Java 与数据库应用开发
实践报告

题目 教务系统

院、系 计算机与控制工程学院

专　业 软件工程

班　级 计 233-1

姓　名 李世赟

学　号 202358503115

2024 年 11 月 30 日

目录

1 项目描述

1.1 问题背景

这是一个使用 Java 语言结合 SQL server 数据库编写的学生管理系统，旨在解决学生、老师以及教务处管理员在个人信息、课程、成绩上的问题。学校需要一个系统来方便地管理学生、课程和教师的相关信息，包括信息的添加、删除、更新、查询以及展示等各种操作。

1.2 数据处理

学生信息（Student）：包括学号、姓名、性别、年龄、专业五项数据，可进行添加、删除、更新和查询操作。

课程信息（Course）：包括课程号、课程名、学分、教师工号四项数据，可进行添加、删除、更新和查询操作。

教师信息（Teacher）：包括教师工号、姓名、性别、年龄、职称、手机号六项数据，可进行添加、删除、更新和查询操作。

选课信息（SC）：包括学号、课程号、成绩三项数据，可进行查询、选课操作。

用户信息（Users）：包括账号、密码、学号、工号、电话号码、角色六项数据，用于用户登录验证和注册操作。

1.3 功能

登录功能：提供管理员、学生和教师三种角色的登录界面，用户输入正确的账号和密码后可进入相应的主界面。

注册功能：管理员、学生和教师可在各自的注册界面进行注册，注册时需填写相关信息，系统会检查用户名是否已存在，若存在则会提示注册失败，反之注册成功，数据存入数据库。

管理员端信息添加功能：管理员可添加学生、课程和教师信息。

管理员端信息删除功能：管理员可删除学生、课程和教师信息。

管理员端信息更新功能：管理员可更新学生、课程和教师信息。

管理员端信息查询功能：管理员可根据学号查询学生的详细信息，包括个人信息、选课信息和成绩等。

管理员端信息展示功能：管理员可展示所有学生、课程、选课、用户和教师的信息。

教师端信息查询功能：教师可在课程管理界面查看自己所授课程信息以及自己的个人信息；教师还可查询自己所授课程的学生成绩信息。

教师端成绩修改功能：教师可以修改学习自己课程学生的成绩。

学生端信息查询功能：；学生可查询个人信息、已选课程信息和成绩。

学生端选课功能：学生可在选课界面查看所有课程信息，并可以选择课程进行选课操作。

1.4 性能要求

系统应具有良好的响应速度,能够快速处理各种操作请求,保证数据的准确性和完整性。

1.5 约束条件

学生表（Student）中，学号、姓名、性别、年龄、专业均非空，学号为主键，性别只能为“男”或“女”，年龄不小于 0。

课程表（Course）中，课程号、课程名、学分、教师工号均非空，课程号为主键，课程名唯一，学分大于 0，教师工号与教师表关联。

教师表（Teacher）中，教师工号、姓名、性别、年龄、职称、手机号均非空，教师工号为主键，性别只能为“男”或“女”，年龄不小于 0，手机号唯一。

用户表（Users）中，账号、密码、手机号均非空，账号为主键，手机号唯一，角色只能为“学生”“教师”或“管理员”，学号和教师工号可空，且分别与学生表和教师表关联。

选课表（SC）中，学号、课程号均非空，成绩在 0 到 100 之间，学号和课程号共同构成主键，分别与学生表和课程表关联。

2 系统分析

2.1 设计目标

（1）构建一个高效、便捷的教务平台，满足学校、学生、教师对课程和个人信息的综合需求。

（2）确保系统具备良好的稳定性和可靠性，能够长期稳定运行。

（3）提供友好的用户界面，使用户操作简单、直观、通俗易懂，降低使用门槛。

（4）实现数据的安全存储和有效管理，防止数据丢失和泄露。

2.2 数据需求

学生信息：涵盖学号、姓名、性别、年龄、专业等关键数据，用于全面描述学生的基本情况。

课程信息：包含课程号、课程名、学分、教师工号等，明确课程的基本属性和授课教师。

教师信息：包括教师工号、姓名、性别、年龄、职称、手机号等，完整记录教师的相关信息。

选课信息：由学号、课程号、成绩等组成，反映学生的选课情况和成绩。

用户信息：包含账号、密码、学号（学生）或教师工号（教师）、手机号、角色等，用于用户身份验证和权限管理。

2.3 数据存储要求

（1）选用关系型数据库 SQLserver 来存储数据，以充分利用其强大的关系模型和数据处理能力。

（2）为每个数据表设计合理的字段类型和长度，确保数据的准确性和完整性。

（3）建立适当的索引，提高数据查询和检索的效率。

2.4 类与接口

2.4.1 界面类

AdminFrame：管理员主界面类。展示学生、课程、选课、用户和教师等信息的表格，并提供添加、修改、删除、查询和全部信息展示等操作按钮。

StudentFrame：学生主界面类。通过选项卡的形式展示个人信息、选课和成绩查询等功能。包含相关信息的表格和操作按钮，如选课、查询成绩等。

TeacherFrame：教师主界面类。同样采用选项卡布局，有个人信息、课程管理和学生成绩管理等功能。可以查看和修改学生成绩。

AddCourseFrame：用于添加课程信息的界面类。包含课程号、课程名、学分、教师工号等文本框以及确定和重置按钮。通过事件监听，实现添加课程信息到数据库的功能。

AddStudentFrame：添加学生信息的界面类。有学号、姓名、性别、年龄、专业等文本框和相应按钮。负责将学生信息插入到数据库中。

AddTeacherFrame：添加教师信息的界面类。包括教师工号、教师姓名、教师性别、教师年龄、教师职称、手机号等文本框和按钮。用于向数据库中添加教师信息。

2.4.2 数据操作类

DBUtil：这是一个工具类，用于建立数据库连接、执行 SQL 语句以及关闭数据库资源。它提供了获取 SQLserver 数据库连接的方法，以及执行增删改查等操作的通用方法。

2.4.3 业务逻辑类

LoginFrame：系统登录主界面类。提供选择管理员、学生和教师登录的按钮，是系统的入口界面。

AdminLoginFrame：管理员登录界面类。验证管理员的账号和密码，登录成功后打开管理员主界面。

AdminRegisterFrame：管理员注册界面类。实现管理员账号的注册功能，检查用户名是

否已存在。

StudentLoginFrame：学生登录界面类。验证学生的账号和密码，登录成功后设置用户名并打开学生主界面。

StudentRegisterFrame：学生注册界面类。用于学生账号的注册，检查用户名唯一性，并向数据库中插入学生和用户信息。

TeacherLoginFrame：教师登录界面类。验证教师账号和密码，登录成功后打开教师主界面。

TeacherRegisterFrame：教师注册界面类。实现教师账号的注册功能，包括插入用户和教师信息。

DeleteCourseFrame：删除课程信息的界面类。根据输入的课程号删除数据库中的课程记录。

DeleteFrame：删除信息的主界面类。提供删除学生、课程和教师信息的按钮，点击后打开相应的删除界面。

DeleteStudentFrame：删除学生信息的界面类。通过学号删除学生记录。

DeleteTeacherFrame：删除教师信息的界面类。依据教师工号删除教师信息。

UpdateCourseFrame：更新课程信息的界面类。根据输入的课程号更新课程的相关信息。

UpdateFrame：更新信息的主界面类。提供更新学生、课程和教师信息的按钮，点击后打开相应的更新界面。

UpdateStudentFram：更新学生信息的界面类。通过学号更新学生的信息。

UpdateTeacherFrame：更新教师信息的界面类。依据教师工号更新教师的信息。

2.5 类相互之间的关系

2.5.1 继承关系

所有的界面类（如 AddCourseFrame、AddStudentFrame、AdminFrame 等）都继承自 JFrame 类，这使得它们具备了窗口的基本特性和功能，能够在图形界面中显示。

2.5.2 包含关系

AdminFrame 类中包含了多个 JButton（添加、修改、删除、查询、全部信息展示按钮）

和 JTable（用于展示学生、课程、选课、用户和教师信息），这些组件共同构成了管理员界面的交互元素。

StudentFrame 类包含了 JTabbedPane，通过选项卡的形式分别包含了个人信息、选课和成绩查询等功能模块，每个模块又包含了相应的 JTable 和 JButton 等组件。

TeacherFrame 类同样包含 JTabbedPane，其选项卡下有个人信息、课程管理和学生成绩管理等功能，也包含了对应的表格和按钮等组件。

2.5.3 关联关系

界面类与业务逻辑类的关联

AdminLoginFrame 类的登录按钮点击事件通过 authenticateAdmin 方法验证管理员账号密码，登录成功后打开 AdminFrame。

StudentLoginFrame 类的登录按钮点击事件调用 authenticateStudent 方法验证学生账号密码，登录成功后设置用户名并打开`StudentFrame`。

TeacherLoginFrame 类的登录按钮点击事件利用 authenticateTeacher 方法验证教师账号密码，登录成功后打开 TeacherFrame。

DeleteCourseFrame 类的删除按钮点击事件关联 deleteCourse 方法，用于删除课程信息。

DeleteStudentFrame 类的删除按钮点击事件与 deleteStudent 方法相关联，实现删除学生信息的功能。

DeleteTeacherFrame 类的删除按钮点击事件调用 deleteTeacher 方法，完成删除教师信息的操作。

UpdateCourseFrame`类的更新按钮点击事件关联 updateCourse 方法，用于更新课程信息。

UpdateStudentFrame`类的更新按钮点击事件与 updateStudent 方法相关联，实现更新学生信息的功能。

UpdateTeacherFrame 类的更新按钮点击事件调用 updateTeacher 方法，完成更新教师信息的操作。

业务逻辑类与数据操作类的关联：

各个业务逻辑类（如 AddCourseFrame、AddStudentFrame 等）在执行数据库操作时，

都通过 DBUtil 类获取数据库连接，并使用其提供的方法执行 SQL 语句，实现与数据库的交互。

2.6 业务模块

登录模块：通过验证用户输入的账号和密码，确定用户身份并赋予相应的权限，实现不同角色的登录功能。

注册模块：允许新用户（管理员、学生、教师）进行注册，检查用户名的唯一性，确保注册信息的合法性。

信息管理模块：

（1）添加信息：提供添加学生、课程和教师信息的功能，对输入数据进行验证，确保数据的完整性和准确性。

（2）删除信息：支持删除学生、课程和教师信息，根据指定的学号或工号进行删除操作。

（3）更新信息：允许修改学生、课程和教师的相关信息，确保更新后的数据符合系统要求。

（4）查询信息：根据不同的条件（如学号、课程号等）查询学生、课程、选课等信息，并以直观的方式展示给用户。

选课模块：学生可以根据课程号选择课程，系统会检查课程是否存在以及学生是否已选过该课程，避免重复选课。

成绩管理模块：教师可以查看所授课程学生的成绩，并能够对成绩进行修改，确保成绩的准确性和及时性。

2.7 处理流程

（1）用户打开系统后，首先进入登录界面，选择相应的角色进行登录。

（2）登录成功后，根据角色进入对应的主界面，如管理员界面、学生界面或教师界面。

（3）在主界面中，用户可以通过点击操作按钮触发相应的事件，如添加信息、删除信息、查询信息等。

（4）系统接收到用户的操作请求后，根据请求的类型调用相应的业务逻辑类进行处理。

（5）业务逻辑类与数据库进行交互，执行数据的增删改查操作，并将处理结果返回给界面进行展示。

3 系统设计

3.1 开发工具和数据库

开发工具：选用 IntelliJ IDEA 作为 Java 开发环境。

DBMS：采用 SQLserver 数据库管理系统。

数据库设计辅助工具：借助 PowerDesigner 进行数据库设计和建模。

3.2 类的设计

3.2.1 界面类

1. AddCourseFrame：

（1）成员变量：

JLabel 数组用于显示课程信息标签，如课程号、课程名、学分、教师工号等。

JTextField 数组用于接收用户输入的课程信息。

JButton 数组包含确定和重置按钮，用于触发添加课程操作和清空输入框。

（2）成员方法：

addCourse 方法：首先获取用户输入的课程信息，然后构建 SQL 插入语句，通过 DBUtil 类连接数据库并执行插入操作。若插入成功，弹出提示框显示“添加课程信息成功”；否则，显示“添加课程信息失败，可能课程号已存在”。

