

# 《数据库系统及应用实践》课程实验报告

## 实验 4: JDBC 数据库应用

姓 名: 李鹏达 学 号: 10225101460 完成日期: 2024 年 5 月 11 日

### 1 实验目标

1. 学习通过 JDBC 接口访问关系数据库的基本方法;
2. 能够根据要求编写程序通过 JDBC 接口访问关系数据库管理系统;

### 2 实验过程记录

#### 2.1 安装和配置 MySQL 的 JDBC 驱动

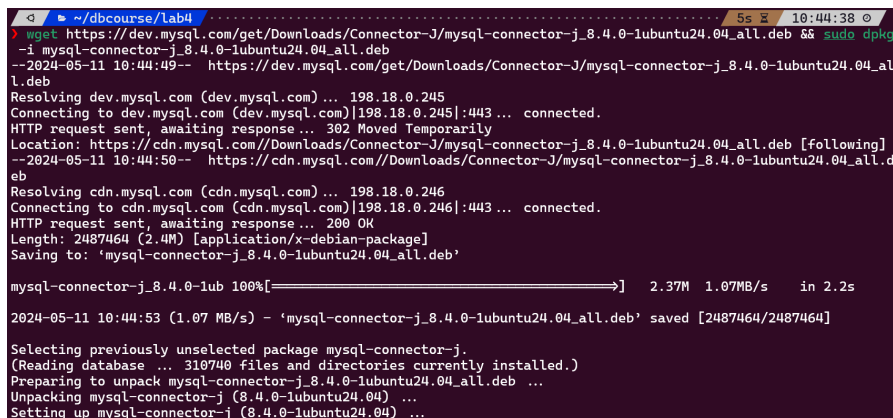
##### 2.1.1 获取 Connector/J 8.3

MySQL provides connectivity for client applications developed in the Java programming language with MySQL Connector/J. Connector/J implements the Java Database Connectivity (JDBC) API, as well as a number of value-adding extensions of it.

实验所用操作系统为 Ubuntu 24.04 LTS on Windows 10 x86\_64(Kernel: 5.15.146.1-microsoft-standard-WSL2), 使用以下命令来安装 MySQL 的 JDBC 驱动:

```
1 wget https://dev.mysql.com/get/Downloads/Connector-J/mysql-connector-j_8.4.0-1
  ubuntu24.04_all.deb
2 sudo dpkg -i mysql-connector-j_8.4.0-1ubuntu24.04_all.deb
```

安装过程如下图所示:



```
q ~/dbcourse/lab4 5s 10:44:38
> wget https://dev.mysql.com/get/Downloads/Connector-J/mysql-connector-j_8.4.0-1ubuntu24.04_all.deb && sudo dpkg
-i mysql-connector-j_8.4.0-1ubuntu24.04_all.deb
--2024-05-11 10:44:49-- https://dev.mysql.com/get/Downloads/Connector-J/mysql-connector-j_8.4.0-1ubuntu24.04_al
l.deb
Resolving dev.mysql.com (dev.mysql.com)... 198.18.0.245
Connecting to dev.mysql.com (dev.mysql.com)|198.18.0.245|:443... connected.
HTTP request sent, awaiting response... 302 Moved Temporarily
Location: https://cdn.mysql.com//Downloads/Connector-J/mysql-connector-j_8.4.0-1ubuntu24.04_all.deb [following]
--2024-05-11 10:44:50-- https://cdn.mysql.com//Downloads/Connector-J/mysql-connector-j_8.4.0-1ubuntu24.04_all.d
eb
Resolving cdn.mysql.com (cdn.mysql.com)... 198.18.0.246
Connecting to cdn.mysql.com (cdn.mysql.com)|198.18.0.246|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2487464 (2.4M) [application/x-debian-package]
Saving to: 'mysql-connector-j_8.4.0-1ubuntu24.04_all.deb'

mysql-connector-j_8.4.0-1ub 100%[====>] 2.37M 1.07MB/s in 2.2s

2024-05-11 10:44:53 (1.07 MB/s) - 'mysql-connector-j_8.4.0-1ubuntu24.04_all.deb' saved [2487464/2487464]

Selecting previously unselected package mysql-connector-j.
(Reading database ... 310740 files and directories currently installed.)
Preparing to unpack mysql-connector-j_8.4.0-1ubuntu24.04_all.deb ...
Unpacking mysql-connector-j (8.4.0-1ubuntu24.04) ...
Setting up mysql-connector-j (8.4.0-1ubuntu24.04) ...
```

