哈尔滨工业大学

<<数据库系统>>

实验报告一

(2023 年度秋季学期)

姓名：	陈俊乐
学号：	2021113694
学院：	未来技术学院
教师：	李东博

实验一 数据库系统开发

一、实验目的

在熟练掌握 MySQL 基本命令、SQL 语言以及用C、C++、Java、Python或Php等语言编写MySQL操作程序的基础上，学习简单数据库系统的设计方法，包括数据库概要设计、逻辑设计

二、实验环境

Window10，MySQL Server 8.0，Intellij idea, jdk20

三、实验过程及结果

3.1 系统需求：

1. 实体

- 学生：学号，姓名，课程号，学院，专业
- 班级：班级号，班长
- 学院：学院名，学院 ID
- 学生证：学号，发放日期，过期日期
- 教室：教室 ID，容量
- 课程：课程号，老师 ID，教室 ID，课程名
- 专业：学生数量，专业名
- 老师：老师名，老师 ID

2.联系

一对一

- 一个学生只能有一个学生证，一个学生证也只能属于一个学生

一对多

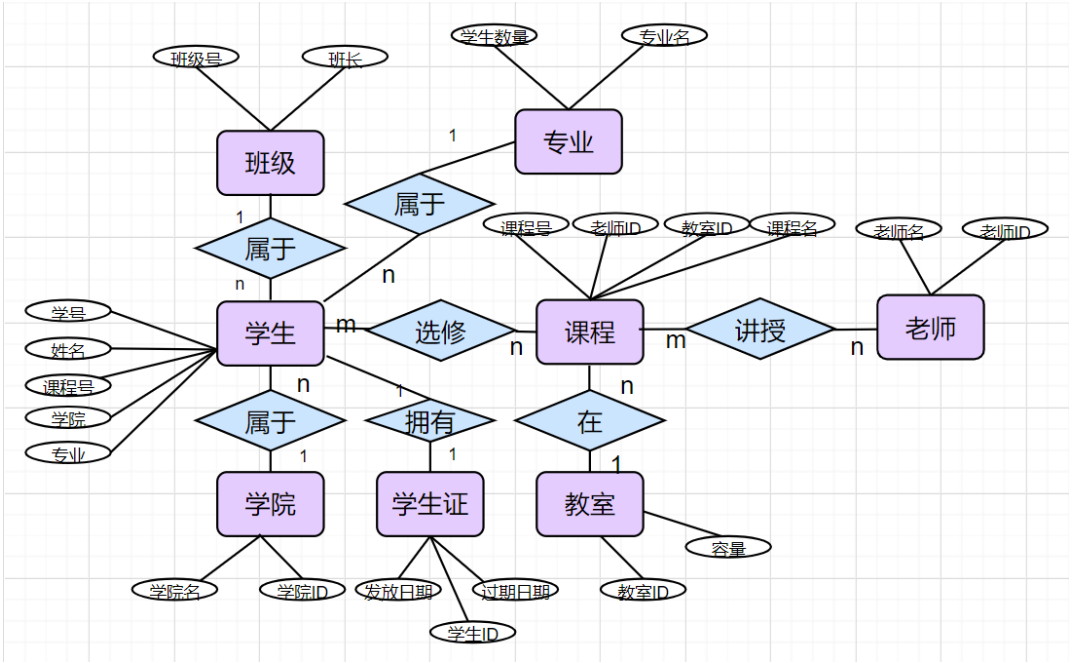
- 一个班级可以有多个学生，但是一个学生只能属于一个班级
- 一个学院可以有多个学生，但是一个学生只能属于一个学院
- 一个专业可以有多个学生，但是一个学生只能属于一个专业
- 一个教室可以上多门课程，但是一门课程在一个教室上课

多对多

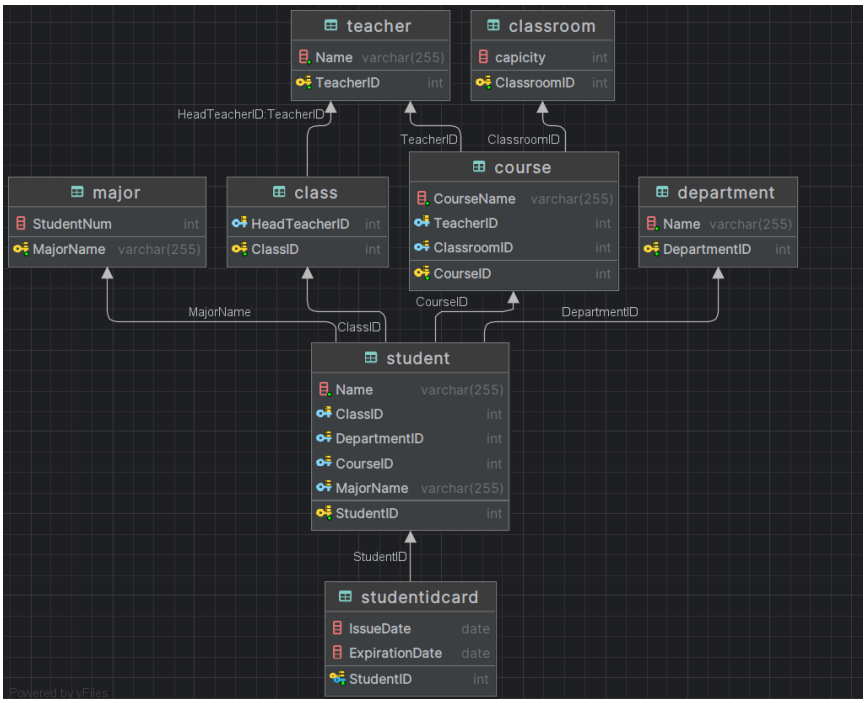
- 一个学生能选修多门课程，一门课程也可以被多名学生选修
- 一门课程可以被多位老师讲授，一个老师也可以讲多门课程