（3）构造函数：

初始化界面，设置窗口标题、大小、位置和布局，添加标签、文本框和按钮到面板，并设置按钮的字体和背景透明效果。

2. AddStudentFrame：

（1）成员变量：

类似 AddCourseFrame，有用于显示学生信息标签的 JLabel 数组、接收用户输入的

JTextField 数组以及确定和重置按钮的 JButton 数组。

（2）成员方法：

addStudent 方法：获取学生信息，构建 SQL 插入语句，执行插入操作。成功则提示“添加学生信息成功”，失败提示“添加学生信息失败，可能学号已存在”。

（3）构造函数：

初始化学生信息添加界面。

3. AddTeacherFrame：

（1）成员变量：

包含显示教师信息标签的 JLabel 数组、接收输入的 JTextField 数组和操作按钮的 JButton 数组。

（2）成员方法：

addTeacher 方法：获取教师信息，构建 SQL 插入语句，执行插入操作。根据结果弹出相应提示框。

（2）构造函数：

构建教师信息添加界面。

4. AdminFrame：

（1）成员变量：

JButton 数组用于添加、修改、删除、查询和全部信息展示等操作按钮。

JTextField 用于输入学号进行查询。

多个 JTable 分别用于展示学生、课程、选课、用户和教师信息的表格模型。

（2）成员方法：

queryData 方法：验证学号格式，根据学号查询学生信息及相关数据，更新对应的表格模型。若学号格式错误或未找到对应信息，弹出相应提示框。

showAllData 方法：查询并展示所有学生、课程、选课、用户和教师的信息，填充到各个表格模型中。若数据库连接失败，弹出错误提示框。

（3）构造函数：

创建管理员主界面，设置窗口属性，添加左侧操作按钮面板和右侧信息表格面板，使用

JSplitPane 分割并设置分割位置。

5. StudentFrame：

（1）成员变量：

JTabbedPane 用于创建选项卡面板，包含个人信息、选课和成绩查询三个选项卡。

每个选项卡下有相应的表格模型和文本框等组件，如个人信息表格、选课表格、成绩查询表格、课程号输入文本框等。

（2）成员方法：

showPersonalInfo 方法：从数据库中获取学生个人信息并展示在个人信息表格中。若未登录或未找到对应信息，弹出提示框。

selectCourse 方法：处理选课操作，检查课程号是否为空，验证课程是否存在及是否已选，若条件满足则插入选课记录并提示选课成功，否则弹出相应错误提示。

showScore 方法：根据输入的课程号查询并展示成绩信息，若未登录或未找到成绩，弹出提示框。

showAllCourses 方法：查询并展示所有课程信息。

showSelectedCourses 方法：查询并展示已选课程信息。

（3）**构造函数**：构建学生主界面，初始化选项卡面板和各个组件，设置布局。

6.TeacherFrame：

（1）成员变量：

JTabbedPane`用于创建选项卡面板，包含个人信息、课程管理和学生成绩管理三个选项卡。每个选项卡下有相应的表格模型和按钮等组件，如个人信息表格、课程管理表格、学生成绩管理表格、修改成绩按钮等。

（2）成员方法：

loadPersonalInfo 方法：根据用户名获取教师工号，再查询并展示教师个人信息。

loadCourses 方法：根据教师工号查询并展示所授课程信息。

loadStudentGrades 方法：根据教师工号查询并展示学生成绩信息。

editStudentGrade 方法：处理修改学生成绩操作，获取选中的学生成绩记录，弹出对话框让教师输入新成绩，更新数据库并刷新成绩表格。

（3）构造函数：创建教师主界面，初始化选项卡面板和各个组件，设置布局。

3.2.2 数据操作类

DBUtil：

（1）成员变量：

private static final String URL：数据库连接 URL，指定数据库的位置和名称。

private static final String USERNAME：数据库用户名。

private static final String PASSWORD：数据库密码。

private static Connection conn：数据库连接对象，用于与数据库建立连接。

（2）成员方法：

getConnection 方法：加载数据库驱动，使用指定的 URL、用户名和密码建立数据库连接并返回。

closeResources 方法：关闭数据库连接、语句和结果集等资源，防止资源泄漏。

（3）构造函数：私有构造函数，防止类被实例化，通过静态方法提供数据库连接和资源关闭服务。

3.2.3 业务逻辑类

1.AdminLoginFrame：

（1）成员变量：

JLabel 数组用于显示账号和密码标签。

JTextField`用于输入账号。

JPasswordField 用于输入密码。

JButton 数组用于登录、返回和注册操作。

（2）成员方法：

authenticateAdmin 方法：验证管理员账号和密码是否正确，根据输入构建 SQL 查询语句，查询数据库并返回结果。

（3）构造函数：构建管理员登录界面，设置窗口属性，添加组件并设置布局。

2.AdminRegisterFrame：

（1）成员变量：

类似 AdminLoginFrame，包含账号、密码、注册和返回按钮等组件。

（2）成员方法：

isUsernameExists 方法：检查用户名是否已存在，构建 SQL 查询语句并执行查询。

registerAdmin 方法：注册管理员账号，向用户表插入账号、密码和角色信息。

（3）构造函数：创建管理员注册界面，初始化组件并设置布局。

3.DeleteCourseFrame：

（1）成员变量：

JLabel 用于显示课程号标签。

JTextField 用于输入课程号。

JButton 数组用于删除和重置操作。

（2）成员方法：

deleteCourse 方法：根据课程号删除课程信息，构建 SQL 删除语句并执行，根据结果弹出提示框。

（3）构造函数：初始化删除课程信息界面，设置窗口属性，添加组件并设置布局。

4.DeleteFrame：

（1）成员变量：

JButton 数组用于删除学生、课程和教师信息的按钮。

（2）成员方法：无（主要用于界面跳转）。

（3）构造函数：构建删除信息主界面，设置窗口属性并添加按钮。

5.DeleteStudentFrame：

（1）成员变量：

包含学号标签和文本框以及删除和重置按钮，与删除课程信息界面类似。

（2）成员方法：

deleteStudent 方法：根据学号删除学生信息，构建 SQL 删除语句并执行，根据结果弹出提示框。

（3）构造函数：创建删除学生信息界面，设置窗口属性并添加组件。

6.DeleteTeacherFrame：

（1）成员变量：

包含教师工号标签和文本框以及删除和重置按钮，与删除课程和学生信息界面类似。

（2）成员方法：

deleteTeacher 方法：根据教师工号删除教师信息，构建 SQL 删除语句并执行，根据结果弹出提示框。

（3）构造函数：初始化删除教师信息界面，设置窗口属性并添加组件。

7.LoginFrame：

（1）成员变量：

JButton 数组用于选择管理员、学生和教师登录的按钮。

（2）成员方法：

setUsername 方法：设置用户名，用于在登录成功后保存当前登录用户的用户名。

getUsername 方法：获取用户名。

（3）构造函数：构建系统登录主界面，设置窗口标题、大小和布局，添加按钮并设置布局。

8. StudentLoginFrame：

（1）成员变量：

包含账号、密码、登录、返回和注册按钮等组件，与管理员登录界面类似。

（2）成员方法：

authenticateStudent 方法：验证学生账号和密码是否正确，构建 SQL 查询语句并查询数据库。

（3）构造函数：创建学生登录界面，设置窗口属性并添加组件。

9.StudentRegisterFrame：

（1）成员变量：

包含账号、密码、学号、手机号、注册和返回按钮等组件。

（2）成员方法：

isUsernameExists 方法：检查用户名是否已存在。

registerStudent 方法：注册学生账号，向用户表插入学生信息，并向学生表插入默认的学生基本信息。

（3）构造函数：构建学生注册界面，初始化组件并设置布局。

10.TeacherLoginFrame：

（1）成员变量：

包含账号、密码、登录、返回和注册按钮等组件，与管理员和学生登录界面类似。

（2）成员方法：

authenticateTeacher 方法：验证教师账号和密码是否正确，构建 SQL 查询语句并查询数据库。

（3）构造函数：创建教师登录界面，设置窗口属性并添加组件。

11.TeacherRegisterFrame：

（1）成员变量：

包含账号、密码、工号、手机号、注册和返回按钮等组件。

（2）成员方法：

isUsernameExists 方法：检查用户名是否已存在。

registerTeacher 方法：注册教师账号，向用户表插入教师信息，并向教师表插入默认的教师基本信息。

（3）构造函数：初始化教师注册界面，设置窗口属性并添加组件。

12.UpdateCourseFrame：

（1）成员变量：

包含课程信息标签的 JLabel 数组、接收输入的 JTextField 数组和更新、重置按钮的 JButton 数组。

（2）成员方法：

updateCourse 方法：获取用户输入的课程信息，构建 SQL 更新语句，执行更新操作并根据结果弹出提示框。

（3）构造函数：构建更新课程信息界面，设置窗口属性，添加组件并设置布局。

13.UpdateFrame：

（1）成员变量：

JButton 数组用于更新学生、课程和教师信息的按钮。

（2）成员方法：无（主要用于界面跳转）。

（3）构造函数：创建更新信息主界面，设置窗口属性并添加按钮。

14.UpdateStudentFrame：

（1）成员变量：

包含学生信息标签的 JLabel 数组、接收输入的 JTextField 数组和更新、重置按钮的 JButton 数组。

（2）成员方法：

updateStudent 方法：获取学生信息，构建 SQL 更新语句，执行更新操作并根据结果弹出提示框。

（3）构造函数：构建更新学生信息界面，设置窗口属性，添加组件并设置布局。

15.UpdateTeacherFrame：

（1）成员变量：

JLabel 数组：用于显示教师信息的标签，如教师工号、教师姓名、教师性别、教师年龄、教师职称、手机号等。

JTextField 数组：用于接收用户输入的教师信息。

JButton 数组：包含更新和重置按钮，用于触发更新教师信息操作和清空输入框。

（2）成员方法：

updateTeacher 方法：首先获取用户在文本框中输入的教师信息，然后构建 SQL 更新语句，通过 DBUtil 类连接数据库并执行更新操作。如果更新成功，弹出提示框显示“更新教师信息成功”；否则，弹出提示框显示“更新教师信息失败，可能教师工号不存在”以及具体的错误信息。

（3）构造函数：初始化更新教师信息界面，为更新按钮添加动作监听器，当点击时调用 updateTeacher 方法进行教师信息更新操作；为重置按钮添加动作监听器，当点击时清空所有文本框的内容。

4 数据库设计

4.1 数据库范式分析

学生与选课之间存在多对多的关系，一个学生可以选多门课程，一门课程也可以被多个学生选。

教师与课程之间存在一对多的关系，一个教师可以教授多门课程，一门课程只能由一个教师教授。

数据库建表设计符合第三范式（3NF）

学生表（Student）

学号（Sno）作为主键，Sname、Ssex、Sage、Sdept 这些非主属性完全依赖于主键 Sno，不存在部分依赖与传递依赖情况，符合 3NF。

课程表（Course）

-以课程号（Cno）为主键，Cname、Ccredit 直接依赖于 Cno，Tno 通过外键关联 Teacher 表形成依赖关系，但无传递依赖，满足 3NF。

教师表（Teacher）

教师工号（Tno）作主键，Tname、Tsex、Tage、Ttitle、phone 均直接依赖于主键，无部分和传递依赖，符合 3NF。

用户表（Users）

-账号（ID）为唯一主键，PAWD、phone、role 直接依赖于 ID，Sno 和 Tno 虽可为空，但与其他表建立外键关联时无传递依赖问题，整体符合 3NF。

选课表（SC）

学号（Sno）和课程号（Cno）构成联合主键，Grade、LEVEL 完全依赖于联合主键，同时通过外键关联 Student 表与 Course 表保证数据完整性且无传递依赖，符合 3NF。