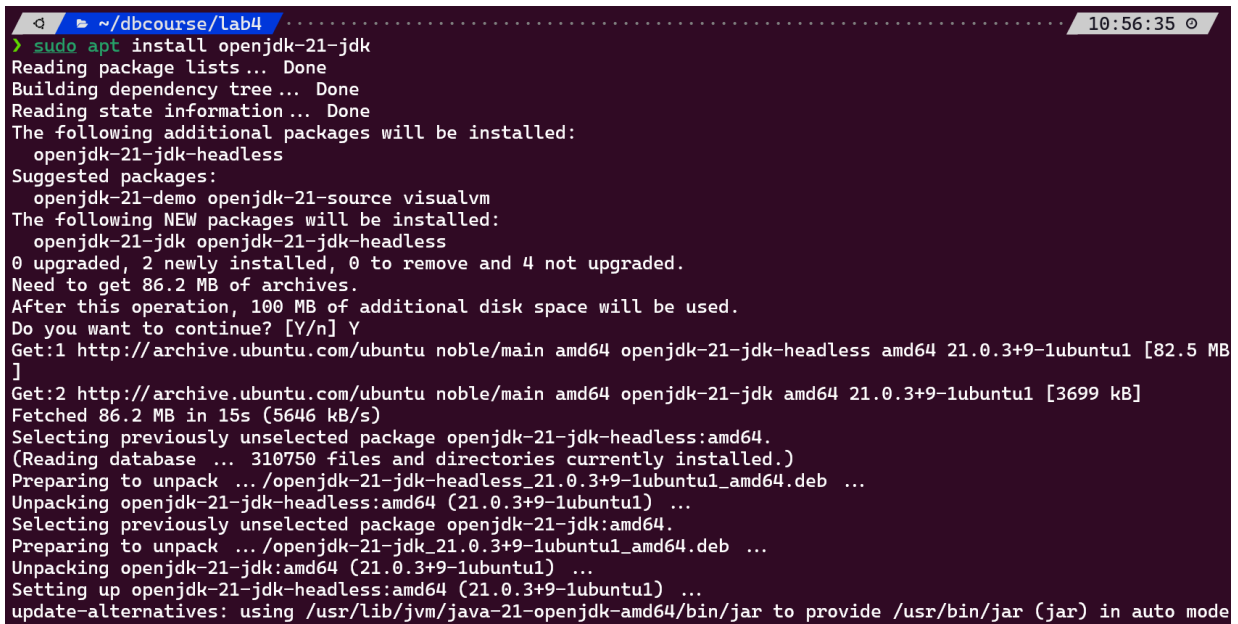
图 1: 安装 MySQL 的 JDBC 驱动

### 2.1.2 安装 JAVA 开发环境

在这里，我们选择安装 `openjdk-21`，使用以下命令来安装：

```
1 sudo apt install openjdk-21-jdk
```

安装过程如下图所示：



```
> sudo apt install openjdk-21-jdk
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  openjdk-21-jdk-headless
Suggested packages:
  openjdk-21-demo openjdk-21-source visualvm
The following NEW packages will be installed:
  openjdk-21-jdk openjdk-21-jdk-headless
0 upgraded, 2 newly installed, 0 to remove and 4 not upgraded.
Need to get 86.2 MB of archives.
After this operation, 100 MB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://archive.ubuntu.com/ubuntu noble/main amd64 openjdk-21-jdk-headless amd64 21.0.3+9-1ubuntu1 [82.5 MB]
Get:2 http://archive.ubuntu.com/ubuntu noble/main amd64 openjdk-21-jdk amd64 21.0.3+9-1ubuntu1 [3699 kB]
Fetched 86.2 MB in 15s (5646 kB/s)
Selecting previously unselected package openjdk-21-jdk-headless:amd64.
(Reading database ... 310750 files and directories currently installed.)
Preparing to unpack .../openjdk-21-jdk-headless_21.0.3+9-1ubuntu1_amd64.deb ...
Unpacking openjdk-21-jdk-headless:amd64 (21.0.3+9-1ubuntu1) ...
Selecting previously unselected package openjdk-21-jdk:amd64.
Preparing to unpack .../openjdk-21-jdk_21.0.3+9-1ubuntu1_amd64.deb ...
Unpacking openjdk-21-jdk:amd64 (21.0.3+9-1ubuntu1) ...
Setting up openjdk-21-jdk-headless:amd64 (21.0.3+9-1ubuntu1) ...
update-alternatives: using /usr/lib/jvm/java-21-openjdk-amd64/bin/jar to provide /usr/bin/jar (jar) in auto mode
```

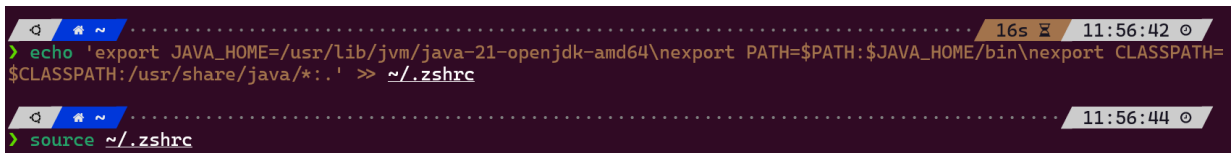
图 2: 安装 JAVA 开发环境

### 2.1.3 配置环境变量

我们使用以下命令来配置环境变量（在这里，我是用的 shell 是 `zsh`）：

```
1 echo 'export JAVA_HOME=/usr/lib/jvm/java-21-openjdk-amd64\nexport PATH=$PATH:\n  $JAVA_HOME/bin\nexport CLASSPATH=$CLASSPATH:/usr/share/java/*:.' >> ~/.zshrc
2 source ~/.zshrc
```

配置过程如下图所示：



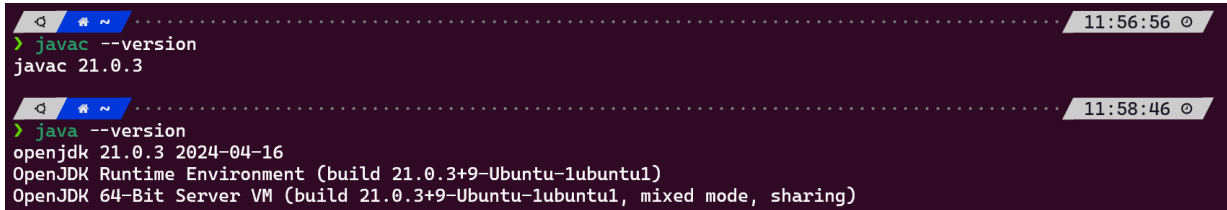
```
> echo 'export JAVA_HOME=/usr/lib/jvm/java-21-openjdk-amd64\nexport PATH=$PATH:$JAVA_HOME/bin\nexport CLASSPATH=$CLASSPATH:/usr/share/java/*:.' >> ~/.zshrc
> source ~/.zshrc
```

图 3: 配置环境变量

使用以下命令来检查系统中 java 的编译和运行环境：

```
1 javac --version
2 java --version
```

结果如下图所示：



```
> javac --version
javac 21.0.3

> java --version
openjdk 21.0.3 2024-04-16
OpenJDK Runtime Environment (build 21.0.3+9-Ubuntu-1ubuntu1)
OpenJDK 64-Bit Server VM (build 21.0.3+9-Ubuntu-1ubuntu1, mixed mode, sharing)
```

图 4: 检查系统中 java 的编译和运行环境

## 2.2 JDBC 基本操作

### 2.2.1 与数据库建立连接

在当前工作文件夹中创建一个名为 jdbcConnect.java 的文件，其中内容如下：

```
1 import java.sql.Connection;
2 import java.sql.DriverManager;
3
4 public class jdbcConnect {
5     public static void main(String args[]) {
6         String dbUserName = "root";
7         String dbPassword = "password";
8         String dbUrl = "jdbc:mysql://localhost:53306/dbcourse?useSSL=false&
9             allowPublicKeyRetrieval=true";
10        Connection c = null;
11        try {
12            Class.forName("com.mysql.cj.jdbc.Driver");
13            c = DriverManager.getConnection(dbUrl, dbUserName, dbPassword);
14        } catch (Exception e) {
15            e.printStackTrace();
16            System.err.println(e.getClass().getName() + ": " + e.getMessage());
17            System.exit(0);
18        }
19        System.out.println("Connect to the MySQL database successfully !");
20    }
}
```

