中国科学技术大学计算机学院
《数据库系统实验报告》

实验题目：学籍管理系统

学生姓名：陈天恒

学生学号：PB21111709

完成时间：2024年6月15日

需求分析

> 1.学生/管理员登录、验证、修改密码。
> 2.添加删除学生信息，包括姓名学号年龄专业，上传个人照片，修改以上信息。
> 3.添加删除修改课程信息。
> 4.添加删除奖项和惩罚。
> 5.添加删除学生的课程成绩
> 6.查询某个学生的所有信息。

总体设计

系统模块结构

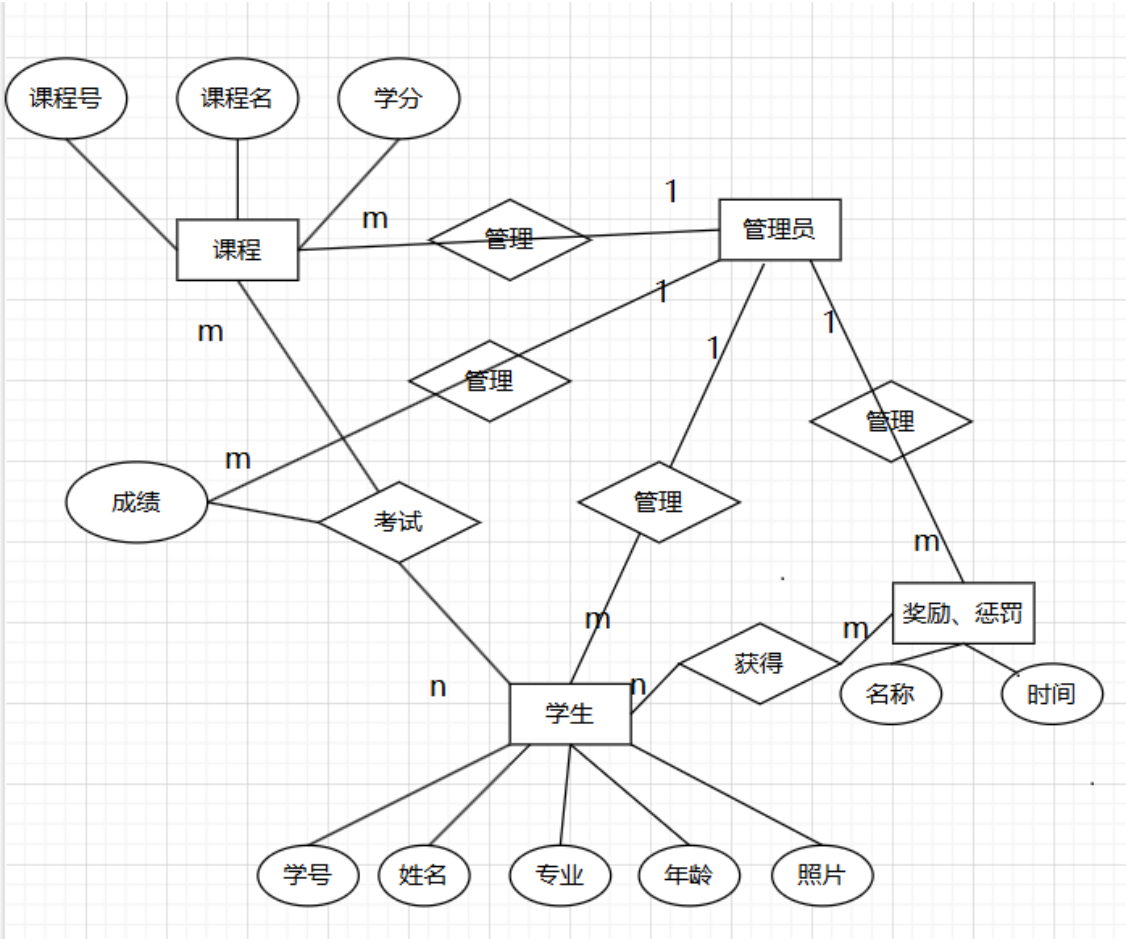
> ## 前端模块
>
> 1. **用户界面**：负责向用户呈现学籍管理系统的各个功能和信息，包括登录、学生信息查询、录入、修改等功能。
> 2. **用户交互**：处理用户在界面上的操作，包括点击按钮、填写信息等，将用户的操作转化为对后端的请求。
>
> ## 后端模块
>
> 1. **业务逻辑处理**：处理来自前端的请求，执行相应的业务逻辑，比如对学生信息的增删改查操作的实现。
> 2. **数据访问**：负责与数据库进行交互，包括数据库连接、数据查询、更新等。

系统工作流程

> 1. **用户登录**：用户通过前端界面输入用户名和密码进行登录，后端确认用户身份后进入系统。
> 2. **学生、课程信息录入**：管理员通过系统录入新的学生信息，包括姓名学号年龄专业照片奖惩等信息，以及课程、成绩信息。录入信息后，信息将被保存到数据库中。
> 3. **学生信息查询**：用户可以通过系统界面查询学生信息，根据登录的学号，系统将从数据库中检索相关信息并返回给用户。