4.2 E-R 图设计

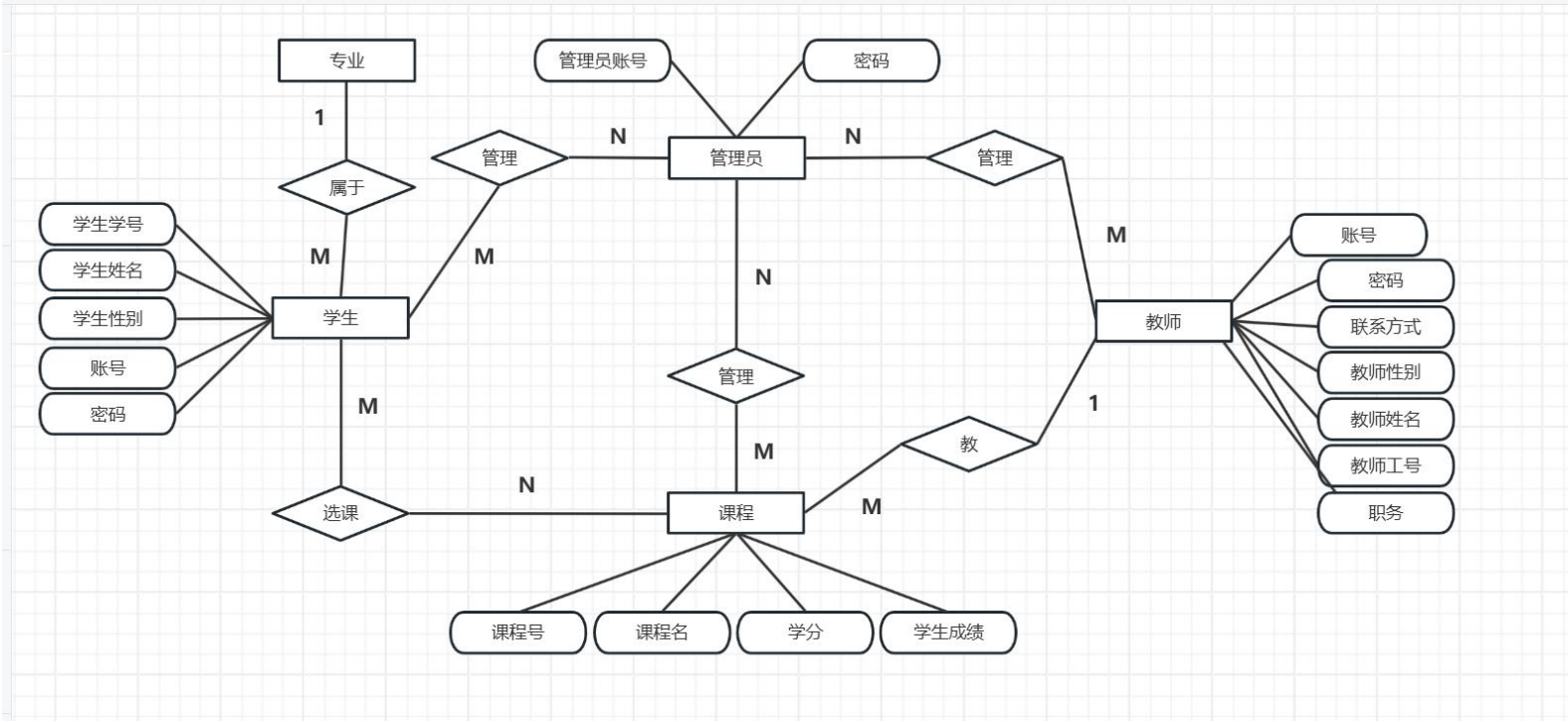
学生（Student）：学号（Sno）为主键，包含姓名（Sname）、性别（Ssex）、年龄（Sage）、专业（Sdept）属性。

课程（Course）：课程号（Cno）为主键，包含课程名（Cname）、学分（Ccredit）、教师工号（Tno）等属性，教师工号与教师（Teacher）表关联。

教师（Teacher）：教师工号（Tno）为主键，包含姓名（Tname）、性别（Tsex）、年龄（Tage）、职称（Ttitle）、手机号（phone）属性。

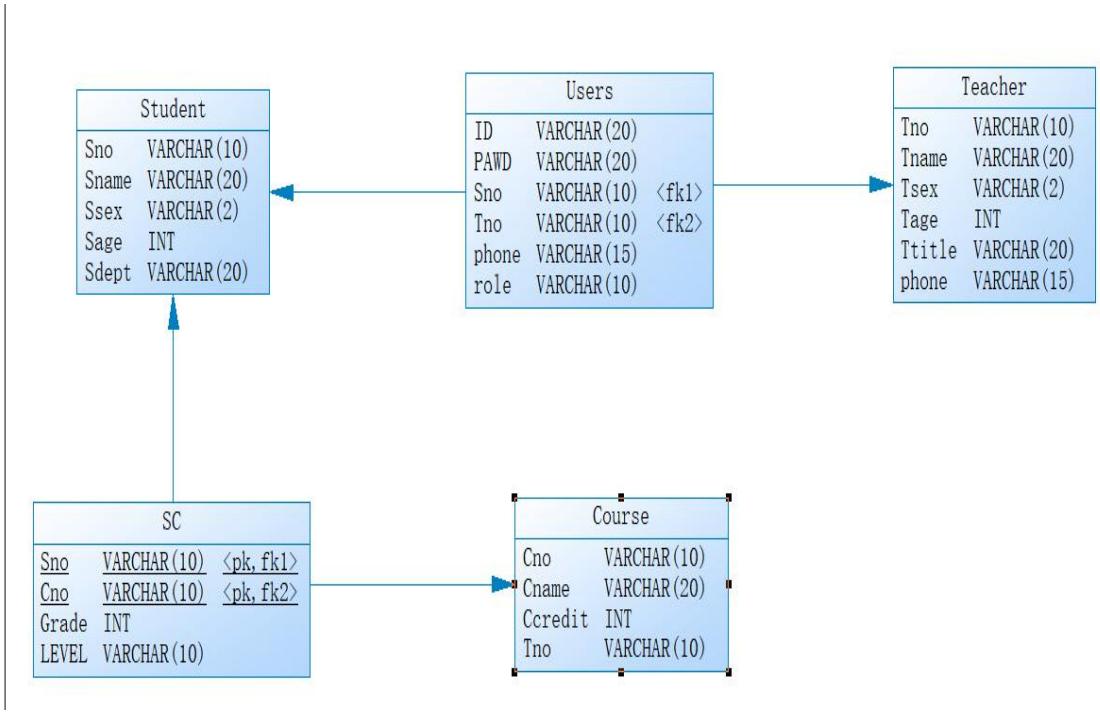
用户（Users）：账号（ID）为主键，包含密码（PAWD）、学号（Sno）、教师工号（Tno）、手机号（phone）、角色（role）等属性，学号与学生表关联，教师工号与教师表关联。

选课（SC）：由学号（Sno）和课程号（Cno）共同构成主键，包含成绩（Grade）、等



4-2 E-R 图

4.3 E-R 图转换成与具体机器上的 DBMS （SQL Server）所支持的数据模型相符合的逻辑结构



4.3 实体关系图

具体转化后的程序：

```
/*==============================================================*/
/* DBMS name:      Sybase SQL Anywhere 12                       */
/* Created on:     2024/11/28 16:27:51                          */
/*==============================================================*/
if exists(select 1 from sys.sysforeignkey where
role='FK_SC_REFERENCE_STUDENT') then
    alter table SC
       delete foreign key FK_SC_REFERENCE_STUDENT
end if;
if exists(select 1 from sys.sysforeignkey where role='FK_SC_REFERENCE_COURSE')
then
```

```
    alter table SC
       delete foreign key FK_SC_REFERENCE_COURSE
end if;
if exists(select 1 from sys.sysforeignkey where role='FK_Users_Student') then
    alter table Users
       delete foreign key FK_Users_Student
end if;
if exists(select 1 from sys.sysforeignkey where role='FK_Users_Teacher') then
    alter table Users
       delete foreign key FK_Users_Teacher
end if;

drop table if exists Course;
drop table if exists SC;
drop table if exists Student;
drop table if exists Teacher;
drop table if exists Users;
/*==============================================================*/
/* Table: Course                                                */
/*==============================================================*/
create table Course
(
   Cno                  VARCHAR(10)                    not null,
   Cname                VARCHAR(20)                    not null,
   Ccredit              INT                            not null
    constraint CKC_CCREDIT_COURSE check (Ccredit > 0),
   Tno                  VARCHAR(10)                    not null
);
/*==============================================================*/
/* Table: SC                                                    */
```

```
/*==============================================================*/
create table SC
(
   Sno                  VARCHAR(10)                    not null,
   Cno                  VARCHAR(10)                    not null,
   Grade                INT                            null
    constraint CKC_GRADE_SC check (Grade is null or (Grade between 0 and 100)),
   LEVEL                VARCHAR(10)                    null,
   constraint PK_SC primary key (Sno, Cno)
);
/*==============================================================*/
/* Table: Student                                               */
/*==============================================================*/
create table Student
(
   Sno                  VARCHAR(10)                    not null,
   Sname                VARCHAR(20)                    not null,
   Ssex                 VARCHAR(2)                     not null
    constraint CKC_SSEX_STUDENT check (Ssex in ('男','女')),
   Sage                 INT                            not null
    constraint CKC_SAGE_STUDENT check (Sage >= 0),
   Sdept                VARCHAR(20)                    not null
);
/*==============================================================*/
/* Table: Teacher                                               */
/*==============================================================*/
create table Teacher
(
   Tno                  VARCHAR(10)                    not null,
   Tname                VARCHAR(20)                    not null,
```

```
   Tsex                 VARCHAR(2)                    not null
    constraint CKC_TSEX_TEACHER check (Tsex in ('男','女')),
   Tage                 INT                           not null
    constraint CKC_TAGE_TEACHER check (Tage >= 0),
   Ttitle               VARCHAR(20)                   not null,
   phone                VARCHAR(15)                   not null
);
/*==============================================================*/
/* Table: Users                                                 */
/*==============================================================*/
create table Users
(
   ID                   VARCHAR(20)                   not null,
   PAWD                 VARCHAR(20)                   not null,
   Sno                  VARCHAR(10)                   null,
   Tno                  VARCHAR(10)                   null,
   phone                VARCHAR(15)                   not null,
   role                 VARCHAR(10)                   not null
    constraint CKC_ROLE_USERS check (role in ('学生','教师','管理员'))
);
alter table SC
   add constraint FK_SC_REFERENCE_STUDENT foreign key (Sno)
      references Student (Sno)
      on update restrict
      on delete restrict;
alter table SC
   add constraint FK_SC_REFERENCE_COURSE foreign key (Cno)
      references Course (Cno)
      on update restrict
      on delete restrict;
```

```
alter table Users
   add constraint FK_Users_Student foreign key (Sno)
      references Student (Sno)
      on update restrict
      on delete restrict;
alter table Users
   add constraint FK_Users_Teacher foreign key (Tno)
      references Teacher (Tno)
      on update restrict
      on delete restrict;
```

4.4 另一种形式的建表代码

```
-- 创建学生表
CREATE TABLE Student (
    Sno VARCHAR(10) NOT NULL PRIMARY KEY, -- 学号，非空且为主键
    Sname VARCHAR(20) NOT NULL, -- 姓名，非空
    Ssex VARCHAR(2) NOT NULL CHECK (Ssex IN ('男', '女')), -- 性别，非空且只能为'男'或'女'
    Sage INT NOT NULL CHECK (Sage >= 0), -- 年龄，非空且大于等于 0
    Sdept VARCHAR(20) NOT NULL -- 专业，非空
);
-- 创建课程表
CREATE TABLE Course (
    Cno VARCHAR(10) NOT NULL PRIMARY KEY, -- 课程号，非空且为主键
    Cname VARCHAR(20) NOT NULL UNIQUE, -- 课程名，非空且唯一
    Ccredit INT NOT NULL CHECK (Ccredit > 0), -- 学分，非空且大于 0
    Tno VARCHAR(10) NOT NULL, -- 教师工号，非空
    FOREIGN KEY (Tno) REFERENCES Teacher(Tno) -- 外键关联 Teacher 表
);
```

```
-- 创建教师表
CREATE TABLE Teacher (
    Tno VARCHAR(10) NOT NULL PRIMARY KEY, -- 教师工号，非空且为主键
    Tname VARCHAR(20) NOT NULL, -- 教师姓名，非空
    Tsex VARCHAR(2) NOT NULL CHECK (Tsex IN ('男', '女')), -- 教师性别，非空且只能为'男'或'女'
    Tage INT NOT NULL CHECK (Tage >= 0), -- 教师年龄，非空且大于等于 0
    Ttitle VARCHAR(20) NOT NULL, -- 教师职称，非空
    phone VARCHAR(15) NOT NULL UNIQUE -- 教师手机号，非空且唯一
);

-- 创建用户表
CREATE TABLE Users (
    ID VARCHAR(20) NOT NULL PRIMARY KEY, -- 账号，非空且为主键
    PAWD VARCHAR(20) NOT NULL, -- 密码，非空
    Sno VARCHAR(10), -- 学号，可为空
    Tno VARCHAR(10), -- 教师工号，可为空
    phone VARCHAR(15) NOT NULL UNIQUE, -- 手机号，非空且唯一
    role VARCHAR(10) NOT NULL CHECK (role IN ('学生', '教师', '管理员')) -- 角色，非空且只能为'学生'、'教师'或'管理员'
);

-- 添加外键约束，确保 Sno 和 Tno 的一致性
ALTER TABLE Users ADD CONSTRAINT FK_Users_Student FOREIGN KEY (Sno) REFERENCES Student(Sno);
ALTER TABLE Users ADD CONSTRAINT FK_Users_Teacher FOREIGN KEY (Tno) REFERENCES Teacher(Tno);

-- 创建选课表
CREATE TABLE SC (
```