接下来，使用以下命令来编译和运行该程序：

```
1 javac jdbcConnect.java && java jdbcConnect
```

运行结果如下图所示：



```
> javac jdbcConnect.java && java jdbcConnect
Connect to the MySQL database successfully !
```

图 5: 与数据库建立连接

### 2.2.2 在数据库中创建数据表

在当前工作文件夹中创建一个名为 `jdbcCreate.java` 的文件，其中内容如下：

```
1 import java.sql.Connection;
2 import java.sql.DriverManager;
3 import java.sql.Statement;
4
5 public class jdbcCreate {
6     public static void main(String args[]) {
7         String dbUserName = "root";
8         String dbPassword = "password";
9         String dbUrl = "jdbc:mysql://localhost:53306/dbcourse?useSSL=false&
            allowPublicKeyRetrieval=true";
10        Connection c = null;
11        Statement stmt = null;
12        try {
13            Class.forName("com.mysql.cj.jdbc.Driver");
14            c = DriverManager.getConnection(dbUrl, dbUserName, dbPassword);
15            System.out.println("Connect to the MySQL database successfully!");
16            stmt = c.createStatement();
17            String sql = ""
18                CREATE TABLE employee (
19                    id INT,
20                    name VARCHAR(20) NOT NULL,
21                    age INT NOT NULL,
22                    address VARCHAR(50),
23                    salary REAL,
24                    PRIMARY KEY (id)
25                )
26            "";
27            stmt.executeUpdate(sql);
28            stmt.close();
29            c.close();
```

```

30         } catch (Exception e) {
31             System.err.println(e.getClass().getName() + ": " + e.getMessage());
32             System.exit(0);
33         }
34         System.out.println("Create table employee successfully !");
35     }
36 }

```

使用以下命令来编译和运行该程序：

```
1 javac jdbcCreate.java && java jdbcCreate
```

运行结果如下图所示：



```

~/dbcourse/lab4 15:01:49
> javac jdbcCreate.java && java jdbcCreate
Connect to the MySQL database successfully!
Create table employee successfully!

```

图 6: 在数据库中创建数据表

### 2.2.3 向数据表中插入数据

在当前工作文件夹中创建一个名为 jdbcInsert.java 的文件，其中内容如下：

```

1  import java.sql.Connection;
2  import java.sql.DriverManager;
3  import java.sql.Statement;
4
5  public class jdbcInsert {
6      public static void main(String args[]) {
7          String dbUserName = "root";
8          String dbPassword = "password";
9          String dbUrl = "jdbc:mysql://localhost:53306/dbcourse?useSSL=false&
              allowPublicKeyRetrieval=true";
10         Connection c = null;
11         Statement stmt = null;
12         try {
13             Class.forName("com.mysql.cj.jdbc.Driver");
14             c = DriverManager.getConnection(dbUrl, dbUserName, dbPassword);
15             c.setAutoCommit(false);
16             System.out.println("Connect to the MySQL database successfully!");
17             stmt = c.createStatement();
18             String sql = "INSERT INTO employee VALUES " +
19                 "(1, 'Zhang', 30, '2075 Kongjiang Road', 20000.00)";
20             stmt.executeUpdate(sql);

```

```

21         sql = "INSERT INTO employee VALUES " +
22             "(2, 'Luan', 25, '3663 Zhongshan Road(N)', 15000.00 );";
23         stmt.executeUpdate(sql);
24         sql = "INSERT INTO employee VALUES " +
25             "(3, 'Hu', 23, '1234 Beijing Road(E)', 14000.00);";
26         stmt.executeUpdate(sql);
27         sql = "INSERT INTO employee VALUES " +
28             "(4, 'Jin', 24, '666 Nanjing Road(W)', 19000.00);";
29         stmt.executeUpdate(sql);
30         sql = "INSERT INTO employee VALUES " +
31             "(5, 'Yi', 24, '100 Renmin Road', 18800.00 );";
32         stmt.executeUpdate(sql);
33         stmt.close();
34         c.commit();
35         c.close();
36     } catch (Exception e) {
37         System.err.println(e.getClass().getName() + ": " + e.getMessage());
38         System.exit(0);
39     }
40     System.out.println("5 records inserted successfully !");
41 }
42 }

```

使用以下命令来编译和运行该程序：

```
1 javac jdbcInsert.java && java jdbcInsert
```

运行结果如下图所示：



```

~/dbcourse/lab4 15:12:50
> javac jdbcInsert.java && java jdbcInsert
Connect to the MySQL database successfully!
5 records inserted successfully !

```

图 7: 向数据表中插入数据

#### 2.2.4 从数据表中查询数据

在当前工作文件夹中创建一个名为 jdbcSelect.java 的文件，其中内容如下：

```

1 import java.sql.Connection;
2 import java.sql.DriverManager;
3 import java.sql.ResultSet;
4 import java.sql.Statement;
5
6 public class jdbcSelect {

```

```
7 public static void main(String args[]) {
8     String dbUserName = "root";
9     String dbPassword = "password";
10    String dbUrl = "jdbc:mysql://localhost:53306/dbcourse?useSSL=false&
        allowPublicKeyRetrieval=true";
11    Connection c = null;
12    Statement stmt = null;
13    try {
14        Class.forName("com.mysql.cj.jdbc.Driver");
15        c = DriverManager.getConnection(dbUrl, dbUserName, dbPassword);
16        c.setAutoCommit(false);
17        System.out.println("Connect to the MySQL database successfully!");
18        stmt = c.createStatement();
19        ResultSet rs = stmt.executeQuery("SELECT * FROM employee;");
20        while (rs.next()) {
21            int id = rs.getInt("id");
22            String name = rs.getString("name");
23            int age = rs.getInt("age");
24            String address = rs.getString("address");
25            float salary = rs.getFloat("salary");
26            System.out.println("ID = " + id);
27            System.out.println("NAME = " + name);
28            System.out.println("AGE = " + age);
29            System.out.println("ADDRESS = " + address);
30            System.out.println("SALARY = " + salary);
31            System.out.println();
32        }
33        rs.close();
34        stmt.close();
35        c.close();
36    } catch (Exception e) {
37        System.err.println(e.getClass().getName() + ": " + e.getMessage());
38        System.exit(0);
39    }
40    System.out.println("Operation done successfully !");
41 }
42 }
```

