ID 与之匹配的学生信息。

查找 ID 与之匹配的学生信息

```

1      private static void searchStudentById(Connection conn) throws
        SQLException {
2          Scanner scanner = new Scanner(System.in);
3          System.out.print("请输入要查询的学生ID(0~99999):");
4          int id = scanner.nextInt();
5          scanner.nextLine();
6          String query = "SELECT ID, name, dept_name, tot_cred FROM
            student WHERE ID=?";
7          try (PreparedStatement stmt = conn.prepareStatement(query)) {
8              stmt.setInt(1, id);
9              ResultSet rs = stmt.executeQuery();
10             if (rs.next()) {
11                 System.out.println("ID:" + rs.getInt("ID") + ", Name:"
                    + rs.getString("name") + ", Dept:" + rs.getString(
                        "dept_name") + ", Credits:" + rs.getInt("tot_cred"
                    ));
12                 searchStudentCourses(conn, id);
13             } else {
14                 System.out.println("无该学生，请重试。");
15             }
16         }
17     }

```

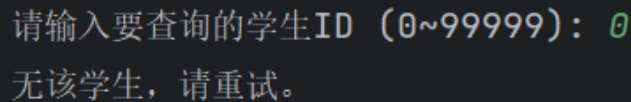
运行结果如下图所示：

```

请输入要查询的学生ID (0~99999): 1000
ID: 1000, Name: Manber, Dept: Civil Eng., Credits: 39

```

图 21: 查询 ID



请输入要查询的学生ID (0~99999): 0
无该学生, 请重试。

图 22: 查询 ID 错误-未找到

3.4.3 学生课程信息查询

1. 若找到学生信息, 用户可输入 1 以查询该学生所修读的所有课程信息, 包括:

- 课程 ID
- 上课年份
- 上课学期
- 课程名称
- 开课院系
- 成绩等级
- 课程学分数

2. 若用户输入 0, 则停止查询。

学生课程信息查询

```

1      private static void searchStudentCourses(Connection conn, int studentId)
        throws SQLException {
2          Scanner scanner = new Scanner(System.in);
3          System.out.print("输入1查看该学生修读的课程信息, 输入0退出:");
4          if (scanner.nextInt() != 1) return;
5          scanner.nextLine();
6
7          String query = "SELECT takes.course_id, year, semester, title, dept_name
                           , grade, credits FROM takes JOIN course ON takes.course_id = course.
                           course_id WHERE ID = ?";
8          try (PreparedStatement stmt = conn.prepareStatement(query)) {
9              stmt.setInt(1, studentId);
10             ResultSet rs = stmt.executeQuery();
11             boolean found = false;
12             while (rs.next()) {
13                 found = true;
14                 System.out.println("课程ID:" + rs.getString("course_id") + ",
                                     年份:" + rs.getInt("year") + ", 学期:" + rs.getString("
                                     semester") + ", 课程名:" + rs.getString("title") + ", 院系:
                                     " + rs.getString("dept_name") + ", 成绩:" + rs.getString("
                                     grade") + ", 学分:" + rs.getInt("credits"));

```



```

15         }
16         if (!found) {
17             System.out.println("该学生没有修读课程。");
18         } else {
19             calculateGPA(conn, studentId);
20         }
21     }
22 }

```

运行结果如下图所示（多余部分省略）：

```

输入1查看该学生修读的课程信息，输入0退出： 1
课程ID: 239, 年份: 2006, 学期: Fall, 课程名: The Music of the Ramones, 院系: Physics, 成绩: C, 学分: 4
课程ID: 319, 年份: 2003, 学期: Spring, 课程名: World History, 院系: Finance, 成绩: B+, 学分: 4
课程ID: 362, 年份: 2005, 学期: Fall, 课程名: Embedded Systems, 院系: Finance, 成绩: B+, 学分: 4
课程ID: 493, 年份: 2010, 学期: Spring, 课程名: Music of the 50s, 院系: Geology, 成绩: A-, 学分: 3
课程ID: 571, 年份: 2004, 学期: Spring, 课程名: Plastics, 院系: Comp. Sci., 成绩: C+, 学分: 4
课程ID: 642, 年份: 2004, 学期: Fall, 课程名: Video Gaming, 院系: Psychology, 成绩: C-, 学分: 3
课程ID: 663, 年份: 2005, 学期: Spring, 课程名: Geology, 院系: Psychology, 成绩: C+, 学分: 3
课程ID: 696, 年份: 2002, 学期: Spring, 课程名: Heat Transfer, 院系: Marketing, 成绩: B, 学分: 4
课程ID: 748, 年份: 2003, 学期: Fall, 课程名: Tort Law, 院系: Cybernetics, 成绩: A, 学分: 4
课程ID: 802, 年份: 2003, 学期: Spring, 课程名: African History, 院系: Cybernetics, 成绩: C+, 学分: 3
课程ID: 959, 年份: 2006, 学期: Fall, 课程名: Bacteriology, 院系: Physics, 成绩: B+, 学分: 4
课程ID: 962, 年份: 2008, 学期: Spring, 课程名: Animal Behavior, 院系: Psychology, 成绩: B-, 学分: 3
课程ID: 972, 年份: 2009, 学期: Spring, 课程名: Greek Tragedy, 院系: Psychology, 成绩: B+, 学分: 4