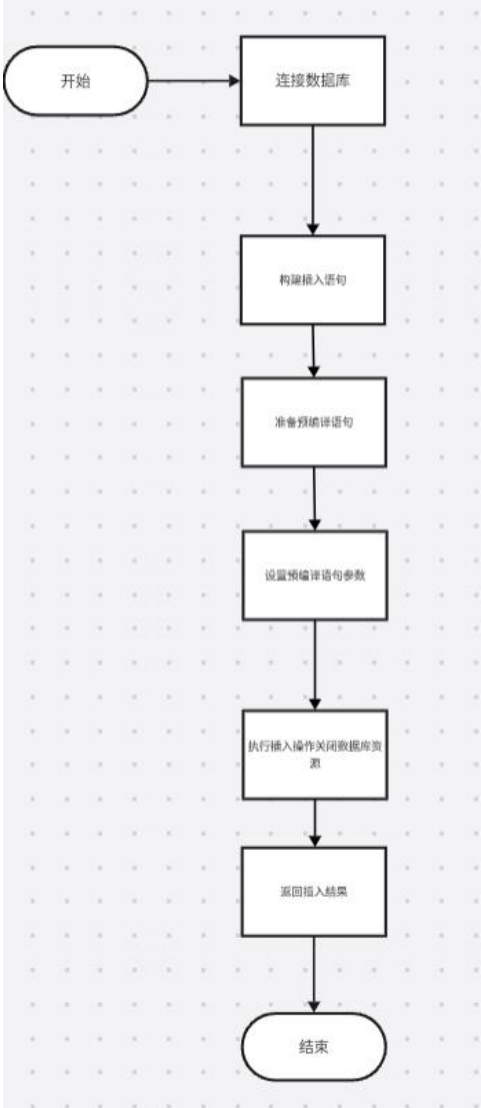
```
    Sno VARCHAR(10) NOT NULL, -- 学号，非空
    Cno VARCHAR(10) NOT NULL, -- 课程号，非空
    Grade INT CHECK (Grade >= 0 AND Grade <= 100), -- 成绩，取值范围在 0 到 100 之间
    LEVEL VARCHAR(10), -- 等级，可为空
    PRIMARY KEY (Sno, Cno), -- 联合主键，确保一个学生对一门课程只有一条选课记录
    FOREIGN KEY (Sno) REFERENCES Student(Sno), -- 外键关联 Student 表
    FOREIGN KEY (Cno) REFERENCES Course(Cno) -- 外键关联 Course 表
);
```

5 程序设计

5.1 关键数据处理算法设计

5.1.1. 数据添加算法（以添加学生信息为例）

```
// 伪代码
添加学生信息(studentData)
    连接数据库 conn = 获取数据库连接()
    构建插入语句 sql = "INSERT INTO Student (Sno, Sname, Ssex, Sage, Sdept) VALUES (?,?,?,?,?)"
    准备预编译语句 pstmt = conn.prepareStatement(sql)
    设置预编译语句参数 pstmt.setString(1, studentData.sno)
    pstmt.setString(2, studentData.sname)
    pstmt.setString(3, studentData.ssex)
    pstmt.setString(4, studentData.sage)
```



5.1.1 流程图

pstmt.setString(5, studentData.sdept)

执行插入操作 result = pstmt.executeUpdate()

关闭数据库资源 closeDatabaseResources(conn, pstmt)

返回插入结果 return result == 1

时间复杂度分析：需遍历学生表找插入位置，最坏情况遍历整张表，时间复杂度为 O(n)（n 为学生表记录数），插入数据本身时间复杂度视为 O(1)，整体为 O(n)。

空间复杂度分析：Java 端存储输入信息空间为常数 C1，数据库新记录占 O(k)（k 为单记录平均空间），整体为 O(k+C1)，近似 O(1)。

5.1.2. 删除数据算法（以删除课程信息为例）

// 伪代码

删除课程信息(courseNo)

连接数据库 conn = 获取数据库连接()

构建删除语句 sql = "DELETE FROM Course WHERE Cno =?"

准备预编译语句 pstmt = conn.prepareStatement(sql)

设置预编译语句参数 pstmt.setString(1, courseNo)

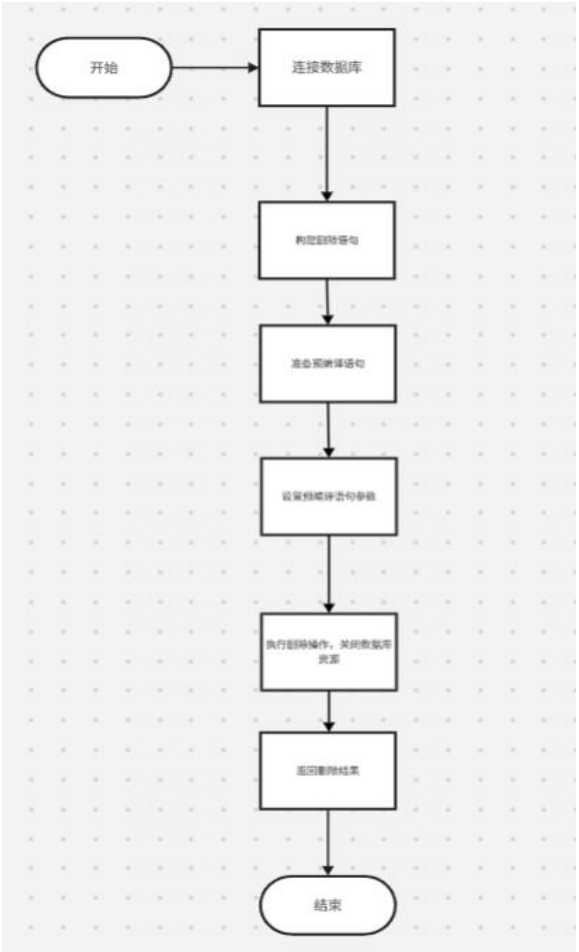
执行删除操作 result = pstmt.executeUpdate()

关闭数据库资源 closeDatabaseResources(conn, pstmt)

返回删除结果 return result == 1

时间复杂度分析：全表扫描课程表找要删除记录，时间复杂度为 O(m) (m 为课程表记录数）。若有级联删除（如涉及选课表），设选课表中相关记录平均有 p 条，级联删除时间复杂度为 O(p)，整体约为 O(m)（若 p 与 m 同量级）。

空间复杂度分析：释放课程记录空间约为 O(k) (k 为课程记录平均占用空间），Java 端常量空间为 C1，整体为 O(k + C1)，近似 O(1)。



5.1.2 流程图

5.1.3. 修改数据算法（以更新教师信息为例）

// 伪代码

更新教师信息(teacherData)

连接数据库 conn = 获取数据库连接()

构建更新语句 sql = "UPDATE Teacher SET Tname =?, Tsex =?, Tage =?, Ttitle =?, phone =? WHERE Tno =?"

准备预编译语句 pstmt = conn.prepareStatement(sql)

设置预编译语句参数 pstmt.setString(1, teacherData.tname)

pstmt.setString(2, teacherData.tsex)

pstmt.setString(3, teacherData.tage)

pstmt.setString(4, teacherData.ttitle)

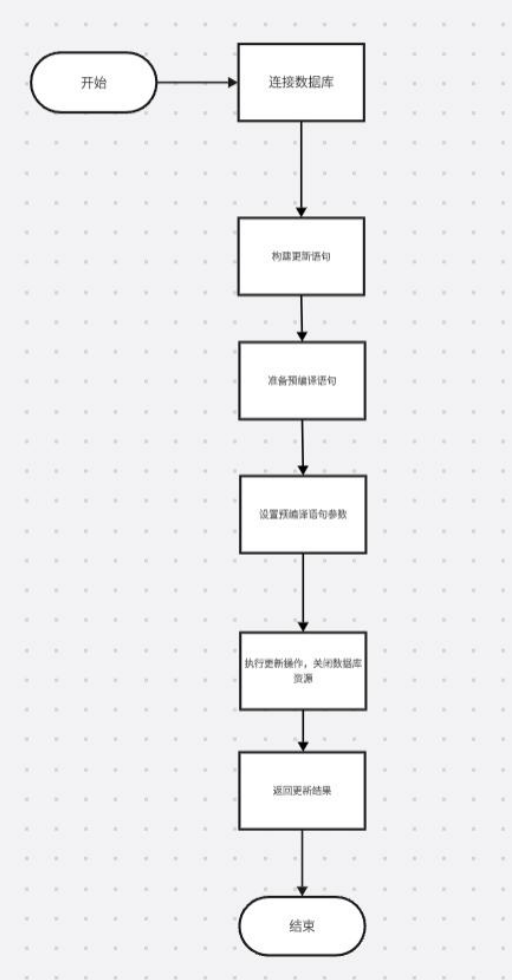
pstmt.setString(5, teacherData.tphone)

pstmt.setString(6, teacherData.tno)

执行更新操作 result = pstmt.executeUpdate()

关闭数据库资源 closeDatabaseResources(conn, pstmt)

返回更新结果 return result == 1



5.1.3 流程图

时间复杂度分析：遍历教师表找要更新记录，时间复杂度为 O(n)（q 为教师表记录数），更新字段操作视为 O(1)，整体为 O(n)。

空间复杂度分析：Java 端暂存信息空间为常数 O(C1)，整体为 O(C1)，即 O(1)。

5.1.4. 查询数据算法（以查询学生信息为例）

// 伪代码

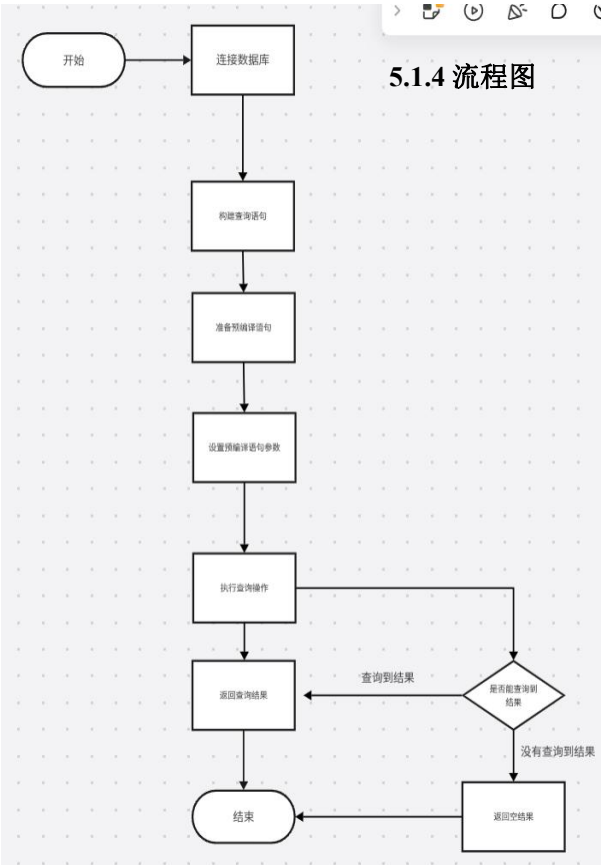
查询学生信息(sno)

连接数据库 conn = 获取数据库连接()

构建查询语句 sql = "SELECT * FROM Student WHERE Sno =?"