使用以下命令来编译和运行该程序：

```
1 javac jdbcSelect.java && java jdbcSelect
```

运行结果如下图所示：

```
~/dbcourse/lab4 15:22:51
> javac jdbcSelect.java && java jdbcSelect
Connect to the MySQL database successfully!
ID = 1
NAME = Zhang
AGE = 30
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 = 1234 Beijing Road(E)
SALARY = 14000.0

ID = 4
NAME = Jin
AGE = 24
ADDRESS = 666 Nanjing Road(W)
SALARY = 19000.0

ID = 5
NAME = Yi
AGE = 24
ADDRESS = 100 Renmin Road
SALARY = 18800.0

Operation done successfully !
```

图 8: 从数据表中查询数据

### 2.2.5 更新数据表中的数据

在当前工作文件夹中创建一个名为 `jdbcUpdate.java` 的文件，其中内容如下：

```
1  import java.sql.Connection;
2  import java.sql.DriverManager;
3  import java.sql.ResultSet;
4  import java.sql.Statement;
5
6  public class jdbcUpdate {
7      public static void main(String args[]) {
8          String dbUserName = "root";
9          String dbPassword = "password";
10         String dbUrl = "jdbc:mysql://localhost:53306/dbcourse?useSSL=false&
            allowPublicKeyRetrieval=true";
11         Connection c = null;
12         Statement stmt = null;
13         try {
14             Class.forName("com.mysql.cj.jdbc.Driver");
15             c = DriverManager.getConnection(dbUrl, dbUserName, dbPassword);
16             c.setAutoCommit(false);
17             System.out.println("Connect to the MySQL database successfully!");
```



```
18      stmt = c.createStatement();
19      String sql = "UPDATE employee set SALARY = 32000.00 where ID = 1;";
20      stmt.executeUpdate(sql);
21      c.commit();
22      System.out.println("1 record is updated successfully");
23      ResultSet rs = stmt.executeQuery("SELECT * FROM employee;");
24      while (rs.next()) {
25          int id = rs.getInt("id");
26          String name = rs.getString("name");
27          int age = rs.getInt("age");
28          String address = rs.getString("address");
29          float salary = rs.getFloat("salary");
30          System.out.println("ID = " + id);
31          System.out.println("NAME = " + name);
32          System.out.println("AGE = " + age);
33          System.out.println("ADDRESS = " + address);
34          System.out.println("SALARY = " + salary);
35          System.out.println();
36      }
37      rs.close();
38      stmt.close();
39      c.close();
40      } catch (Exception e) {
41          System.err.println(e.getClass().getName() + ": " + e.getMessage());
42          System.exit(0);
43      }
44      System.out.println("Operation done successfully !");
45  }
46 }
```

使用以下命令来编译和运行该程序：

```
1 javac jdbcUpdate.java && java jdbcUpdate
```

运行结果如下图所示：

```
~/dbcourse/lab4 15:29:33
> javac jdbcUpdate.java && java jdbcUpdate
Connect to the MySQL database successfully!
1 record is updated successfully
ID = 1
NAME = Zhang
AGE = 30
ADDRESS = 2075 Kongjiang Road
SALARY = 32000.0

ID = 2
NAME = Luan
AGE = 25
ADDRESS = 3663 Zhongshan Road(N)
SALARY = 15000.0

ID = 3
NAME = Hu
AGE = 23
ADDRESS = 1234 Beijing Road(E)
SALARY = 14000.0

ID = 4
NAME = Jin
AGE = 24
ADDRESS = 666 Nanjing Road(W)
SALARY = 19000.0

ID = 5
NAME = Yi
AGE = 24
ADDRESS = 100 Renmin Road
SALARY = 18800.0

Operation done successfully !
```

图 9: 更新数据表中的数据

### 2.2.6 删除数据表中的数据

在当前工作文件夹中创建一个名为 `jdbcDelete.java` 的文件，其中内容如下：

```
1  import java.sql.Connection;
2  import java.sql.DriverManager;
3  import java.sql.ResultSet;
4  import java.sql.Statement;
5
6  public class jdbcDelete {
7      public static void main(String args[]) {
8          String dbUserName = "root";
9          String dbPassword = "password";
10         String dbUrl = "jdbc:mysql://localhost:53306/dbcourse?useSSL=false&
            allowPublicKeyRetrieval=true";
11         Connection c = null;
12         Statement stmt = null;
13         try {
14             Class.forName("com.mysql.cj.jdbc.Driver");
15             c = DriverManager.getConnection(dbUrl, dbUserName, dbPassword);
16             c.setAutoCommit(false);
17             System.out.println("Connect to the MySQL database successfully!");
```

```
18      stmt = c.createStatement();
19      String sql = "DELETE from employee where ID=2;";
20      stmt.executeUpdate(sql);
21      c.commit();
22      System.out.println("1 record is deleted successfully !");
23      ResultSet rs = stmt.executeQuery("SELECT * FROM employee;");
24      while (rs.next()) {
25          int id = rs.getInt("id");
26          String name = rs.getString("name");
27          int age = rs.getInt("age");
28          String address = rs.getString("address");
29          float salary = rs.getFloat("salary");
30          System.out.println("ID = " + id);
31          System.out.println("NAME = " + name);
32          System.out.println("AGE = " + age);
33          System.out.println("ADDRESS = " + address);
34          System.out.println("SALARY = " + salary);
35          System.out.println();
36      }
37      rs.close();
38      stmt.close();
39      c.close();
40      } catch (Exception e) {
41          System.err.println(e.getClass().getName() + ": " + e.getMessage());
42          System.exit(0);
43      }
44      System.out.println("Operation done successfully !");
45  }
46 }
```