3.2 E-R 图：

1.个人绘制 ER 图



2．软件生成 ER 图



3.3 关系表：

学生（学号，姓名，课程号，学院，专业，班级号）

班级（班级号，班长）

学院（学院名，学院 ID）

学生证（学号，发放日期，过期日期）

教室（教室 ID，容量）

课程（课程号，老师 ID，教室 ID，课程名）

专业（学生数量，专业名）

老师（老师名，老师 ID）

选修关系（学生 ID，课程 ID）

讲授关系（课程 ID，老师 ID）

加下划线表示主键

3.3 关系的完整性约束：

1.实体完整性:

- 学生的学号是学生实体的主键，必须是唯一的。
- 班级的班级号是班级实体的主键，必须是唯一的。
- 学院的学院 ID 是学院实体的主键，必须是唯一的。
- 学生证的学号与学生的学号相关联，并且是学生证实体的主键。
- 教室的教室 ID 是教室实体的主键，必须是唯一的。
- 课程的课程号是课程实体的主键，必须是唯一的。
- 专业的专业名是专业实体的主键，必须是唯一的。
- 老师的老师 ID 是老师实体的主键，必须是唯一的。

2.参照完整性:

- 学生证中的学号必须引用一个存在的学生实体的学号。
- 课程中的教室 ID 必须引用一个存在的教室实体的教室 ID。
- 课程中的老师 ID 必须引用一个存在的老师实体的老师 ID。
- 学生实体中的班级号、学院和专业必须分别引用存在的班级、学院和专业实体。

3.用户定义的完整性:

- 学生和学生证之间的关系是一对一的，这意味着每个学生证号必须唯一对应一个学生，反之亦然。
- 班级和学生之间的关系是一对多的，每个学生只能属于一个班级，但一个班级可以有多个学生。
- 学院和学生之间的关系是一对多的，每个学生只能属于一个学院，但一个学院可以有多个学生。
- 专业和学生之间的关系是一对多的，每个学生只能属于一个专业，但一个专业可以有多个学生。
- 教室和课程之间的关系是一对多的，一门课程只能在一个教室上课，但一个教室可以用于多门课程。
- 学生和课程之间的关系是多对多的，一个学生可以选修多门课程，一门课程可以由多个学生选修。
- 老师和课程之间的关系是多对多的，一个老师可以讲授多门课程，一门课程也可以由多位老师讲授。

3.4 创建的视图和索引

我创建了两个视图：

- 学生视图，它包含了 StudentID 大于 1 的所有学生记录，索引是学号
- 课程视图，它包含了 CourseID 大于 1 的所有课程记录，索引是课程号

我使用 createView 方法用于根据给定的查询创建一个新的视图。这个方法采用两个参数：视图的名称（viewName）和用于创建视图的查询（query）。方法内部，首先使用数据库连接来创建一个 Statement 对象，然后使用该对象来执行一个包含 CREATE VIEW 语句的查询，从而在数据库中创建视图。

在这个过程中，还使用了正则表达式来查找 CREATE VIEW 查询中的 WHERE 子句，如果找到了，就将条件打印出来。

```
public void createView(String viewName, String query) {
    try {
        Statement statement = connection.createStatement();
        String createViewQuery = "CREATE VIEW " + viewName + " AS " + query;
        statement.executeUpdate(createViewQuery);
        Pattern pattern = Pattern.compile( regex: "WHERE\\s+(.*)");
        Matcher matcher = pattern.matcher(createViewQuery);
        String condition = null;
        if (matcher.find()) {
            condition = matcher.group(1);
        }
        System.out.println("View " + viewName + " created successfully.");
        System.out.println(createViewQuery);
        printTableColumnWithCondition(viewName, columnName: "*", condition);

    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

调用 createView 方法的部分包含了两个 TODOs，分别是创建学生视图和课程视图，使用了 SELECT * FROM Student WHERE StudentID > 1 和 SELECT * FROM Course WHERE CourseID > 1 这两个查询。这意味着您想要创建两个视图，一个是学生视图，它包含了 StudentID 大于 1 的所有学生记录；另一个是课程视图，它包含了 CourseID 大于 1 的所有课程记录。

```
//TODO:创建学生视图
dbOperations.createView( viewName: "StudentView", query: "SELECT * FROM Student WHERE StudentID > 1");
//TODO:创建课程视图
dbOperations.createView( viewName: "CourseView", query: "SELECT * FROM Course WHERE CourseID > 1");
```

3.5 最关键代码

1.数据库连接（URL,USERNAME,PASSWORD）

url 为"jdbc:mysql://localhost:3306"，USERNAME 和 PASSWORD 对应用户名和密码