准备预编译语句 pstmt = conn.prepareStatement(sql)

设置预编译语句参数 pstmt.setString(1, sno)



5.1.4 流程图

执行查询操作 resultSet = pstmt.executeQuery()

若查询到结果

处理查询结果并返回（存储至数据结构）

否则

返回空结果

关闭数据库资源 closeDatabaseResources(conn, pstmt, resultSet)

时间复杂度分析：全表扫描学生表，时间复杂度为 O(n)(n 为学生表记录数）。

空间复杂度分析：少量记录查询结果存储占 O(k)（单记录平均占 k 空间），近似 O(1)；多量记录查询结果存储占 O(nk)，此时为 O(n)。

5.2 客户端程序实现

5.2.1.模块结构

登录模块：

LoginFrame 作为程序入口，利用 BorderLayout 布局，在顶部放置角色选择提示标签，底部放置管理员、学生、教师按钮，按钮设置透明背景与特定字体，通过 ActionListener 实现点击跳转至对应登录界面（如 AdminLoginFrame、StudentLoginFrame、TeacherLoginFrame）并关闭当前窗口功能，为用户提供清晰初始交互点，奠定多角色系统访问基础，提升易用性与访问效率。

以下为 LoginFrame 关键代码示例：

```
public class LoginFrame extends JFrame {
    public LoginFrame() {
        setTitle("登录");
        setSize(300, 250);
        setLocation(300, 120);
        setLayout(new BorderLayout());
        // 顶部提示信息面板
        JPanel topPanel = new JPanel();
```

```
JLabel label = new JLabel("请选择登录角色");
label.setHorizontalAlignment(SwingConstants.CENTER);
label.setFont(new Font("楷体", Font.PLAIN, 25));
topPanel.add(label);
add(topPanel, BorderLayout.NORTH);
// 底部按钮面板
JPanel bottomPanel = new JPanel();
JButton adminButton = new JButton("管理员");
JButton studentButton = new JButton("学生");
JButton teacherButton = new JButton("教师");
// 设置按钮背景透明及字体
adminButton.setContentAreaFilled(false);
studentButton.setContentAreaFilled(false);
teacherButton.setContentAreaFilled(false);
adminButton.setFont(new Font("楷体", Font.PLAIN, 17));
studentButton.setFont(new Font("楷体", Font.PLAIN, 17));
teacherButton.setFont(new Font("楷体", Font.PLAIN, 17));
bottomPanel.add(adminButton);
bottomPanel.add(studentButton);
bottomPanel.add(teacherButton);
add(bottomPanel, BorderLayout.CENTER);

// 按钮点击事件处理
adminButton.addActionListener(e -> {
    // 跳转到管理员登录界面
    new AdminLoginFrame().setVisible(true);
    dispose();
});
studentButton.addActionListener(e -> {
    // 跳转到学生登录界面
```

```
                new StudentLoginFrame().setVisible(true);
                dispose();
            });
            teacherButton.addActionListener(e -> {
                // 跳转到教师登录界面
                new TeacherLoginFrame().setVisible(true);
                dispose();
            });
        }
```

AdminLoginFrame 等登录界面类采用 null 布局精准定位账号密码输入框及登录按钮等组件，增强界面定制性与空间管理灵活性。以 AdminLoginFrame 为例，其 authenticateAdmin 方法通过 DBUtil.getConnection 获取数据库连接，构建查询语句验证管理员账号密码，依据 ResultSet 结果判定认证状态，操作完成后由 DBUtil.closeResources 关闭数据库资源连接，确保数据访问安全可靠及资源有效管理，防止资源泄露与浪费。

关键代码如下：

```
private boolean authenticateAdmin(String username, String password) throws SQLException, ClassNotFoundException {
        Connection conn = DBUtil.getConnection();
        String sql = "SELECT * FROM dbo.Users WHERE ID =? AND PAWD =? AND role = '管理员'";
        PreparedStatement pstmt = conn.prepareStatement(sql);
        pstmt.setString(1, username);
        pstmt.setString(2, password);
        ResultSet rs = pstmt.executeQuery();
        boolean authenticated = rs.next();
        DBUtil.closeResources(conn, pstmt, rs);
        return authenticated;
    }
```

StudentLoginFrame 与 TeacherLoginFrame 结构和功能类似，分别针对学生和教师角色定制登录验证流程，依角色对应表字段校验身份信息，如学生角色验证 role = '学生' 字段，保障不同角色登录安全准确，维护系统权限层级清晰稳定，各角色登录框独立构建强化账号管理与访问控制针

对性。

数据操作模块：

添加操作：AddFrame 作为添加信息总入口，以 null 布局组织添加学生、课程、教师信息按钮，按钮点击触发对应 AddStudentFrame、AddCourseFrame、AddTeacherFrame 显示。AddStudentFrame 的 addStudent 方法构建插入语句插入学生记录至 Student 表，先从文本框获取学号、姓名等信息，严格校验非空与格式合法性（如学号唯一性、年龄数值范围）后，将信息传入插入语句并执行，依据数据库操作返回值反馈添加成功或失败，确保数据完整性与一致性，防止错误或缺失值入库，维护数据库学生信息质量与参照完整性。

下面代码以添加学生为例：

```
private boolean addStudent(String sno, String sname, String ssex, String sage, String sdept) throws SQLException, ClassNotFoundException {
    Connection conn = DBUtil.getConnection();
    String sql = "INSERT INTO dbo.Student (Sno, Sname, Ssex, Sage, Sdept) VALUES (?,?,?,?,?)";
    PreparedStatement pstmt = conn.prepareStatement(sql);
    pstmt.setString(1, sno);
    pstmt.setString(2, sname);
    pstmt.setString(3, ssex);
    pstmt.setString(4, sage);
    pstmt.setString(5, sdept);
    try {
        int result = pstmt.executeUpdate();
        DBUtil.closeResources(conn, pstmt, null);
        return result == 1;
    } catch (SQLException e) {
        JOptionPane.showMessageDialog(null, "添加学生信息失败：" + e.getMessage(), "错误", JOptionPane.ERROR_MESSAGE);
        return false;
```

```
        }
    }
```

AddCourseFrame 和 AddTeacherFrame 的添加方法逻辑与 AddStudentFrame 类似，分别针对课程和教师信息插入构建对应 SQL 语句并处理插入操作，保障不同实体数据准确添加。

删除操作：DeleteFrame 以 null 布局放置删除学生、课程、教师信息按钮，点击转至如 DeleteStudentFrame 等子框架。DeleteStudentFrame 的 deleteStudent 方法依学号构建删除语句从 Student 表删记录，先获取输入学号并验证有效性，然后执行删除操作，依据操作反馈提示用户，保障删除操作精准性与可追溯性，维护数据操作事务完整性与系统数据动态平衡。

下面代码删除学生为例：

```
private boolean deleteStudent(String sno) throws SQLException, ClassNotFoundException {
        Connection conn = DBUtil.getConnection();
        String sql = "DELETE FROM dbo.Student WHERE Sno =?";
        PreparedStatement pstmt = conn.prepareStatement(sql);
        pstmt.setString(1, sno);
        try {
            int result = pstmt.executeUpdate();
            DBUtil.closeResources(conn, pstmt, null);
            return result == 1;
        } catch (SQLException e) {
            JOptionPane.showMessageDialog(null, "删除学生信息失败：" + e.getMessage(), "
错误", JOptionPane.ERROR_MESSAGE);
            return false;
        }
    }
```

DeleteCourseFrame 和 DeleteTeacherFrame 的删除方法与 DeleteStudentFrame 类似，依相应条件从对应表删除记录并处理结果反馈，确保数据删除操作严谨可靠。

修改操作：UpdateFrame 以 null 布局管理更新学生、课程、教师信息按钮，点击导向如 UpdateStudentFrame 等子框架。UpdateStudentFrame 的 updateStudent 方法依学号定位

更新学生记录，先获取界面更新后的学号、姓名等信息，构建 UPDATE 语句更新 Student 表指定字段，校验更新内容完整性与合法性后提交更新，依据数据库响应处理成功失败逻辑，保证数据更新准确性与时效性，满足数据动态变更需求。

下面代码修改学生为例：

```
private boolean updateStudent(String sno, String sname, String ssex, String sage, String sdept) throws SQLException, ClassNotFoundException {
    Connection conn = DBUtil.getConnection();
    String sql = "UPDATE dbo.Student SET Sname =?, Ssex =?, Sage =?, Sdept =? WHERE Sno =?";
    PreparedStatement pstmt = conn.prepareStatement(sql);
    pstmt.setString(1, sname);
    pstmt.setString(2, ssex);
    pstmt.setString(3, sage);
    pstmt.setString(4, sdept);
    pstmt.setString(5, sno);
    int result = pstmt.executeUpdate();
    DBUtil.closeResources(conn, pstmt, null);
    return result == 1;
}
```

UpdateCourseFrame 和 UpdateTeacherFrame 的更新方法与 UpdateStudentFrame 类似，针对课程和教师表记录构建更新语句处理更新操作，确保数据更新符合要求。

查询操作：AdminFrame 的 queryData 方法为核心查询实现，依输入学号从多表关联查询学生、用户、选课及课程信息。先验证学号格式为数字，用 DBUtil.getConnection 连库，于 Student 、 Users 、 SC 、 Course 表构建复杂嵌套查询语句，通过多 PreparedStatement 及 ResultSet 迭代处理结果集，填充对应表格模型（studentModel、userModel、scModel、courseModel）展示信息，查询结束由 DBUtil.closeResources 关闭资源，实现高效准确多表数据融合查询与可视化呈现，助力管理员深度洞察数据关联。

下面代码删除学生为例：

```
private void queryData(String sno) throws SQLException, ClassNotFoundException {
    Connection conn = DBUtil.getConnection();
```

```
            if (conn == null) {
                JOptionPane.showMessageDialog(null, "数据库连接失败，请检查配置或联系管
理员", "错误", JOptionPane.ERROR_MESSAGE);
                return;
            }
            // 验证学号格式是否为数字
            if (!sno.matches("\\d+")) {
                JOptionPane.showMessageDialog(null, "学号必须为数字，请重新输入", "错误",
JOptionPane.ERROR_MESSAGE);
                return;
            }
            // 清空查询相关表格数据
            clearRelatedTables(sno);
            // 查询学生信息
            String studentSql = "SELECT * FROM dbo.Student WHERE Sno =?";
            PreparedStatement studentPstmt = conn.prepareStatement(studentSql);
            studentPstmt.setString(1, sno);
            ResultSet studentRs = studentPstmt.executeQuery();
            PreparedStatement userPstmt = null;
            ResultSet userRs = null;
            if (studentRs.next()) {
                Vector<String> studentRow = new Vector<>();
                studentRow.addElement(studentRs.getString("Sno"));
                studentRow.addElement(studyRs.getString("Sname"));
                studentRow.addElement(studentRs.getString("Ssex"));
                studentRow.addElement(studentRs.getString("Sage"));
                studentRow.addElement(studentRs.getString("Sdept"));
                studentModel.addRow(studentRow);
                // 查询用户信息
                String userSql = "SELECT * FROM dbo.Users WHERE Sno =? OR Tno =?";
```

```
        userPstmt = conn.prepareStatement(userSql);
        userPstmt.setString(1, sno);
        userPstmt.setString(2, sno);
        userRs = userPstmt.executeQuery();
        if (userRs.next()) {
        Vector<String> userRow = new Vector<>();
        userRow.addElement(userRs.getString("ID"));
        userRow.addElement(userRs.getString("PAWD"));
        userRow.addElement(userRs.getString("Sno"));
        userRow.addElement(userRs.getString("Tno"));
        userRow.addElement(userRs.getString("phone"));
        userRow.addElement(userRs.getString("role"));
        userModel.addRow(userRow);
        }
}
// 查询选课信息
String scSql = "SELECT * FROM dbo.SC WHERE Sno =?";
PreparedStatement scPstmt = conn.prepareStatement(scSql);
scPstmt.setString(1, sno);
ResultSet scRs = scPstmt.executeQuery();
PreparedStatement coursePstmt = null;
ResultSet courseRs = null;
while (scRs.next()) {
        Vector<String> scRow = new Vector<>();
        scRow.addElement(scRs.getString("Sno"));
        scRow.addElement(scRs.getString("Cno"));
        scRow.addElement(scRs.getString("Grade"));
        scModel.addRow(scRow);
        // 查询课程信息
        String courseSql = "SELECT * FROM dbo.Course WHERE Cno =?";
```

```
            coursePstmt = conn.prepareStatement(courseSql);
            coursePstmt.setString(1, scRs.getString("Cno"));
            courseRs = coursePstmt.executeQuery();
            if (courseRs.next()) {
            Vector<String> courseRow = new Vector<>();
            courseRow.addElement(courseRs.getString("Cno"));
            courseRow.addElement(courseRs.getString("Cname"));
            courseRow.addElement(courseRs.getString("Ccredit"));
            courseModel.addRow(courseRow);
            }
        }
        DBUtil.closeResources(conn, studentPstmt, studentRs);
        DBUtil.closeResources(conn, userPstmt, userRs);
        DBUtil.closeResources(conn, scPstmt, scRs);
        DBUtil.closeResources(conn, coursePstmt, courseRs);
    }
```