数据库设计

> **ER图**



模式分解:

学生表（学号，姓名，年龄，专业，照片）
课程表（课程号，课程名，学分）
成绩表（学号，课程号，成绩）
获奖表（学号，奖项名称，日期）
惩罚表（学号，惩罚名称，日期）

存储过程、触发器、函数等设计思路

存储过程：修改有外键约束的主键 学生id和课程号id

```
# ./src/procedure_func_trigger.py
 DROP PROCEDURE IF EXISTS updateStudentID
 CREATE PROCEDURE updateStudentID(IN old_id CHAR(10), IN new_id CHAR(10))
 BEGIN
     ALTER TABLE Scores
     DROP FOREIGN KEY Scores_sforeign;

     UPDATE Scores
     SET StudentID = new_id
     WHERE StudentID = old_id;

     ALTER TABLE Punishtime
     DROP FOREIGN KEY Punishtime_sforeign;
```

```
    UPDATE Punishtime
    SET StudentID = new_id
    WHERE StudentID = old_id;

    ALTER TABLE Prizetime
    DROP FOREIGN KEY Prizetime_sforeign;

    UPDATE Prizetime
    SET StudentID = new_id
    WHERE StudentID = old_id;

    UPDATE Students
    SET StudentID = new_id
    WHERE StudentID = old_id;

    UPDATE users
    SET password = new_id
    WHERE username = old_id;

    UPDATE users
    SET username = new_id
    WHERE username = old_id;

    ALTER TABLE Scores
    ADD CONSTRAINT Scores_sforeign FOREIGN KEY (StudentID) REFERENCES
Students (StudentID);

    ALTER TABLE Punishtime
    ADD CONSTRAINT Punishtime_sforeign FOREIGN KEY (StudentID) REFERENCES
Students (StudentID);

    ALTER TABLE Prizetime
    ADD CONSTRAINT Prizetime_sforeign FOREIGN KEY (StudentID) REFERENCES
Students (StudentID);
 END
```

触发器： 有外键约束的情况下，学生表删除学生的同时在分数和奖惩表中删除对应id的记录

```
# ./src/procedure_func_trigger.py
CREATE TRIGGER trg_delete_student
 BEFORE DELETE ON Students
 FOR EACH ROW
 BEGIN
    SELECT COUNT(*) INTO @count FROM Scores WHERE StudentID =
OLD.StudentID;

    IF @count > 0 THEN
        DELETE FROM Scores WHERE StudentID = OLD.StudentID;
    END IF;

    SELECT COUNT(*) INTO @count FROM Punishtime WHERE StudentID =
OLD.StudentID;

    IF @count > 0 THEN
        DELETE FROM Punishtime WHERE StudentID = OLD.StudentID;
```

```
    END IF;

    SELECT COUNT(*) INTO @count FROM Prizetime WHERE StudentID =
OLD.StudentID;

    IF @count > 0 THEN
        DELETE FROM Prizetime WHERE StudentID = OLD.StudentID;
    END IF;
 END
```

函数：统计一个学生所有课程的平均分和总学分

```
# ./src/procedure_func_trigger.py
CREATE FUNCTION GetAvgPoints(student_id CHAR(10))
    RETURNS FLOAT
    DETERMINISTIC
    BEGIN
        DECLARE avg_grade FLOAT;

        SELECT AVG(score) INTO avg_grade
        FROM Scores
        WHERE student_id = StudentID;

        RETURN avg_grade;
    END

CREATE FUNCTION GetTotalPoints (student_id CHAR(10)) RETURNS INT
 DETERMINISTIC
 BEGIN
     DECLARE total_points INT;
     SELECT SUM(Classes.Point) INTO total_points
     FROM Classes
     WHERE ClassID IN (SELECT ClassID FROM Scores WHERE Scores.StudentID =
student_id);
     RETURN total_points;
 END
```

核心代码解析（可改名为对应模块，如后端实现）

仓库地址

> [w1ssen/db_lab (github.com)]