```
    public static Connection getConnection() {
        try {
            return DriverManager.getConnection(URL, USERNAME, PASSWORD);
        } catch (SQLException e) {
            e.printStackTrace();
            throw new RuntimeException("Failed to connect to the database.");
        }
    }
}
```

2.数据库创建

1. 创建数据库：

- 首先检查使用 databaseExists 方法（未在代码中显示）来确定要创建的数据库是否已存在。
- 如果不存在，它将执行一个 CREATE DATABASE SQL 语句来创建一个新的数据库。创建数据库时，指定了字符集 utf8mb4 和校对规则 utf8mb4_unicode_ci，这意味着数据库支持最新的 Unicode 字符集，这对于存储表情符号等字符非常有用。
- 创建数据库成功后，控制台将打印出成功消息。

2. 切换到新数据库：

- 使用 connection.setCatalog(databaseName)切换到新创建的数据库。

3. 创建表和插入数据：

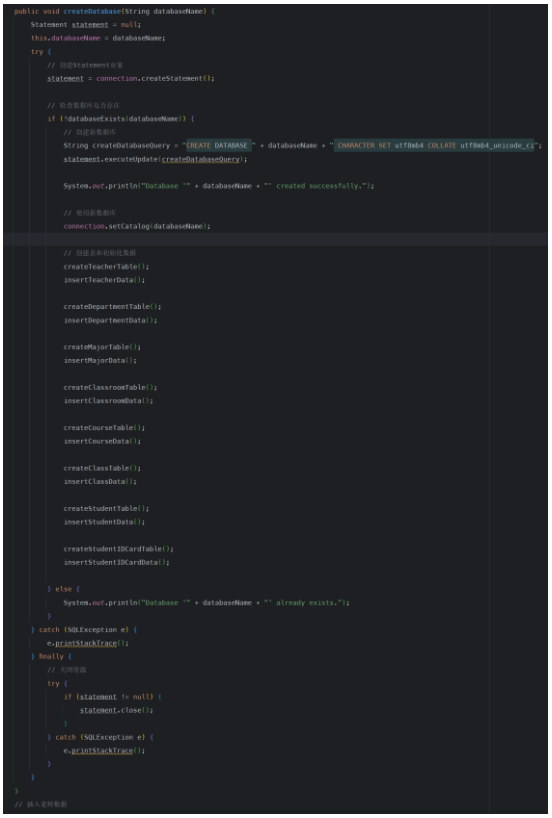
- 通过一系列的 createXTable()和 insertXData()方法调用来创建表和插入数据。这些方法的名字暗示了它们分别用于创建不同的表和向这些表中插入数据。表可能包括老师、系、专业、教室、课程、班级、学生和学生证等。

4. 异常处理：

- 如果在执行过程中遇到任何 SQLException，异常的信息将通过 printStackTrace()方法打印出来，以便调试。

5. 资源清理：

- 在 finally 块中，代码确保 Statement 对象被关闭，这是避免潜在内存泄漏的良好实践。



```
public void createDatabase(String databaseName) {
    Statement statement = null;
    this.databaseName = databaseName;
    try {
        // 创建Statement对象
        statement = connection.createStatement();

        // 检查数据库是否存在
        if (!databaseExists(databaseName)) {
            // 创建新数据库
            String createDatabaseQuery = "CREATE DATABASE " + databaseName + " CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci";
            statement.executeUpdate(createDatabaseQuery);

            System.out.println("Database '" + databaseName + "' created successfully.");

            // 使用新数据库
            connection.setCatalog(databaseName);

            // 创建表和初始化数据
            createTeacherTable();
            insertTeacherData();

            createDepartmentTable();
            insertDepartmentData();

            createMajorTable();
            insertMajorData();

            createClassroomTable();
            insertClassroomData();

            createCourseTable();
            insertCourseData();

            createClassTable();
            insertClassData();

            createStudentTable();
            insertStudentData();

            createStudentIDCardTable();
            insertStudentIDCardData();

        } else {
            System.out.println("Database '" + databaseName + "' already exists.");
        }
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        // 关闭资源
        try {
            if (statement != null) {
                statement.close();
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
// 插入老师数据
```

3.数据的插入

1. **选择数据库：**
   - 使用 **USE** SQL 命令选择特定的数据库，**databaseName** 变量包含了数据库名称。
2. **构造插入语句：**
   - **columnsStr** 是由 **columns** 数组中的字段名拼接成的字符串，用于指定 **INSERT INTO** 语句中的列名部分。
   - **valuesStr** 是一个由问号**?**组成的字符串，其数量与 **columns** 数组的长度相同，每个问号代表一个值将要被插入。这部分是预备语句（PreparedStatement）的参数占位符。
3. **准备和执行插入操作：**
   - 一个 **INSERT INTO** 语句被创建，指定了表名 **tableName**、列名 **columnsStr** 和对应的值 **valuesStr**。
   - 使用 **connection.prepareStatement** 创建一个 **PreparedStatement** 对象，这有助于防止 SQL 注入，并且可以提高性能。
   - **for** 循环遍历 **values** 数组，通过 **setString** 方法为每个占位符设置相应的值。这里假设所有的值都是字符串类型。
4. **执行更新：**
   - 调用 **executeUpdate** 方法执行 **PreparedStatement**，这通常用于执行 **INSERT**、**UPDATE** 或 **DELETE** 语句。
   - 插入成功后，控制台会输出“Data inserted successfully.”的信息。
5. **异常处理：**
   - 如果在执行过程中遇到 **SQLException**，会检查错误信息。
   - 如果错误信息包含“Duplicate entry”，则打印出“Error: Duplicate entry. Please check your data.”，提示数据违反了唯一性约束。
   - 如果错误信息包含“cannot be null”，则打印出“Error: Null value is not allowed. Please check your data.”，提示数据违反了非空约束。
   - 如果是其他类型的 **SQLException**，则调用 **printStackTrace** 方法来打印异常信息，便于调试。