数据库连接与工具模块：DBUtil 类提供获取数据库连接、关闭数据库资源等静态方法，确保数据库连接的有效管理和资源释放，被其他数据操作类频繁调用。

```
    public class DBUtil {
        Private static final String URL = "jdbc:sqlserver://127.0.0.1:1433;DatabaseName=keshe";
        private static final String USERNAME = "sa";
        private static final String PASSWORD = "123456";
        static {
            try {
                // 加载数据库驱动
                Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
            } catch (ClassNotFoundException e) {
                e.printStackTrace();
            }
```

```
    }
    // 获取数据库连接
    public static Connection getConnection() throws SQLException {
        return DriverManager.getConnection(URL, USERNAME, PASSWORD);
    }
    // 关闭数据库资源（连接、语句、结果集）
    public static void closeResources(Connection conn, PreparedStatement stmt, ResultSet
rs) {
        if (rs!= null) {
            try {
                rs.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
        if (stmt!= null) {
            try {
                stmt.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
        if (conn!= null) {
            try {
                conn.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }
```

}

数据展示模块：

如 AdminFrame 中的多个表格模型（studentModel、courseModel 等）及对应的表格组件用于展示不同类型数据；StudentFrame 的 personalInfoModel、courseModel、scoreModel 等在不同选项卡展示学生个人信息、课程信息和成绩信息；TeacherFrame 的相关表格模型展示教师个人信息、课程信息和学生成绩信息等。通过在图形用户界面布局中添加表格组件并关联模型，实现数据可视化展示。

AdminFrame 数据代码部分展示：

```
// 左侧操作按钮面板布局设置
leftPanel.setLayout(null);
addButton.setBounds(20, 30, 70, 30);
updateButton.setBounds(105, 30, 70, 30);
deleteButton.setBounds(20, 100, 70, 30);
queryButton.setBounds(105, 100, 70, 30);
inputLabel.setBounds(395, 30, 100, 25);
inputField.setBounds(500, 30, 200, 25);
showAllButton.setBounds(460, 85, 210, 30);

leftPanel.add(addButton);
leftPanel.add(updateButton);
leftPanel.add(deleteButton);
leftPanel.add(queryButton);
leftPanel.add(inputLabel);
leftPanel.add(inputField);
leftPanel.add(showAllButton);

// 右侧信息表格面板布局设置及模型关联
rightPanel.setLayout(null);
```

```
studentModel = new DefaultTableModel(null, studentTitles);
userModel = new DefaultTableModel(null, userTitles);
scModel = new DefaultTableModel(null, scTitles);
courseModel = new DefaultTableModel(null, courseTitles);
teacherModel = new DefaultTableModel(null, teacherTitles);

JTable studentTable = new JTable(studentModel);
JTable userTable = new JTable(userModel);
JTable scTable = new JTable(scModel);
JTable courseTable = new JTable(courseModel);
JTable teacherTable = new JTable(teacherModel);

JScrollPane studentScrollPane = new JScrollPane(studentTable);
JScrollPane userScrollPane = new JScrollPane(userTable);
JScrollPane scScrollPane = new JScrollPane(scTable);
JScrollPane courseScrollPane = new JScrollPane(courseTable);
JScrollPane teacherScrollPane = new JScrollPane(teacherTable);

studentScrollPane.setPreferredSize(new java.awt.Dimension(600, 100));
userScrollPane.setPreferredSize(new java.awt.Dimension(600, 100));
scScrollPane.setPreferredSize(new java.awt.Dimension(600, 100));
courseScrollPane.setPreferredSize(new java.awt.Dimension(600, 100));
teacherScrollPane.setPreferredSize(new java.awt.Dimension(600, 100));

studentScrollPane.setBounds(170, 0, 600, 100);
userScrollPane.setBounds(170, 330, 600, 100);
scScrollPane.setBounds(170, 220, 600, 100);
courseScrollPane.setBounds(170, 110, 600, 100);
teacherScrollPane.setBounds(170, 440, 600, 100);
rightPanel.add(studentScrollPane);
```

```
        rightPanel.add(userScrollPane);
        rightPanel.add(scScrollPane);
        rightPanel.add(courseScrollPane);
        rightPanel.add(teacherScrollPane);

        // 使用 JSplitPane 分割左右面板并设置分割位置
        JSplitPane splitPane = new JSplitPane(JSplitPane.VERTICAL_SPLIT, leftPanel, rightPanel);
        splitPane.setDividerLocation(150);
        add(splitPane, BorderLayout.CENTER);
```

StudentFrame 数据代码部分展示：

```
        // 个人信息选项卡创建及数据加载
        private JPanel createPersonalInfoPanel() {
            JPanel personalInfoPanel = new JPanel();
            String[] personalInfoTitles = {"学号", "姓名", "性别", "年龄", "专业"};
            personalInfoModel = new DefaultTableModel(null, personalInfoTitles);
            JTable personalInfoTable = new JTable(personalInfoModel);
            personalInfoTable.setRowHeight(20);
            JScrollPane personalInfoScrollPane = new JScrollPane(personalInfoTable);
            personalInfoScrollPane.setPreferredSize(new java.awt.Dimension(550, 100));

            try {
                // 检查用户名是否已设置，如果未设置则提示用户登录
                if (LoginFrame.getUsername() == null || LoginFrame.getUsername().isEmpty()) {
                    JOptionPane.showMessageDialog(null, "请先登录", "错误",
JOptionPane.ERROR_MESSAGE);
                    dispose();
                    return null;
                }
                showPersonalInfo(personalInfoModel);
```

```
        } catch (SQLException | ClassNotFoundException ex) {
            ex.printStackTrace();
        }
        personalInfoPanel.setLayout(null);
        personalInfoScrollPane.setBounds(50, 190, 550, 70);
        personalInfoPanel.add(personalInfoScrollPane);
        return personalInfoPanel;
    }
    // 选课选项卡创建及选课逻辑处理
    private JPanel createCourseSelectionPanel() {
        JPanel courseSelectionPanel = new JPanel();
        JLabel courseSelectionLabel = new JLabel("请输入你想选的课的课程号：");
        JTextField courseNoField = new JTextField(20);
        JButton selectCourseButton = new JButton("选课");
        courseSelectionLabel.setFont(new Font("楷体", Font.BOLD, 17));
        selectCourseButton.setFont(new Font("楷体", Font.PLAIN, 17));
        // 设置按钮背景透明
        selectCourseButton.setContentAreaFilled(false);
        courseSelectionLabel.setBounds(50, 50, 250, 30);
        courseNoField.setBounds(300, 50, 150, 30);
        selectCourseButton.setBounds(500, 50, 80, 30);
        // 课程信息表格相关组件
        String[] courseTitles = {"课程号", "课程名", "学分"};
        courseModel = new DefaultTableModel(null, courseTities);
        courseTable = new JTable(courseModel);
        courseTable.setRowHeight(20);
        JScrollPane courseScrollPane = new JScrollPane(courseTable);
        courseScrollPane.setPreferredSize(new java.awt.Dimension(550, 100));
        courseScrollPane.setBounds(50, 100, 550, 220);
        // 查询并显示所有课程信息
```

```
        try {
            showAllCourses(courseModel);
        } catch (SQLException | ClassNotFoundException ex) {
        ex.printStackTrace();
        }
        courseSelectionPanel.setLayout(null);
        courseSelectionPanel.add(courseSelectionLabel);
        courseSelectionPanel.add(courseNoField);
        courseSelectionPanel.add(selectCourseButton);
        courseSelectionPanel.add(courseScrollPane);
        // 选课按钮点击事件处理
        selectCourseButton.addActionListener(e -> {
            String courseNo = courseNoField.getText().trim();
            if (courseNo.isEmpty()) {
                JOptionPane.showMessageDialog(null, "请输入课程号", "错误",
JOptionPane.ERROR_MESSAGE);
                return;
            }
            try {
                if (selectCourse(courseNo)) {
                    JOptionPane.showMessageDialog(null, "选课成功", "提示",
JOptionPane.INFORMATION_MESSAGE);
                } else {
                    JOptionPane.showMessageDialog(null, "选课失败，可能课程号不存在
或已选过该课程", "错误", JOptionPane.ERROR_MESSAGE);
                }
            } catch (SQLException | ClassNotFoundException ex) {
                ex.printStackTrace();
            }
        });
```

```
        return courseSelectionPanel;
    }
    // 成绩查询选项卡创建及成绩查询逻辑处理
    private JPanel createScoreQueryPanel() {
        JPanel scoreQueryPanel = new JPanel();
        JLabel courseNoQueryLabel = new JLabel("课程号：");
        JTextField courseNoQueryField = new JTextField(20);
        JButton queryScoreButton = new JButton("查询");
        courseNoQueryLabel.setFont(new Font("楷体", Font.BOLD, 17));
        queryScoreButton.setFont(new Font("楷体", Font.PLAIN, 17);
        // 设置按钮背景透明
        queryScoreButton.setContentAreaFilled(false);
        courseNoQueryLabel.setBounds(50, 50, 220, 30);
        courseNoQueryField.setBounds(120, 50, 250, 30);
        queryScoreButton.setBounds(100, 150, 80, 30);
        // 已选课程信息表格相关组件
        String[] scoreTitles = {"课程号", "课程名", "成绩"};
        scoreModel = new DefaultTableModel(null, scoreTitles);
        JTable scoreTable = new JTable(scoreModel);
        scoreTable.setRowHeight(20);
        JScrollPane scoreScrollPane = new JScrollPane(scoreTable);
        scoreScrollPane.setPreferredSize(new java.awt.Dimension(550, 100);
        scoreScrollPane.setBounds(50, 200, 550, 220);
        // 查询并显示已选课程信息
        try {
            showSelectedCourses(scoreModel);
        } catch (SQLException | ClassNotFoundException ex) {
        ex.printStackTrace();
        }
        scoreQueryPanel.setLayout(null);
```

```
        scoreQueryPanel.add(courseNoQueryLabel);
        scoreQueryPanel.add(courseNoQueryField);
        scoreQueryPanel.add(queryScoreButton);
        scoreQueryPanel.add(scoreScrollPane);
        // 成绩查询按钮点击事件处理
        queryScoreButton.addActionListener(e -> {
            String courseNo = courseNoQueryField.getText().trim;
            if (courseNo.isEmpty()) {
                JOptionPane.showMessageDialog(null, "请输入课程号", "错误",
JOptionPane.ERROR_MESSAGE);
                return;
            }
            try {
                // 检查用户名是否已设置，如果未设置则提示用户登录
                if (LoginFrame.getUsername() ==null || LoginFrame.getUsername().isEmpty())
{
                JOptionPane.showMessageDialog(null, "请先登录", "错误",
JOptionPane.ERROR_MESSAGE);
                return;
                }
                showScore(courseNo, scoreModel);
            } catch (SQLException | ClassNotFoundException ex) {
                ex.printStackTrace();
            }
        });
        return scoreQueryPanel;
    }
```