使用以下命令来编译和运行该程序：

```
1 javac jdbcDelete.java && java jdbcDelete
```

运行结果如下图所示：

```
~/dbcourse/lab4 15:34:41
> javac jdbcDelete.java && java jdbcDelete
Connect to the MySQL database successfully!
1 record is deleted successfully !
ID = 1
NAME = Zhang
AGE = 30
ADDRESS = 2075 Kongjiang Road
SALARY = 32000.0

ID = 3
NAME = Hu
AGE = 23
ADDRESS = 1234 Beijing Road(E)
SALARY = 14000.0

ID = 4
NAME = Jin
AGE = 24
ADDRESS = 666 Nanjing Road(W)
SALARY = 19000.0

ID = 5
NAME = Yi
AGE = 24
ADDRESS = 100 Renmin Road
SALARY = 18800.0

Operation done successfully !
```

图 10: 删除数据表中的数据

### 2.2.7 使用 prepare 语句向数据表中批量插入（更新）数据

在当前工作文件夹中创建一个名为 DBTest.java 的文件，其中内容如下：

```
1  import java.sql.Connection;
2  import java.sql.DriverManager;
3  import java.sql.SQLException;
4  import java.sql.Statement;
5  import java.sql.PreparedStatement;
6
7  public class DBTest {
8      // Connect to database
9      public static Connection GetConnection(String username, String passwd) {
10         String driver = "com.mysql.cj.jdbc.Driver";
11         String sourceURL = "jdbc:mysql://localhost:53306/dbcourse?useSSL=false&
            allowPublicKeyRetrieval=true";
12         Connection conn = null;
13         try {
14             Class.forName(driver);
15             conn = DriverManager.getConnection(sourceURL, username, passwd);
16             System.out.println("Connect to database successfully!");
17         } catch (Exception e) {
18             e.printStackTrace();
19             return null;
20         }
21     }
```

```
21         return conn;
22     };
23
24     // Create table customer
25     public static void CreateTable(Connection conn) {
26         Statement stmt = null;
27         try {
28             stmt = conn.createStatement();
29             int rc = stmt.executeUpdate("CREATE TABLE customer" +
30                                     "(id INTEGER, name VARCHAR(20), PRIMARY KEY(id));");
31             System.out.println("Create table customer successfully!");
32             stmt.close();
33         } catch (SQLException e) {
34             if (stmt != null) {
35                 try {
36                     stmt.close();
37                 } catch (SQLException e1) {
38                     e1.printStackTrace();
39                 }
40             }
41             e.printStackTrace();
42         }
43     }
44
45     // Execute prepare statement, batch insert
46     public static void BatchInsertData(Connection conn) {
47         PreparedStatement pst = null;
48         try {
49             pst = conn.prepareStatement("INSERT INTO customer VALUES (?,?)");
50             for (int i = 0; i < 10; i++) {
51                 pst.setInt(1, i);
52                 pst.setString(2, "ECNUer " + i);
53                 pst.addBatch();
54             }
55             pst.executeBatch();
56             System.out.println("Insert 10 record in 1 batch successfully!");
57             pst.close();
58         } catch (SQLException e) {
59             if (pst != null) {
60                 try {
61                     pst.close();
62                 } catch (SQLException e1) {
63                     e1.printStackTrace();
64                 }
65             }
66         }
67     }
68 }
```

```
65         }
66         e.printStackTrace();
67     }
68 }
69
70 // Update by prepare statement
71 public static void ExecPreparedSQL(Connection conn) {
72     PreparedStatement pstmt = null;
73     try {
74         pstmt = conn
75             .prepareStatement("UPDATE customer SET name = ? WHERE id = 1");
76         pstmt.setString(1, "Wang");
77         int rowcount = pstmt.executeUpdate();
78         System.out.println("Update customer 1's name successfully!");
79         pstmt.close();
80     } catch (SQLException e) {
81         if (pstmt != null) {
82             try {
83                 pstmt.close();
84             } catch (SQLException e1) {
85                 e1.printStackTrace();
86             }
87         }
88         e.printStackTrace();
89     }
90 }
91
92 /**
93  * Main program
94  */
95 public static void main(String[] args) {
96     Connection conn = GetConnection("root", "password");
97     CreateTable(conn);
98     BatchInsertData(conn);
99     ExecPreparedSQL(conn);
100    try {
101        conn.close();
102    } catch (SQLException e) {
103        e.printStackTrace();
104    }
105 }
106 }
```

使用以下命令来编译和运行该程序：

```
1 javac DBTest.java && java DBTest
```

运行结果如下图所示：

```
> javac DBTest.java && java DBTest
Connect to database successfully!
Create table customer successfully!
Insert 10 record in 1 batch successfully!
Update customer 1's name successfully!
```

图 11: 使用 prepare 语句向数据表中批量插入（更新）数据

## 2.3 JDBC 应用

### 2.3.1 Teaching Record Management System

Write a Java program that allows university administrators to print the teaching record of an instructor.

- (a) Start by having the user input the login **ID** and **password**; then open the proper connection.
- (b) The user is asked next for a search substring and the system returns (**ID**, **name**) pairs of instructors whose names match the substring. Use the like ('%substring%') construct in SQL to do this. If the search comes back empty, allow continued searches until there is a nonempty result.
- (c) Then the user is asked to enter an **ID** number, which is a number between 0 and 99999. Once a valid number is entered, check if an instructor with that ID exists. If there is no instructor with the given ID, print a reasonable message and quit.
- (d) If the instructor has taught no courses, print a message saying that. Otherwise, print the teaching record for the instructor, showing the **department name**, **course identifier**, **course title**, **section number**, **semester**, **year**, and **total enrollment** (and sort those by dept name, course id, year, semester).

Test carefully for bad input. Make sure your SQL queries won't throw an exception. At login, exceptions may occur since the user might type a bad password, but catch those exceptions and allow the user to try again.