```
try {

    String selectDatabaseQuery = "USE " + databaseName;
    Statement statement = connection.createStatement();
    statement.executeUpdate(selectDatabaseQuery);
    // 构建插入语句
    String columnsStr = String.join( delimiter: ", ", columns);
    String valuesStr = String.join( delimiter: ", ", Collections.nCopies(values.length, o: "?")); // 用?替代值

    String query = "INSERT INTO " + tableName + " (" + columnsStr + ") VALUES (" + valuesStr + ")";
    PreparedStatement preparedStatement = connection.prepareStatement(query);

    // 设置参数值
    for (int i = 0; i < values.length; i++) {
        preparedStatement.setString( parameterIndex: i + 1, values[i]);
    }

    preparedStatement.executeUpdate();
    System.out.println("Data inserted successfully.");



} catch (SQLException e) {
    // 处理插入异常，例如空值或重复值
    if (e.getMessage().contains("Duplicate entry")) {
        System.out.println("Error: Duplicate entry. Please check your data.");
    } else if (e.getMessage().contains("cannot be null")) {
        System.out.println("Error: Null value is not allowed. Please check your data.");
    } else {
        e.printStackTrace();
    }
}
}
```

4.数据的删除

1. **构建删除语句：**
   - 构造一个 **DELETE** SQL 语句，其中包含表名、条件列和一个参数占位符（**?**）。
   - 参数占位符用于预防 SQL 注入攻击，并且可以让你在执行语句之前设置参数值。
2. **设置参数并执行删除：**
   - 使用 **PreparedStatement** 来执行删除操作。与普通的 **Statement** 相比，**PreparedStatement** 可以防止 SQL 注入攻击，并且通常性能更好。
   - 设置占位符的值为 **conditionValue**，这是调用方法时传入的删除条件值。
   - 调用 **executeUpdate** 来执行删除操作，并返回一个整数值 **rowsAffected**，表示受影响的行数。
3. **处理删除结果：**
   - 如果 **rowsAffected** 大于 0，表示有数据被删除，控制台将输出“Data deleted successfully.”。
   - 如果 **rowsAffected** 为 0，表示没有找到匹配的数据进行删除，控制台将输出“Error: Data not found for deletion.”。

```
public void deleteData(String tableName, String conditionColumn, String conditionValue) {
    try {
        String selectDatabaseQuery = "USE "+ databaseName;
        Statement statement = connection.createStatement();
        statement.executeUpdate(selectDatabaseQuery);

        // 构建删除语句
        String query = "DELETE FROM " + tableName + " WHERE " + conditionColumn + " = ?";

        // 执行删除
        PreparedStatement preparedStatement = connection.prepareStatement(query);
        preparedStatement.setString( parameterIndex: 1, conditionValue);
        int rowsAffected = preparedStatement.executeUpdate();

        // 处理删除结果
        if (rowsAffected > 0) {
            System.out.println("Data deleted successfully.");
        } else {
            System.out.println("Error: Data not found for deletion.");
        }
    } catch (SQLException e) {
        e.printStackTrace();
```

4.数据库的连接查询

1. **根据模式选择联接类型：**
   - 如果 mode 等于 0，执行自然联接（Natural Join），该联接基于两个表中有相同名称的所有列。
   - 如果 mode 等于 1，执行左外联接（Left Outer Join），该联接返回左表（firstTable）的所有记录以及右表（secondTable）中联接字段匹配的记录。
   - 如果 mode 等于 2，执行右外联接（Right Outer Join），该联接返回右表的所有记录以及左表中联接字段匹配的记录。
2. **构建查询并执行：**
   - 根据所选的模式，构建一个 CREATE TABLE 语句，该语句使用对应的联接类型将结果集创建为一个新表。
   - 使用 connection.prepareStatement 创建一个 PreparedStatement 对象，这有助于

防止 SQL 注入。

- 执行 executeUpdate 方法运行这个 CREATE TABLE 语句，创建一个新的表，并填充联接结果的数据。