TeacherFrame 数据代码部分展示：

```
// 学生成绩管理选项卡创建及成绩修改逻辑处理
private JPanel createStudentManagementPanel(String username) {
    JPanel studentManagementPanel = new JPanel();
    studentManagementPanel.setLayout(new BorderLayout());
    // 创建学生成绩表格模型并设置列标题
    String[] columnNames = {"学号", "姓名", "课程号", "课程名", "成绩"};
    studentModel = new DefaultTableModel(null, columnNames);
    JTable studentTable = new JTable(studentModel);
    studentTable.setRowHeight(20);
    studentTable.setFillsViewportHeight(true);
    // 创建滚动面板
    JScrollPane studentScrollPane = new JScrollPane(studentTable);
    studentManagementPanel.add(studentScrollPane, BorderLayout.CENTER);
    // 添加按钮功能
    JPanel buttonPanel = new JPanel();
    JButton editGradeButton = new JButton("修改成绩");
    editGradeButton.addActionListener(e -> editStudentGrade(studentTable));
    buttonPanel.add(editGradeButton);
    studentManagementPanel.add(buttonPanel, BorderLayout.SOUTH);
    // 加载学生成绩
    try {
        loadStudentGrades(username);
    } catch (SQLException | ClassNotFoundException e) {
        e.printStackTrace();
    }

    return studentManagementPanel;
}
// 修改成绩功能实现
private void editStudentGrade(JTable studentTable) {
```

```
        int selectedRow = studentTable.getSelectedRow();
        if (selectedRow!= -1) {
            // 获取当前选择的学生成绩信息
            String studentNo = (String) studentModel.getValueAt(selectedRow, 0);
            String courseNo = (String) studentModel.getValueAt(selectedRow, 2);
            String currentGrade = (String) studentModel.getValueAt(selectedRow, 4);
            // 弹出对话框让用户输入新的成绩
            String newGrade = JOptionPane.showInputDialog(this,
                    "修改成绩：请输入新的成绩", currentGrade);
            if (newGrade!= null &&!newGrade.isEmpty()) {
                // 执行修改成绩操作
                try {
                    updateGrade(studentNo, courseNo, newGrade);
                    // 成功修改后刷新表格
                    refreshStudentGrades();
                } catch (SQLException | ClassNotFoundException e) {
                    e.printStackTrace();
                }
            } else {
                JOptionPane.showMessageDialog(this, "成绩不能为空", "错误",
JOptionPane.ERROR_MESSAGE);
            }
        } else {
            JOptionPane.showMessageDialog(this, "请选择一条成绩记录进行修改", "错误",
JOptionPane.ERROR_MESSAGE);
        }
    }
    // 更新成绩到数据库方法
    private void updateGrade(String studentNo, String courseNo, String newGrade) throws
SQLException, ClassNotFoundException {
```

```
        Connection conn = DBUtil.getConnection();
        String sql = "UPDATE SC SET Grade =? WHERE Sno =? AND Cno =?";
        try (PreparedStatement stmt = conn.prepareStatement(sql)) {
            stmt.setString(1, newGrade);
            stmt.setString(2, studentNo);
            stmt.setString(3, courseNo);
            stmt.executeUpdate();
        } finally {
            DBUtil.closeResources(conn, null, null);
        }
    }
    // 刷新学生成绩方法
    private void refreshStudentGrades() throws SQLException, ClassNotFoundException {
        studentModel.setRowCount(0);
        loadStudentGrades(LoginFrame.username);
    }
```

5.2.2. 图形用户界面实现

布局管理：主要使用 BorderLayout、JSplitPane 和 null 布局等布局管理器组合。如 AdminFrame 用 JSplitPane 分割左右面板，左侧为操作按钮面板（leftPanel）用 null 布局放置按钮和输入框，右侧为信息表格面板（rightPanel）同样用 null 布局添加多个表格滚动面板，实现灵活界面布局，方便用户操作与信息查看。

事件处理：通过为按钮等组件添加 ActionListener 实现事件响应。如 AddStudentFrame 中添加学生信息按钮的点击事件处理方法获取文本框输入的学生信息，调用 addStudent 方法插入数据库，根据返回结果提示用户添加成功或失败，确保用户交互与数据操作紧密结合，提供直观反馈。

AddStudentFrame 的事件响应代码：

```
aa.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
```

```
                String sno = c.getText().trim();
                String sname = c1.getText().trim();
                String ssex = c2.getText().trim();
                String sage = c3.getText().trim();
                String sdept = c4.getText().trim();

                try {
                    if (addStudent(sno, sname, ssex, sage, sdept)) {
                        JOptionPane.showMessageDialog(null, "添加学生信息成功", "提示",
JOptionPane.INFORMATION_MESSAGE);
                    } else {
                        JOptionPane.showMessageDialog(null, "添加学生信息失败，可能学号
已存在", "错误", JOptionPane.ERROR_MESSAGE);
                    }
                } catch (SQLException | ClassNotFoundException ex) {
                    ex.printStackTrace();
                }
            }
        });
```

AdminFrame 查询按钮交互代码：

```
        queryButton.addActionListener(e -> {
            String sno = inputField.getText().trim();
            new Thread(() -> {
                try {
                    queryData(sno);
                } catch (SQLException | ClassNotFoundException ex) {
                    ex.printStackTrace();
                }
            }).start();
        });
```

程序测试

功能测试：对各功能模块单独测试。在添加功能测试中，输入合法和非法数据验证添加操作正确性与有效性，如添加学生信息时，输入合法学号、姓名等应成功添加，输入已存在学号应提示错误；删除功能测试类似，尝试删除存在和不存在的记录，检查提示信息与数据库实际变化；更新功能测试输入不同更新值验证记录是否正确修改；查询功能测试输入有效和无效查询条件，查看结果准确性与完整性，确保功能符合预期。

兼容性测试：在不同操作系统（如 Windows、Linux、Mac）和 Java 运行环境版本下测试客户端程序，检查界面显示是否正常、功能是否可用，及时处理因环境差异导致的问题，如布局错乱、字体显示异常或特定操作失败，保证程序广泛兼容性和稳定性，满足不同用户需求。

性能测试：模拟大量数据操作场景，如批量添加、删除、查询数据，监测程序响应时间、内存占用和 CPU 使用率等性能指标。优化数据库查询语句、索引设置和算法实现，减少响应时间和资源消耗，确保系统在高负载下稳定高效运行，为实际应用中大量数据处理提供保障。

5.2.3.多线程技术与网络程序设计

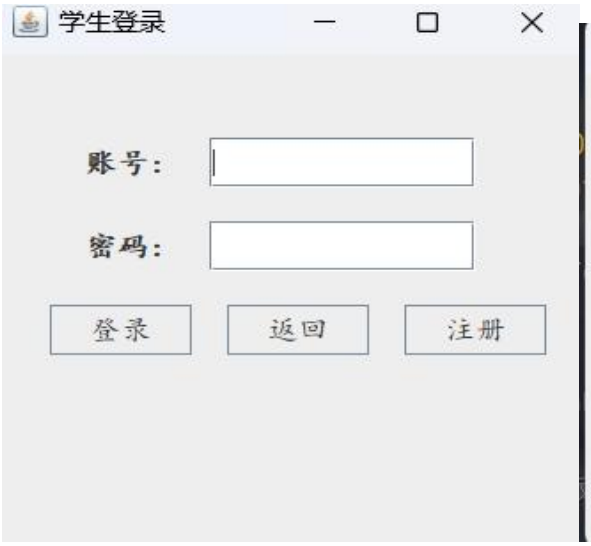
数据库操作：在所有的数据添加、删除、修改和查询操作中，代码都是在主线程中直接执行数据库连接和操作逻辑

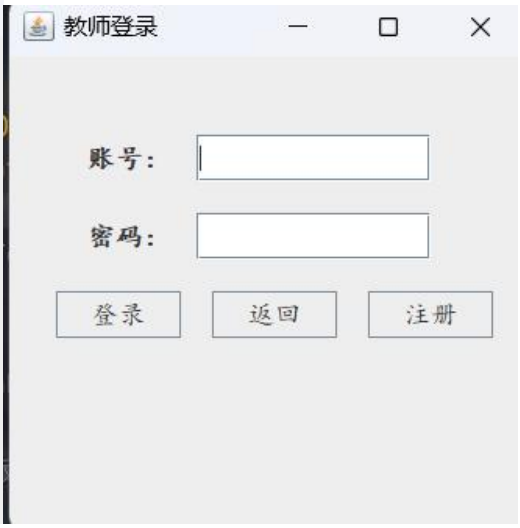
本地数据库连接方式：整个程序的数据库连接配置和操作都是基于本地数据库的直接连接模式。在各个数据操作类中，如 DBUtil 类的 getConnection 方法，是使用本地数据库驱动和连接字符串来建立与本地 SQL Server 数据库的连接。