所写的 Java 程序如下：

```
1 import java.sql.Connection;
2 import java.sql.DriverManager;
3 import java.sql.PreparedStatement;
4 import java.sql.ResultSet;
5 import java.sql.SQLException;
6 import java.util.Scanner;
7
8 public class InstructorSystem {
```

```
9      public static final String url = "jdbc:mysql://localhost:53306/dbcourse?useSSL=
10         false&allowPublicKeyRetrieval=true";
11
12      public static Connection c;
13
14      public static Scanner sc = new Scanner(System.in);
15
16      public static final String spliter = "-----";
17
18      public static void login() {
19          try {
20              String id, password;
21              System.out.print("Enter your ID: ");
22              id = sc.nextLine();
23              System.out.print("Enter your password: ");
24              password = sc.nextLine();
25              Class.forName("com.mysql.cj.jdbc.Driver");
26              c = DriverManager.getConnection(url, id, password);
27          } catch (SQLException e) {
28              System.out.println("Invalid ID or password! Please try again!");
29              login();
30          } catch (Exception e) {
31              e.printStackTrace();
32              System.err.println(
33                  "Something went wrong! Please contact the administrator for help
34                  or try again later!(check the error message above)");
35              System.exit(1);
36          }
37      }
38
39      public static void searchInstructors() {
40          String searchString;
41          try {
42              System.out.print("Enter the name of the instructor you want to search
43                  for: ");
44              searchString = sc.nextLine();
45              PreparedStatement ps = c.prepareStatement("SELECT `ID`, `name` FROM `
46                  instructor` WHERE `name` LIKE ?");
47              ps.setString(1, "%" + searchString + "%");
48              ResultSet rs = ps.executeQuery();
49              if (!rs.isBeforeFirst()) {
50                  System.out.println("No instructor found! Please try again!");
51                  searchInstructors();
52              }
53          }
54      }
55  }
```



```
49         System.out.println("Search results:");
50         System.out.println(spliter);
51         while (rs.next()) {
52             System.out.println("ID: " + rs.getInt("id"));
53             System.out.println("Name: " + rs.getString("name"));
54             System.out.println(spliter);
55         }
56         rs.close();
57         ps.close();
58     } catch (Exception e) {
59         e.printStackTrace();
60         System.err.println(
61             "Something went wrong! Please contact the administrator for help
62             or try again later!(check the error message above)");
63         System.exit(1);
64     }
65 }
66
67 public static Integer checkInstructor() {
68     Integer id = null;
69     try {
70         System.out.print("Enter the ID of the instructor you select: ");
71         id = sc.nextInt();
72         PreparedStatement ps = c.prepareStatement("SELECT `ID`, `name` FROM `
73             instructor` WHERE `ID` = ?");
74         ps.setInt(1, id);
75         ResultSet rs = ps.executeQuery();
76         if (!rs.isBeforeFirst()) {
77             System.out.println("No instructor found! Quitting...");
78             System.exit(0);
79         }
80         rs.close();
81         ps.close();
82     } catch (SQLException e) {
83         e.printStackTrace();
84         System.err.println(
85             "Something went wrong! Please contact the administrator for help
86             or try again later!(check the error message above)");
87         System.exit(1);
88     }
89     return id;
90 }
91
92 public static void searchTeachingRecords(int id) {
```

```

90     try {
91         PreparedStatement ps = c.prepareStatement(
92             """
93             SELECT `course`.`dept_name`, `course`.`course_id`, `course`.`title`,
94             `section`.`sec_id`, `section`.`semester`, `section`.`year`,
95             COUNT(`takes`.`ID`) AS `total_enrollment`
96             FROM `teaches` JOIN `course` ON `teaches`.`course_id` = `course`.`
97             course_id`
98             JOIN `section` ON `teaches`.`course_id` = `section`.`course_id` AND
99             `teaches`.`sec_id` = `section`.`sec_id`
100             LEFT JOIN `takes` ON `teaches`.`course_id` = `takes`.`course_id` AND
101             `teaches`.`sec_id` = `takes`.`sec_id`
102             WHERE `teaches`.`ID` = ?
103             GROUP BY `course_id`, `sec_id`, `semester`, `year`
104             ORDER BY `dept_name`, `course_id`, `year`, `semester`""");
105         ps.setInt(1, id);
106         ResultSet rs = ps.executeQuery();
107         if (!rs.isBeforeFirst()) {
108             System.out.println("No teaching record found! Quitting...");
109             System.exit(0);
110         }
111         System.out.println("Teaching records for instructor with ID " + id + ":");
112         System.out.println(splitter);
113         while (rs.next()) {
114             System.out.println("Department: " + rs.getString("dept_name"));
115             System.out.println("Course ID: " + rs.getString("course_id"));
116             System.out.println("Title: " + rs.getString("title"));
117             System.out.println("Section ID: " + rs.getString("sec_id"));
118             System.out.println("Semester: " + rs.getString("semester"));
119             System.out.println("Year: " + rs.getInt("year"));
120             System.out.println("Total Enrollment: " + rs.getInt("
121                 total_enrollment"));
122             System.out.println(splitter);
123         }
124     } catch (Exception e) {
125         e.printStackTrace();
126         System.err.println(
127             "Something went wrong! Please contact the administrator for help
128             or try again later!(check the error message above)");
129         System.exit(1);
130     }
131 }

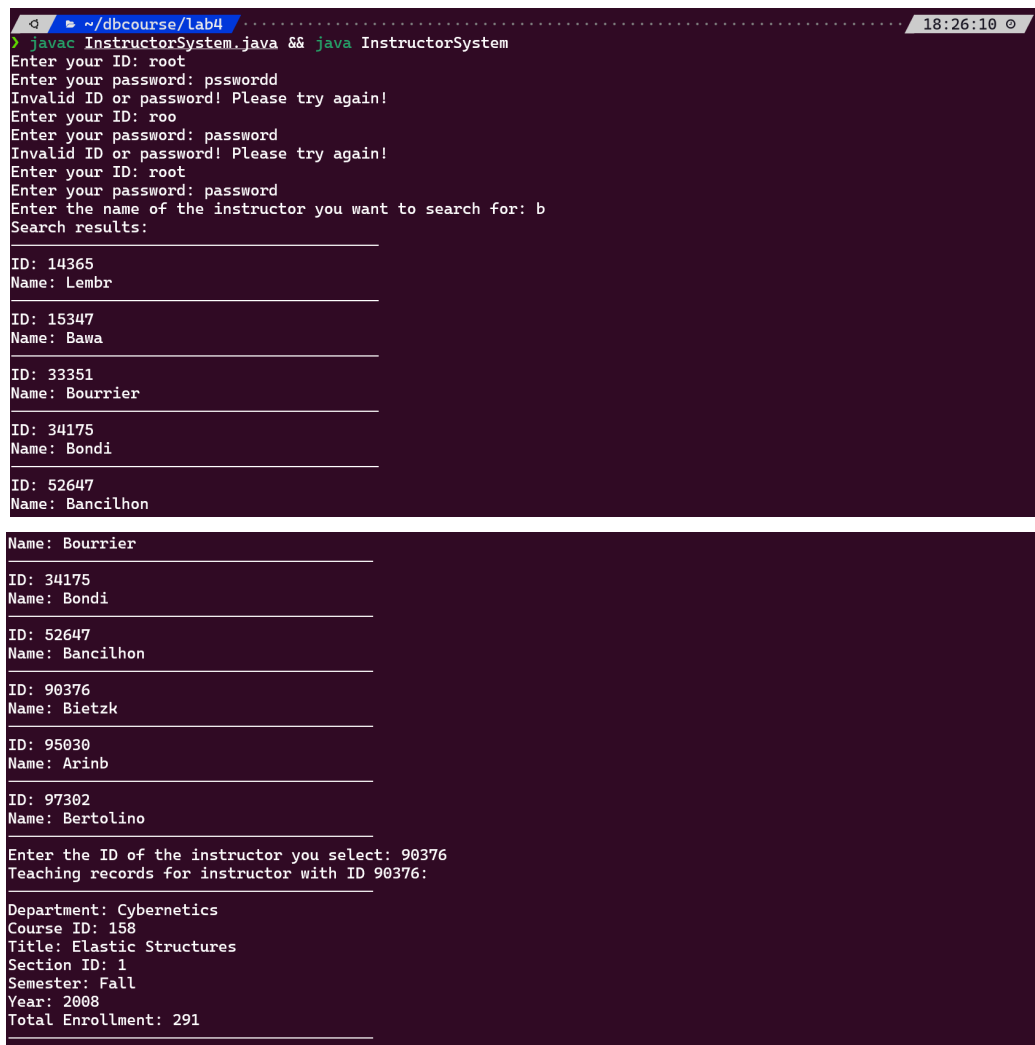
```

```
128     public static void main(String[] args) {
129         login();
130         searchInstructors();
131         int instructorID = checkInstructor();
132         searchTeachingRecords(instructorID);
133     }
134 }
```