```
public void performDynamicJoinQuery(String firstTable, String secondTable, String newTableName,int mode) {
    try {
        String query = null;
        // 自然连接
        if (mode == 0)
            query = "CREATE TABLE "+ newTableName +" AS SELECT * FROM " + firstTable + " NATURAL JOIN " + secondTable;
        // 左外连接
        if (mode == 1)
            query = "CREATE TABLE "+ newTableName +" AS SELECT Student.StudentID, Student.Name, Student.ClassID, " +
                    "Student.DepartmentID, Student.CourseID, Student.MajorName, Course.CourseName, Course.TeacherID, " +
                    "Course.ClassroomID FROM " + firstTable +
                    " LEFT JOIN " + secondTable + " ON " + firstTable + ".CourseID = " + secondTable + ".CourseID";
        //右外连接
        if (mode == 2)
            query = "CREATE TABLE "+ newTableName +" AS SELECT Student.StudentID, Student.Name, Student.ClassID, " +
                    "Student.DepartmentID, Student.CourseID, Student.MajorName, Course.CourseName, Course.TeacherID, " +
                    "Course.ClassroomID FROM " + firstTable +
                    " RIGHT JOIN " + secondTable + " ON " + firstTable + ".CourseID = " + secondTable + ".CourseID";

        PreparedStatement preparedStatement = connection.prepareStatement(query);
        preparedStatement.executeUpdate();
```

5.数据库的嵌套查询

从表中选择所有学生的信息，但是只选择那些选修了"Calculus I"课程的学生

```
// 嵌套查询
1 usage
public void performNestedQuery() {
    // 构建嵌套查询语句
    String query = "SELECT * FROM Student WHERE CourseID IN (SELECT CourseID FROM Course WHERE CourseName = 'Calculus I')";

    // 执行查询并打印结果
    printTableColumnWithCondition( tableName: "SC",  columnName: "*",  condition: "CourseID IN (SELECT CourseID FROM" +
            " Course WHERE CourseName = 'Calculus I')");
}
```

6.数据库的分组查询

统计每门课程的学生人数，并且只返回学生人数大于等于 1 的课程

```
public void performGroupByQueryWithHaving() throws SQLException {

    // 构建带有HAVING子句的分组查询语句
    String query = "SELECT CourseID, COUNT(*) FROM SC GROUP BY CourseID HAVING COUNT(*) > 1";
    String selectDatabaseQuery = "USE " + databaseName;
    Statement selectStatement = connection.createStatement();
    selectStatement.executeUpdate(selectDatabaseQuery);

    Statement statement = connection.createStatement();
    System.out.println(query);
    ResultSet resultSet = statement.executeQuery(query);
```

3.6 系统功能运行界面

1.创建数据库和初始化数据库的数据，包含创建数据库成功的提示信息，还有初始化数据插入成功信息

```
Database 'School' dropped successfully.
Database 'School' created successfully.
Teacher table created successfully.
Data inserted successfully.
Data inserted successfully.
Data inserted successfully.
Data inserted successfully.
Department table created successfully.
Data inserted successfully.
Data inserted successfully.
Data inserted successfully.
Data inserted successfully.
Major table created successfully.
Data inserted successfully.
Data inserted successfully.
Data inserted successfully.
Data inserted successfully.
Data inserted successfully.
```

2.选择打印学生表和专业表

```
| StudentID      | Name           | ClassID        | DepartmentID   | CourseID       | MajorName      |
+----------------+----------------+----------------+----------------+----------------+----------------+
| 1              | Alice Johnson  | 2136005        | 1              | 1              | Computer Science|
| 2              | Bob Williams   | 2136005        | 2              | 2              | Mathematics    |
| 3              | Charlie Smith  | 2136005        | 2              | 2              | AI             |
| 4              | David Jones    | 2136006        | 4              | 4              | Chemistry      |
| 5              | Emma Brown     | 2136006        | 3              | 3              | AI             |
| 6              | Grace Davis    | 2136006        | 3              | 3              | AI             |
| 7              | Helen Miller   | 2136006        | 3              | 3              | Computer Science|
| 8              | Isabella Wilson| 2136006        | 3              | 3              | Data Science   |
| 9              | Jack Taylor    | 2136006        | 3              | 3              | Data Science   |
| MajorName      | StudentNum     |
+----------------+----------------+
| AI             | 60             |
| Chemistry      | 100            |
| Computer Science| 100            |
| Data Science   | 80             |
| Mathematics    | 80             |
```

3.插入 ID 为 10 的学生信息

```
| StudentID      | Name           | ClassID        | DepartmentID   | CourseID       | MajorName      |
+----------------+----------------+----------------+----------------+----------------+----------------+
| 1              | Alice Johnson  | 2136005        | 1              | 1              | Computer Science|
| 2              | Bob Williams   | 2136005        | 2              | 2              | Mathematics    |
| 3              | Charlie Smith  | 2136005        | 2              | 2              | AI             |
| 4              | David Jones    | 2136006        | 4              | 4              | Chemistry      |
| 5              | Emma Brown     | 2136006        | 3              | 3              | AI             |
| 6              | Grace Davis    | 2136006        | 3              | 3              | AI             |
| 7              | Helen Miller   | 2136006        | 3              | 3              | Computer Science|
| 8              | Isabella Wilson| 2136006        | 3              | 3              | Data Science   |
| 9              | Jack Taylor    | 2136006        | 3              | 3              | Data Science   |
| 10             | Jack Hug       | 2136005        | 3              | 3              | Data Science   |
```

4.插入空值，重复值的提醒

比如插入 ID 等于 10 或者插入信息中出现 NULL 就会进行提醒