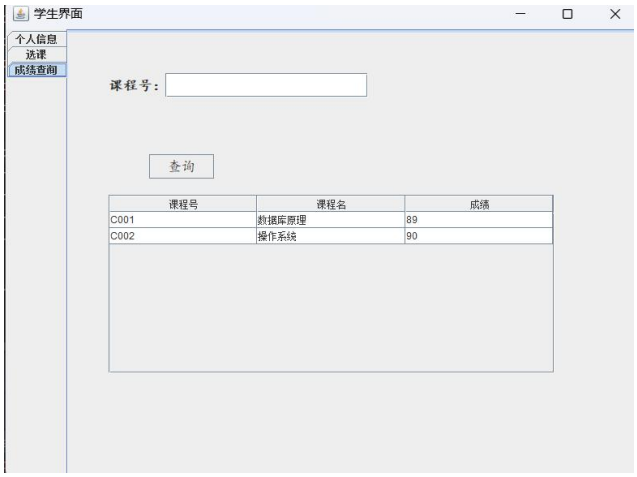
6 实现结果与测试

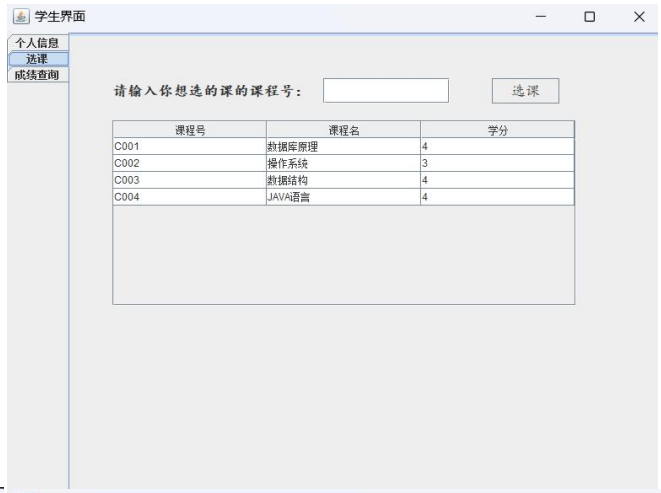

登录
请选择登录角色
管理员
学生
教师

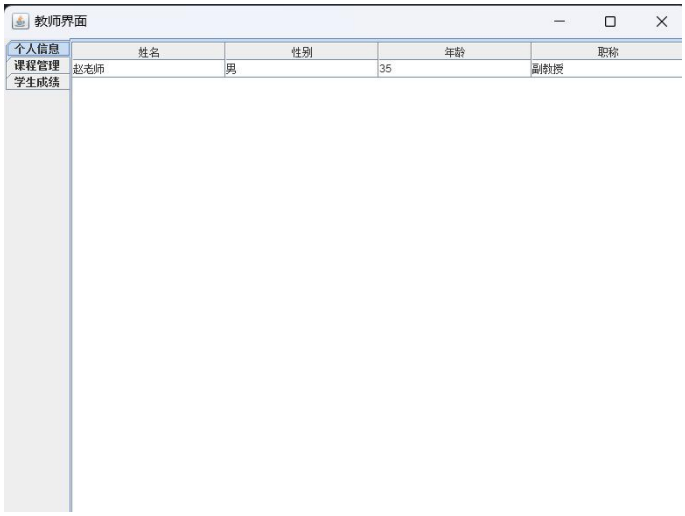

管理员登录
账号：
密码：
登录
返回
注册

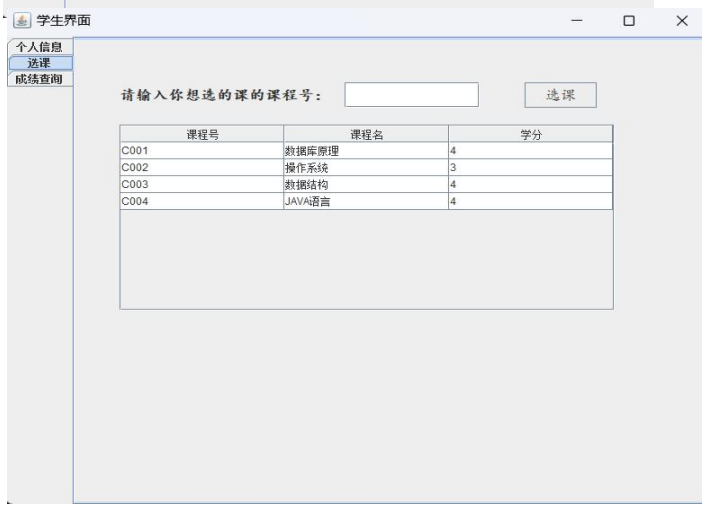

学生登录
账号：
密码：
登录
返回
注册

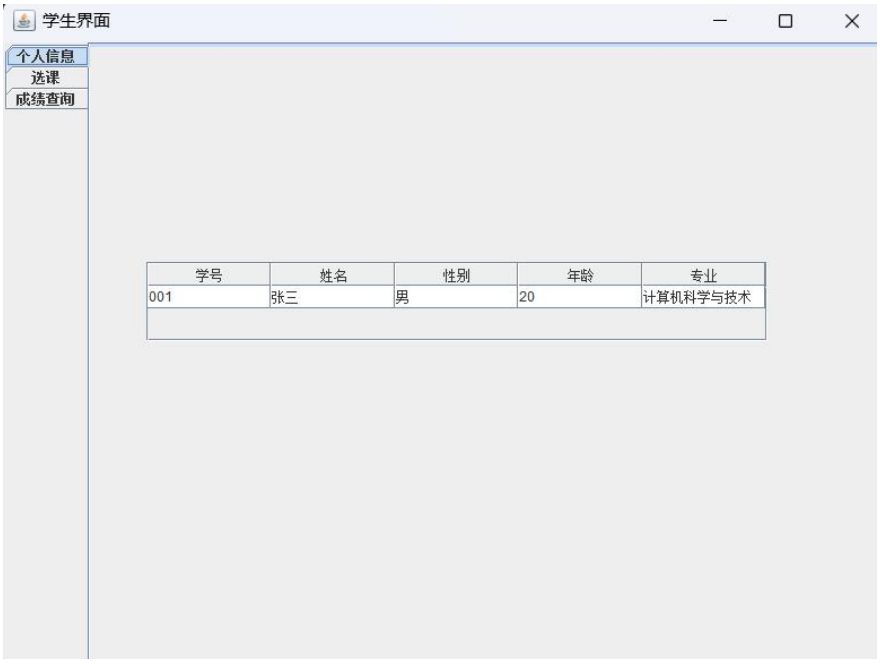

教师登录
账号：
密码：
登录
返回
注册


学生界面
个人信息
选课
成绩查询
课程号：
查询
课程号 课程名 成绩
C001 数据库原理 89
C002 操作系统 90


学生界面
个人信息
选课
成绩查询
请输入你想选的课的课程号：
选课
课程号 课程名 学分
C001 数据库原理 4
C002 操作系统 3
C003 数据结构 4
C004 JAVA语言 4


教师界面
个人信息
课程管理
学生成绩
姓名 性别 年龄 职称
赵老师 男 35 副教授


学生界面
个人信息
选课
成绩查询
请输入你想选的课的课程号：
选课
课程号 课程名 学分
C001 数据库原理 4
C002 操作系统 3
C003 数据结构 4
C004 JAVA语言 4


学生界面
个人信息
选课
成绩查询
学号 姓名 性别 年龄 专业
001 张三 男 20 计算机科学与技术

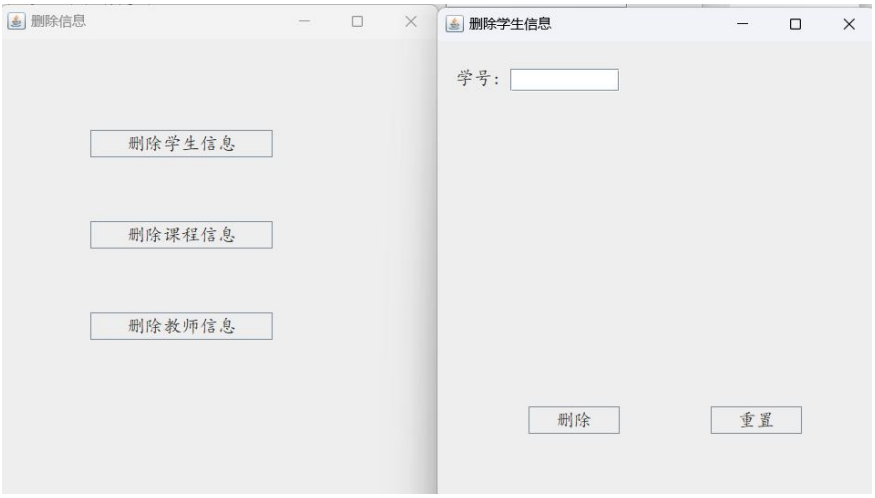
学号	姓名	性别	年龄	专业
001	张三	男	20	计算机科学与技术
002	李四	女	19	电子信息工程
003	王五	男	21	软件工程
004	李世云	男	33	计算机

课程号	课程名	学分	工号
C001	数据库原理	4	T001
C002	操作系统	3	T002
C003	数据结构	4	T003
C004	JAVA语言	4	T001

学号	课程号	成绩
001	C001	89
001	C002	90
002	C001	88
002	C003	88
003	C002	92

账号	密码	学号	工号	电话号码	角色
lsy	111	001		13712345678	学生
sa	111			13945678901	管理员
student002	123456	002		13823456789	学生
student003	123456	003		13934567890	学生
teacher002	123456		T002	13723456789	教师

教师工号	教师姓名	教师性别	教师年龄	教师职称	手机号
T001	赵老师	男	35	副教授	13812345678
T002	钱老师	女	32	讲师	13923456789
T003	孙老师	男	40	教授	13634567890
T004	刘老师	男	44	教师	11111111



7 总结

7.1 设计成果优点

（一）功能完整性

实现了涵盖学生、课程、教师、选课等多实体信息管理的全面功能集。管理员可精准操控所有数据增删改查；学生能高效管理个人信息、便捷选课与查成绩；教师可轻松处理自身信息及学生成绩。各角色功能紧密契合校园信息管理实际需求，构建起完整业务闭环，从基础数据录入到复杂关联查询与事务处理，均提供稳定高效操作途径，有力支撑校园数据管理全流程。

（二）界面设计合理性

界面布局精心规划，管理员界面的 `JSplitPane` 与操作按钮、信息表格巧妙搭配，操作逻辑清晰直观，信息展示井然有序；学生和教师界面的 `JTabbedPane` 驱动多选项卡布局，各功能区界限分明、切换流畅。组件设计注重细节，如按钮透明背景、文本框规范提示、表格行列清晰布局，配合精准事件响应机制，输入校验严格、操作反馈即时准确，大幅提升用户交互体验与操作效率，降低误操作概率，使用户能迅速上手并流畅执行任务。

（三）数据库交互稳定性

`DBUtil` 类精心封装数据库连接获取与资源释放流程，在频繁数据操作中确保连接稳定可靠，有效规避资源泄露与连接异常风险。数据操作类严格遵循数据库设计范式与约束，插入操作严谨校验数据合法性防止脏数据，删除更新精准定位目标且妥善处理关联影响，查询语句优化关联检索确保数据一致性完整性，多表复杂查询精准关联融合，为系统数据层筑牢坚实基础，保障数据持久层稳健运行与数据质量可靠。

7.2 设计成果不足

（一）性能优化空间

面对海量数据增长，部分操作性能瓶颈渐显。批量添加大量学生记录时，插入效率受限于当前单条插入模式及事务频繁提交开销，响应迟缓易引发用户等待焦虑；复杂多表关联查询中，索引利用深度不足、缓存策略粗放致使大数据集检索耗时过长，尤其在高并发查询场景下系统响应性陡降，资源占用飙升，严重影响用户体验与系统可用性，难以满足大数据量下实时性与高效性需求。

（二）可扩展性局限

系统架构对功能拓展适应性欠佳。新业务规则或功能模块引入时，如增添特殊选课规则、扩展成绩分析维度，现有分层架构与模块耦合度高，代码改动牵一发而动全身，易引发连锁错误，导致开发维护成本剧增、周期延长。数据结构灵活性不足，难以快速响应动态业务变化对数据存储与处理要求，限制系统业务创新与功能演进能力，在长期发展与功能迭代中渐显疲态。

（三）安全性强化需求

安全防护层级单薄，数据传输阶段未加密易遭嗅探窃取，敏感信息（学生成绩、教师隐私）裸传风险巨大；访问控制粒度粗放，角色权限界定存在模糊地带，可能引发越权访问操作，危及数据保密性、完整性与可用性。缺乏关键操作审计追踪机制，数据变更与重要事务操作历史难回溯，无法精准定位安全隐患源头，难以满足校园信息管理严格安全合规标准，数据资产安全面临潜在威胁。

7.3、个人学习心得

（一）技术能力提升

深度钻研 Java 桌面端开发技术栈，熟练掌握 Swing 组件体系构建复杂界面，精准运用布局管理器实现多平台适配布局，以事件驱动模型高效处理交互逻辑；精通 SQL Server 数据库设计与操作，从范式优化表结构到复杂 SQL 语句编写、索引调优，深刻理解数据持久化原理与实践技巧；熟练运用多线程并发控制改善程序响应性能，通过线程池优化资源调度提升处理效率，掌握线程安全同步机制保障数据一致性，多维度技术融合运用大幅提升综

合编程实力与问题解决能力。

（二）软件工程理解深化

严格遵循软件开发流程，从需求剖析精准提炼功能模型，架构设计精心规划模块层次与交互接口，编码实施注重代码规范风格与模块化封装，测试阶段全面覆盖功能、兼容、性能测试保障质量，部署运维考虑环境配置与系统监控优化。体会到各阶段紧密协同、迭代优化重要性，软件工程理念为开发大型复杂项目提供科学方法路径，确保软件质量、可维护性与可扩展性，提升项目管理与交付成功率。

（三）团队协作认知

虽课程设计多独立完成，但与同学交流、借鉴开源项目、参考技术论坛中体会到团队协作精髓。不同视角观点碰撞激发创新思维火花，多元技术方案共享拓宽知识视野边界，协同开发模式下分工协作、优势互补能攻克难题、加速项目推进。未来项目中应积极投身团队协作，强化沟通交流、凝聚团队共识、提升协作效能，借团队力量打造卓越软件产品，创造更大技术价值与社会效益。

附录：源程序

附录：源程序.docx

Java 与数据库应用开发实践成绩评价表

课程目标	评价依据	评价指标及标准	满分	学生自评	教师确认
课程目标 1：掌握软件工程全周期的基本方法和技术，进行问题分析、系统建模，了解影响软件设计目标和技术方案的各种因素。	课程设计报告中“1 项目描述”和“2 系统分析”部分为主，涵盖各部分。	项目描述清晰、完整。 功能完善，工作量充实。 报告按模板要求写作，（手写）字迹清楚，书写工整，或（打印）版面在边距、行距等方面处理恰当，整体可读性好。	20	20	
课程目标 2：能够针对特定需求，进行程序设计与调试，完成软件模块的设计与实现。	课程设计报告中 “5 程序设计”部分和“6 程序实现结果及测试”部分	对系统中关键的数据处理设计处理算法。程序模块功能划分合理，接口清晰、高效。解决问题运用较复杂的 Java 技术（多线程、网络、GUI 等）。	30	25	
课程目标 3：能够进行软件过程的设计或软件系统的设计与实现，完成具体系统的设计，在设计中体现创新意识。	课程设计报告中“3 系统设计”部分和“4 数据库设计”部分	确定类主要的成员以及接口的成员。对引入的每个成员要详细说明其功用、初值和操作的特点。进行数据库设计，运用视图、索引、触发器、存储过程、函数、游标等机制。	20	15	
课程目标 4：在软件设计中，特别是在选题时，能够考虑安全、健康、法律、文化及环境等制约因素。	整个设计过程中	答辩：安全、健康、法律、文化及环境等制约因素	15	15	
课程目标 5：分组协作，能够在团队中独立或合作开展工作，具有团队合作意识和良好的沟通能力。	整个设计过程中	答辩：团队合作意识	15	15	
合　计			100	90	

注：请在验收答辩前，对除“答辩”以外的其他项目完成自评。