使用以下命令来编译和运行该程序：

```
1 javac InstructorSystem.java && java InstructorSystem
```

运行结果如下图所示：



```
> javac InstructorSystem.java && java InstructorSystem
Enter your ID: root
Enter your password: psswordd
Invalid ID or password! Please try again!
Enter your ID: roo
Enter your password: password
Invalid ID or password! Please try again!
Enter your ID: root
Enter your password: password
Enter the name of the instructor you want to search for: b
Search results:
ID: 14365
Name: Lembr
ID: 15347
Name: Bawa
ID: 33351
Name: Bourrier
ID: 34175
Name: Bondi
ID: 52647
Name: Bancelhon
Name: Bourrier
ID: 34175
Name: Bondi
ID: 52647
Name: Bancelhon
ID: 90376
Name: Bietzk
ID: 95030
Name: Arinb
ID: 97302
Name: Bertolino
Enter the ID of the instructor you select: 90376
Teaching records for instructor with ID 90376:
Department: Cybernetics
Course ID: 158
Title: Elastic Structures
Section ID: 1
Semester: Fall
Year: 2008
Total Enrollment: 291
```

图 12: Teaching Record Management System

## 2.4 MetaDisplay

Suppose you were asked to define a class **MetaDisplay** in Java, containing a method

```
static void printTable(String r);
```

the method takes a relation name **r** as input, executes the query **select \* from r**, and prints the result out in tabular format, with the attribute names displayed in the header of the table.

- (a) What do you need to know about relation **r** to be able to print the result in the specified tabular format?

我们需要知道关系 **r** 的属性名和属性类型，以便能够正确地打印出表格。

- (b) What JDBC methods(s) can get you the required information?

我们可以使用 **ResultSetMetaData** 类的 **getColumnCount()** 和 **getColumnName()** 方法来获取关系 **r** 的属性名。

- (c) Write the method **printTable(String r)** using the JDBC API.