```
Error: Duplicate entry. Please check your data.
Error: Null value is not allowed. Please check your data.
Transaction rolled back.
Error: Duplicate entry. Please check your data.
Transaction committed successfully.
Transaction rolled back.
Error: Duplicate entry. Please check your data.
Transaction committed successfully.
Transaction rolled back.
```

5.当删除不存在的值的时候会打印 Error: Data not found for deletion.，下图是删除不存在值打印的 error 和删除 ID=10 以后的学生表，可以看到 ID 为 10 的学生信息被删除，同时打印 delete student with id 10

```
Error: Data not found for deletion.
Data deleted successfully.
delete student with id 10
| StudentID      | Name           | ClassID        | DepartmentID   | CourseID       | MajorName      |
+----------------+----------------+----------------+----------------+----------------+----------------+
| 1              | Alice Johnson  | 2136005        | 1              | 1              | Computer Science|
| 2              | Bob Williams   | 2136005        | 2              | 2              | Mathematics    |
| 3              | Charlie Smith  | 2136005        | 2              | 2              | AI             |
| 4              | David Jones    | 2136006        | 4              | 4              | Chemistry      |
| 5              | Emma Brown     | 2136006        | 3              | 3              | AI             |
| 6              | Grace Davis    | 2136006        | 3              | 3              | AI             |
| 7              | Helen Miller   | 2136006        | 3              | 3              | Computer Science|
| 8              | Isabella Wilson| 2136006        | 3              | 3              | Data Science   |
| 9              | Jack Taylor    | 2136006        | 3              | 3              | Data Science   |
```

6.自然连接演示，将课程表和学生表进行自然连接

连接前的课程表和学生表

```
| StudentID      | Name           | ClassID        | DepartmentID   | CourseID       | MajorName      |
+----------------+----------------+----------------+----------------+----------------+----------------+
| 1              | Alice Johnson  | 2136005        | 1              | 1              | Computer Science|
| 2              | Bob Williams   | 2136005        | 2              | 2              | Mathematics    |
| 3              | Charlie Smith  | 2136005        | 2              | 2              | AI             |
| 4              | David Jones    | 2136006        | 4              | 4              | Chemistry      |
| 5              | Emma Brown     | 2136006        | 3              | 3              | AI             |
| 6              | Grace Davis    | 2136006        | 3              | 3              | AI             |
| 7              | Helen Miller   | 2136006        | 3              | 3              | Computer Science|
| 8              | Isabella Wilson| 2136006        | 3              | 3              | Data Science   |
| 9              | Jack Taylor    | 2136006        | 3              | 3              | Data Science   |
| CourseID       | CourseName     | TeacherID      | ClassroomID    |
+----------------+----------------+----------------+----------------+
| 1              | Programming    | 1              | 101            |
| 2              | Calculus I     | 2              | 102            |
| 3              | Calculus II    | 3              | 103            |
| 4              | Linear Algebra | 4              | 104            |
```

连接后的 SC 表，通过 CourseID 连接

```
| CourseID      | StudentID     | Name          | ClassID       | DepartmentID  | MajorName     | CourseName    | TeacherID     | ClassroomID   |
+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+
| 1             | 1             | Alice Johnson | 2136005       | 1             | Computer Science| Programming   | 1             | 101           |
| 2             | 2             | Bob Williams  | 2136005       | 2             | Mathematics   | Calculus I    | 2             | 102           |
| 2             | 3             | Charlie Smith | 2136005       | 2             | AI            | Calculus I    | 2             | 102           |
| 4             | 4             | David Jones   | 2136006       | 4             | Chemistry     | Linear Algebra| 4             | 104           |
| 3             | 5             | Emma Brown    | 2136006       | 3             | AI            | Calculus II   | 3             | 103           |
| 3             | 6             | Grace Davis   | 2136006       | 3             | AI            | Calculus II   | 3             | 103           |
| 3             | 7             | Helen Miller  | 2136006       | 3             | Computer Science| Calculus II   | 3             | 103           |
| 3             | 8             | Isabella Wilson| 2136006      | 3             | Data Science  | Calculus II   | 3             | 103           |
| 3             | 9             | Jack Taylor   | 2136006       | 3             | Data Science  | Calculus II   | 3             | 103           |
```

连接查询，在连接后的 SC 表中查询 CourseID=2 的信息

```
start query CourseID=2 in SC
| CourseID      | StudentID     | Name          | ClassID       | DepartmentID  | MajorName     | CourseName    | TeacherID     | ClassroomID   |
+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+
| 2             | 2             | Bob Williams  | 2136005       | 2             | Mathematics   | Calculus I    | 2             | 102           |
| 2             | 3             | Charlie Smith | 2136005       | 2             | AI            | Calculus I    | 2             | 102           |
```

7. 嵌套查询，从 SC 表中选择所有学生的信息，但是只选择那些选修了"Calculus I"课程的学生