```

图 23: 查询课程信息

3.4.4 计算平均绩点（GPA）

1. 若学生有修读课程，用户可输入 1 计算该学生的平均绩点（GPA）并显示。
2. 若用户输入 0，则停止查询。

这里采用 $\text{SUM}(\text{学分} \times \text{绩点}) / \text{SUM}(\text{学分})$ 的方法来计算平均绩点

计算平均绩点 (GPA)

```

1     private static void calculateGPA(Connection conn, int studentId) throws
        SQLException {
2         Scanner scanner = new Scanner(System.in);
3         System.out.print("输入1计算该学生的平均绩点，输入0退出:");
4         if (scanner.nextInt() != 1) return;
5         scanner.nextLine();
6
7         String query = "SELECT grade_point, credits FROM (takes JOIN course ON
            takes.course_id = course.course_id) JOIN gpa ON gpa.grade = TRIM(
            takes.grade) WHERE ID = ";
8         try (PreparedStatement stmt = conn.prepareStatement(query)) {
9             stmt.setInt(1, studentId);

```

```
10         ResultSet rs = stmt.executeQuery();
11         double totalPoints = 0;
12         int totalCredits = 0;
13         while (rs.next()) {
14             float grade_point = rs.getFloat("grade_point");
15             int credits = rs.getInt("credits");
16             totalCredits += credits;
17             totalPoints += grade_point * credits;
18         }
19         if (totalCredits == 0) {
20             System.out.println("该学生无可计算绩点的课程。");
21         } else {
22             double gpa = totalPoints / totalCredits;
23             System.out.printf("该学生的平均绩点(两位小数)为: %.2f\n", gpa);
24         }
25     }
26 }
```

TRIM 函数的使用

在这里我使用了 TRIM 函数，因为在 takes 表格中，A,B,C 等绩点，存储格式是"A ", "B " 这样会导致 JOIN 的时候由于条件不符合而被忽略，所以使用 TRIM 函数来取消空格带来的影响

```
输入1计算该学生的平均绩点，输入0退出： 1
该学生的平均绩点(两位小数)为： 2.88
```

图 24: 计算平均绩点

3.4.5 异常处理要求

1. 连接数据库成功后，所有数据库操作需捕获异常并进行处理，程序不应因异常停止运行。
2. 若用户输入不合法数据，需给出相应提示，并允许重新输入。

所有的执行过程都用 try-catch 来包裹，可以做出自定义的提示，而不是报错中断，具体代码与运行结果上面已经展示，此处不再展示

4 存在的问题及解决方案

在实验过程中，可能会遇到以下问题及相应的解决方案：

- 数据库连接失败：

- 确保 MySQL 服务器已启动。
- 检查 JDBC URL、用户名和密码是否正确。
- 确保 MySQL 允许远程连接（修改 my.cnf 或 my.ini 配置文件）。
- **SQL 语句执行错误：**
 - 使用 PreparedStatement 避免 SQL 注入问题。
 - 通过日志调试 SQL 语句是否正确执行。
- **编码问题（如中文乱码）：**
 - 在数据库连接 URL 末尾添加 useUnicode=true&characterEncoding=UTF-8。
 - 设置 MySQL 数据库的字符集为 UTF-8。
- **驱动加载失败：**
 - 确保 MySQL JDBC 驱动包已正确导入。
 - 使用 'Class.forName("com.mysql.cj.jdbc.Driver")' 确保驱动正确加载。
- **takes 与 gpa 里的 grade 格式不一致：**
 - 在 takes 表格中，A,B,C 等绩点，存储格式是"A ","B "
 - 在 gpa 表格中，A,B,C 等绩点，存储格式是"A","B"
 - 使用 TRIM 来消除空格带来的影响。

5 实验小结

通过本次实验，学生学习了 JDBC 技术的基础知识，并成功实现了 Java 应用程序与 MySQL 数据库的连接与交互。实验过程中遇到的问题，如数据库连接失败、SQL 语句错误等，都得到了有效解决。通过本实验，学生对数据库操作有了更深入的理解，并为后续的数据库应用开发打下了良好的基础。此外，学生还认识到了编写健壮、安全的数据库交互代码的重要性，为后续的编程实践提供了宝贵的经验。