所写的 Java 程序如下：

```
1  import java.sql.Connection;
2  import java.sql.DriverManager;
3  import java.sql.ResultSet;
4  import java.sql.ResultSetMetaData;
5  import java.sql.SQLException;
6  import java.sql.Statement;
7
8  public class MetaDisplay {
9      public static void printTable(String r) {
10         String url = "jdbc:mysql://localhost:53306/dbcourse?useSSL=false&
            allowPublicKeyRetrieval=true";
11         String username = "root";
12         String password = "password";
13
14         String query = "SELECT * FROM " + r;
15
16         try (Connection connection = DriverManager.getConnection(url, username,
            password);
17             Statement statement = connection.createStatement(ResultSet.
                TYPE_SCROLL_INSENSITIVE,
18                 ResultSet.CONCUR_READ_ONLY);
19             ResultSet resultSet = statement.executeQuery(query)) {
20
21             ResultSetMetaData metaData = resultSet.getMetaData();
22             int columnCount = metaData.getColumnCount();
```

```
23
24     int[] columnWidths = new int[columnCount + 1];
25
26     while (resultSet.next()) {
27         for (int i = 1; i <= columnCount; i++) {
28             String columnValue = resultSet.getString(i);
29             if (columnValue == null) {
30                 columnValue = "NULL";
31             }
32             columnWidths[i] = Math.max(columnValue.length(), columnWidths[i]);
33         }
34     }
35
36     resultSet.beforeFirst();
37
38     for (int i = 1; i <= columnCount; i++) {
39         System.out.print("+" + "-".repeat(columnWidths[i] + 2));
40     }
41     System.out.println("+");
42
43     for (int i = 1; i <= columnCount; i++) {
44         String columnName = metaData.getColumnName(i);
45         System.out.printf("| %" + columnWidths[i] + "s ", columnName);
46     }
47     System.out.println("|");
48
49     for (int i = 1; i <= columnCount; i++) {
50         System.out.print("+" + "-".repeat(columnWidths[i] + 2));
51     }
52     System.out.println("+");
53
54     while (resultSet.next()) {
55         for (int i = 1; i <= columnCount; i++) {
56             String columnValue = resultSet.getString(i);
57             if (columnValue == null) {
58                 columnValue = "NULL";
59             }
60             System.out.printf("| %" + columnWidths[i] + "s ", columnValue);
61         }
62         System.out.println("|");
63     }
64     for (int i = 1; i <= columnCount; i++) {
65         System.out.print("+" + "-".repeat(columnWidths[i] + 2));
```

```

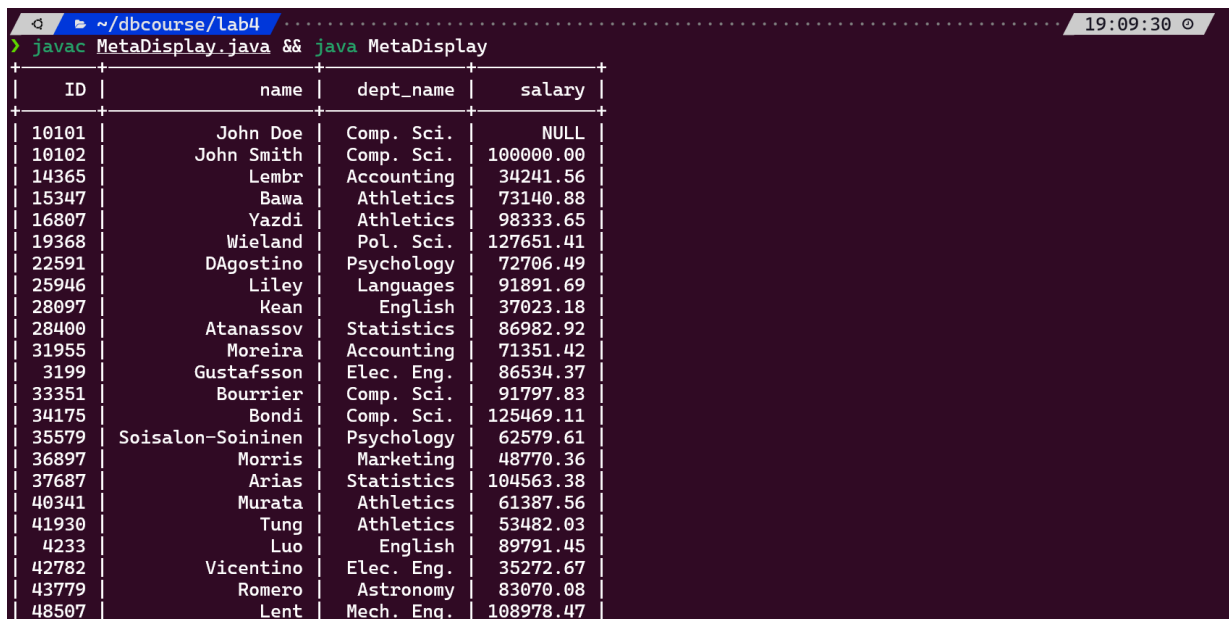
66         }
67         System.out.println("+");
68
69     } catch (SQLException e) {
70         e.printStackTrace();
71     }
72 }
73
74 public static void main(String[] args) {
75     printTable("instructor");
76 }
77 }

```

使用以下命令来编译和运行该程序：

```
1 javac MetaDisplay.java && java MetaDisplay
```

运行结果如下图所示：



ID	name	dept_name	salary
10101	John Doe	Comp. Sci.	NULL
10102	John Smith	Comp. Sci.	100000.00
14365	Lembr	Accounting	34241.56
15347	Bawa	Athletics	73140.88
16807	Yazdi	Athletics	98333.65
19368	Wieland	Pol. Sci.	127651.41
22591	DAgostino	Psychology	72706.49
25946	Liley	Languages	91891.69
28097	Kean	English	37023.18
28400	Atanassov	Statistics	86982.92
31955	Moreira	Accounting	71351.42
3199	Gustafsson	Elec. Eng.	86534.37
33351	Bourrier	Comp. Sci.	91797.83
34175	Bondi	Comp. Sci.	125469.11
35579	Soisalon-Soininen	Psychology	62579.61
36897	Morris	Marketing	48770.36
37687	Arias	Statistics	104563.38
40341	Murata	Athletics	61387.56
41930	Tung	Athletics	53482.03
4233	Luo	English	89791.45
42782	Vicentino	Elec. Eng.	35272.67
43779	Romero	Astronomy	83070.08
48507	Lent	Mech. Eng.	108978.47

图 13: MetaDisplay

### 3 存在的问题及解决方案

1. 在尝试与数据库建立连接时，出现“找不到 xx 类”的错误，这是由于环境变量 CLASSPATH 设置错误，在设置 CLASSPATH 时，应该将 mysql-connector-java-8.4.0.jar 的详细路径添加到 CLASSPATH 中，需

要精确到.jar 文件的路径,而不是目录,或者也可以使用 /\* 来通配目录下的所有.jar 文件。另外,还需要注意,CLASSPATH 中还应该包含当前目录,即.,否则也会出现找不到类的错误。

2. 运行时出现报错 `java.sql.SQLNonTransientConnectionException: Public Key Retrieval is not allowed`,这是由于新版 MySQL 默认不允许 `PublicKeyRetrieval`,需要在连接 URL 中添加 `allowPublicKeyRetrieval=true` 参数。
3. 在判断 `ResultSet` 是否为空时,使用了 `if(!rs.next())`,导致第一行数据丢失,可以考虑改用 `if(!rs.isBeforeFirst())`。

## 4 实验小结

通过本次实验,我学会了通过 JDBC 接口访问关系数据库的基本方法,掌握了 JDBC 的基本操作,包括与数据库建立连接、在数据库中创建数据表、向数据表中插入数据、从数据表中查询数据、更新数据表中的数据、删除数据表中的数据、使用 `prepare` 语句向数据表中批量插入(更新)数据、以及 JDBC 应用,包括 `Teaching Record Management System` 和 `MetaDisplay` 两个例子。通过本次实验,我对 JDBC 有了更深入的了解,掌握了 JDBC 的基本操作,为以后的数据库应用开发打下了基础。