```
start query CourseID IN (SELECT CourseID FROM Course WHERE CourseName = 'Calculus I') in SC
| CourseID      | StudentID     | Name          | ClassID       | DepartmentID  | MajorName     | CourseName    | TeacherID     | ClassroomID   |
+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+
| 2             | 2             | Bob Williams  | 2136005       | 2             | Mathematics   | Calculus I    | 2             | 102           |
| 2             | 3             | Charlie Smith | 2136005       | 2             | AI            | Calculus I    | 2             | 102           |
```

8. 分组查询查询，从 SC 表中统计每门课程的学生人数，并且只返回学生人数大于等于 1 的课程

```
SELECT CourseID, COUNT(*) FROM SC GROUP BY CourseID HAVING COUNT(*) > 1
| CourseID      | COUNT(*)      |
+---------------+---------------+
| 2             | 2             |
| 3             | 5             |
```

9.创建学号大于 1 的学生视图

```
CREATE VIEW StudentView AS SELECT * FROM Student WHERE StudentID > 1
start query StudentID > 1 in StudentView
| StudentID     | Name          | ClassID       | DepartmentID  | CourseID      | MajorName     |
+---------------+---------------+---------------+---------------+---------------+---------------+
| 2             | Bob Williams  | 2136005       | 2             | 2             | Mathematics   |
| 3             | Charlie Smith | 2136005       | 2             | 2             | AI            |
| 4             | David Jones   | 2136006       | 4             | 4             | Chemistry     |
| 5             | Emma Brown    | 2136006       | 3             | 3             | AI            |
| 6             | Grace Davis   | 2136006       | 3             | 3             | AI            |
| 7             | Helen Miller  | 2136006       | 3             | 3             | Computer Science|
| 8             | Isabella Wilson| 2136006      | 3             | 3             | Data Science  |
| 9             | Jack Taylor   | 2136006       | 3             | 3             | Data Science  |
```

3.7 加分项说明

1. 包含事务管理

1. **事务提交：**
    - 代码中调用 connection.commit()来手动提交事务。这意味着如果插入操作成功执行，所有更改都将被永久地保存到数据库。

2. **异常处理与事务回滚：**
   - 如果在执行插入语句过程中发生异常，会捕获 SQLException。
   - 在异常处理中，调用 connection.rollback()来回滚事务，这样就可以撤销从事务开始以来所做的所有更改。
   - 如果回滚成功，将输出“Transaction rolled back.”消息；如果回滚时抛出异常，则打印异常堆栈跟踪。
3. **错误消息处理：**
   - 如果捕获的异常消息包含“Duplicate entry”，表明尝试插入重复的数据到一个唯一索引字段，将输出对应的错误消息。
   - 如果异常消息包含“cannot be null”，表明尝试插入 null 值到一个不允许 null 的字段，将输出对应的错误消息。
4. **确保事务提交：**
   - 在 finally 块中，首先设置自动提交为 true，以确保如果上述代码中没有显示提交或回滚，任何之后的操作都将自动作为独立事务执行。
   - 然后，检查连接是否未设置为自动提交（意味着之前有可能开启了事务），如果是，则调用 commit()来提交事务。
   - 输出“Transaction committed successfully.”消息表示事务已经提交。

```
    // 插入数据
    String selectDatabaseQuery = "USE "+ databaseName;
    Statement statement = connection.createStatement();
    statement.executeUpdate(selectDatabaseQuery);
    // 构建插入语句
    String columnsStr = String.join( delimiter: ", ", columns);
    String valuesStr = String.join( delimiter: ", ", Collections.nCopies(values.length, o: "?")); // 用?替代值

    String query = "INSERT INTO " + tableName + " (" + columnsStr + ") VALUES (" + valuesStr + ")";
    PreparedStatement preparedStatement = connection.prepareStatement(query);

    // 设置参数值
    for (int i = 0; i < values.length; i++) {
        preparedStatement.setString( parameterIndex: i + 1, values[i]);
    }

    preparedStatement.executeUpdate();
    System.out.println("Data inserted successfully.");

    // 如果没有异常，提交事务
    connection.commit();

} catch (SQLException e) {
    try {
        connection.rollback(); // 回滚事务
        System.out.println("Transaction rolled back.");
    } catch (SQLException ex) {
        ex.printStackTrace();
    }

    // 处理异常
    if (e.getMessage().contains("Duplicate entry")) {
        System.out.println("Error: Duplicate entry. Please check your data.");
    } else if (e.getMessage().contains("cannot be null")) {
        System.out.println("Error: Null value is not allowed. Please check your data.");
    } else {
        e.printStackTrace();
    }
} finally {
    try {
        connection.setAutoCommit(true); // 恢复自动提交
        if (!connection.getAutoCommit()) {
            connection.commit(); // 确保自动提交已经开启，以免影响后续操作
        }
        System.out.println("Transaction committed successfully.");
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

2. 包含触发器

在插入一条新的学生数据的时候，会触发触发器，根据学生的专业，为对应专业的人数加一

下面是结果演示和代码

插入学生 ID=10 之前的学生表和专业表

StudentID	Name	ClassID	DepartmentID	CourseID	MajorName
1	Alice Johnson	2136005	1	1	Computer Science
2	Bob Williams	2136005	2	2	Mathematics
3	Charlie Smith	2136005	2	2	AI
4	David Jones	2136006	4	4	Chemistry
5	Emma Brown	2136006	3	3	AI
6	Grace Davis	2136006	3	3	AI
7	Helen Miller	2136006	3	3	Computer Science
8	Isabella Wilson	2136006	3	3	Data Science
9	Jack Taylor	2136006	3	3	Data Science

MajorName	StudentNum
AI	60
Chemistry	100
Computer Science	100
Data Science	80
Mathematics	80

现在我插入学生 ID=10，专业为 Data Science 的信息，现在打印专业表和学生表，可以看到学生表插入一个新的数据，专业表对应的 DataScience 的 StudentNum+1

StudentID	Name	ClassID	DepartmentID	CourseID	MajorName
1	Alice Johnson	2136005	1	1	Computer Science
2	Bob Williams	2136005	2	2	Mathematics
3	Charlie Smith	2136005	2	2	AI
4	David Jones	2136006	4	4	Chemistry
5	Emma Brown	2136006	3	3	AI
6	Grace Davis	2136006	3	3	AI
7	Helen Miller	2136006	3	3	Computer Science
8	Isabella Wilson	2136006	3	3	Data Science
9	Jack Taylor	2136006	3	3	Data Science
10	Jack Hug	2136005	3	3	Data Science

```
Data inserted successfully.
```

MajorName	StudentNum
AI	60
Chemistry	100
Computer Science	100
Data Science	81
Mathematics	80

下面是具体的代码

```java
public void insertStudentData(String[] values) throws SQLException {
    String[] columns = {"StudentID", "Name", "ClassID", "DepartmentID", "CourseID", "MajorName"};
    if (values == null) {
        System.out.println("Error: Null value is not allowed. Please check your data.");
        return;
    }
    insertData( tableName: "Student", columns, values);

    //keypoint:更新专业人数
    String MajorName = values[5];
    String updateMajorQuery = "UPDATE Major SET StudentNum = StudentNum + 1 WHERE MajorName = '" + MajorName + "'";

    String selectDatabaseQuery = "USE " + databaseName;
    Statement statement = connection.createStatement();
    statement.executeUpdate(selectDatabaseQuery);
    statement.executeUpdate(updateMajorQuery);
}
```

3.界面友好

采用表格的形式打印，方便查看信息

```
| CourseID      | StudentID     | Name          | ClassID       | DepartmentID  | MajorName     | CourseName    | TeacherID     | ClassroomID
+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+
| 1             | 1             | Alice Johnson | 2136005       | 1             | Computer Science| Programming   | 1             | 101
| 2             | 2             | Bob Williams  | 2136005       | 2             | Mathematics   | Calculus I    | 2             | 102
| 2             | 3             | Charlie Smith | 2136005       | 2             | AI            | Calculus I    | 2             | 102
| 4             | 4             | David Jones   | 2136006       | 4             | Chemistry     | Linear Algebra| 4             | 104
| 3             | 5             | Emma Brown    | 2136006       | 3             | AI            | Calculus II   | 3             | 103
| 3             | 6             | Grace Davis   | 2136006       | 3             | AI            | Calculus II   | 3             | 103
| 3             | 7             | Helen Miller  | 2136006       | 3             | Computer Science| Calculus II | 3             | 103
| 3             | 8             | Isabella Wilson| 2136006      | 3             | Data Science  | Calculus II   | 3             | 103
| 3             | 9             | Jack Taylor   | 2136006       | 3             | Data Science  | Calculus II   | 3             | 103
```

四、实验心得

通过这次数据库实验，我对关系型数据库的结构设计和操作有了更深入的了解。实验中，我学习了如何创建 E-R 图来指导数据库表的创建，以及如何通过编写 SQL 语句来实现数据的增删改查。特别是，通过 Java 代码与数据库的交互让我明白了程序如何在实际应用中操作数据库。

在实现事务管理时，我遇到了一些挑战，比如在数据插入失败时如何正确回滚事务。这让我更加重视异常处理的重要性，并且让我认识到了编码时必须考虑到程序的健壮性。

我特别感兴趣的是事务的 ACID 属性，它保证了数据库操作的可靠性。此外，通过预处理语句，我学会了如何提高数据库操作的安全性，防止 SQL 注入攻击。

反思这次实验，我认为我在编写 SQL 语句时还可以更加注意其效率，尤其是在涉及到大量数据的联接查询时。在未来，我希望能将这些知识应用到更复杂的数据库系统设计和优化中，也希望能继续提升我的数据库调优技能。

总的来说，在本次数据库实验中，我的理解关于数据库中的关系、实体与联系有了显著的提升。通过对 SQL 查询的深入练习，从最初的数据库连接困境，到精心绘制 E-R 图，再到编写各种复杂的查询语句，我对数据库的认识逐渐加深。实验中的不同测试，如处理空

值和避免重复记录的插入，不仅帮助我完善了数据库的查询机制，也加深了我对数据完整性和数据库安全性的理解。这一系列过程不仅锻炼了我的技术技能，也提高了我的问题解决能力，让我对数据库的内在逻辑有了更深层次的认识。