目录

```
C:.
├─.idea
│  │  .gitignore
│  │  db_lab2.iml
│  │  misc.xml
│  │  modules.xml
│  │  vcs.xml
```

```
│ │ workspace.xml
│ │
│ └─inspectionProfiles
│       profiles_settings.xml
│
├─pic
│    cypher.jpg
│    jett.jpg
│    omen.jpg
│
└─src
  │ change_password.py          -------修改密码界面
  │ classes.py                  -------课程的后端代码
  │ class_table.py              -------课程的前端代码
  │ init.py                     -------初始化数据库
  │ login.py                    -------登录界面
  │ main.py                     -------主代码
  │ main_ui.py                  -------管理员主界面
  │ prizes.py                   -------奖励的后端代码
  │ prize_table.py              -------奖励的前端代码
  │ punishments.py              -------惩罚的后端代码
  │ punish_table.py             -------惩罚的前端代码
  │ scores.py                   -------分数的后端代码
  │ score_table.py              -------分数的前端代码
  │ search_stu.py               -------查询学生信息的前端代码
  │ selects.py                  -------查询整张表的后端代码
  │ students.py                 -------学生信息增改的后端代码
  │ stu_table.py                -------学生表的前端代码
  │ procedure_func_trigger.py   -------存储过程、函数、触发器

  ├─build
  │ └─main
  │   │ Analysis-00.toc
  │   │ base_library.zip
  │   │ EXE-00.toc
  │   │ main.pkg
  │   │ PKG-00.toc
  │   │ PYZ-00.pyz
  │   │ PYZ-00.toc
  │   │ warn-main.txt
  │   │ xref-main.html
  │   │
  │   └─localpycs
  │         pyimod01_archive.pyc
  │         pyimod02_importers.pyc
  │         pyimod03_ctypes.pyc
  │         pyimod04_pywin32.pyc
  │         struct.pyc
  │
  ├─dist
```

```
│   main.exe
│
└─pycache
    change_password.cpython-310.pyc
    classes.cpython-310.pyc
    classes.cpython-38.pyc
    class_table.cpython-310.pyc
    init.cpython-310.pyc
    init.cpython-38.pyc
    login.cpython-310.pyc
    main_ui.cpython-310.pyc
    prizes.cpython-310.pyc
    prizes.cpython-38.pyc
    prize_table.cpython-310.pyc
    punish_table.cpython-310.pyc
    scores.cpython-310.pyc
    score_table.cpython-310.pyc
    search_stu.cpython-310.pyc
    selects.cpython-310.pyc
    selects.cpython-38.pyc
    students.cpython-310.pyc
    students.cpython-38.pyc
    stu_table.cpython-310.pyc
```

登录验证

```python
# ./src/login.py
class LoginWindow(QWidget):
 def __init__(self):
     super().__init__()

     self.setWindowTitle("学籍管理系统 登录")
     self.setGeometry(600, 300, 350, 200)

     self.layout = QVBoxLayout()

     self.user_label = QLabel("用户名:")
     self.layout.addWidget(self.user_label)
     self.user_input = QLineEdit()
     self.layout.addWidget(self.user_input)

     self.password_label = QLabel("密码:")
     self.layout.addWidget(self.password_label)
     self.password_input = QLineEdit()
     self.password_input.setEchoMode(QLineEdit.EchoMode.Password)
     self.layout.addWidget(self.password_input)

     self.login_button = QPushButton("登录")
     self.login_button.clicked.connect(self.check_login)
     self.layout.addWidget(self.login_button)

     self.change_button = QPushButton("修改密码")
     self.change_button.clicked.connect(self.change_password)
```

```
    self.layout.addWidget(self.change_button)

    self.setLayout(self.layout)
```

后端验证密码是否正确

```
def authenticate_admin(self, username, password):
    try:
        connection = pymysql.connect(
        )
        cursor = connection.cursor()
        cursor.execute("SELECT * FROM admin WHERE username=%s AND
password=%s", (username, password))
        result = cursor.fetchone()
        connection.close()

        if result:
            return True
        else:
            return False
    except pymysql.connect.Error as err:
        QMessageBox.critical(self, "错误", f"数据库连接失败: {err}")
        return False
```

更改密码

```
# ./src/change_password.py
def change_password(self, username, password):
            db = pymysql.connect(
            )
            cursor = db.cursor()

            sql = "UPDATE users SET password = %s WHERE username = %s"
            try:
                cursor.execute(sql, (password, username))
                db.commit()
                return True
            except Exception as e:
                print(f"发生错误: {e}")
                db.rollback()
                return False
```

显示所有学生名单的前端，包含添加删改功能

```
# ./src/stu_table.py
class Stu_Table(QWidget):
 def __init__(self, db, cursor):
    super().__init__()
    self.db = db
    self.cursor = cursor
    self.photo = None
    self.setWindowTitle('学生名单')
    self.resize(640, 480)
    self.init_ui()
```

```python
def init_ui(self):
    info = selects.select_students_all(self.db, self.cursor)
    leninfo = 4
    self.table = QTableWidget(len(info), leninfo)
    self.table.setHorizontalHeaderLabels(['学号', '姓名', '年龄', '专业'])

    # 填充表格数据
    for i in range(len(info)):
        for j in range(leninfo):
            item = QTableWidgetItem(str(info[i][j]))
            self.table.setItem(i, j, item)

    # 添加学生信息的输入框和按钮
    self.inputLayout = QHBoxLayout()
    self.student_id_input = QLineEdit()
    self.student_id_input.setPlaceholderText('学号')
    self.student_name_input = QLineEdit()
    self.student_name_input.setPlaceholderText('姓名')
    self.student_age_input = QLineEdit()
    self.student_age_input.setPlaceholderText('年龄')
    self.student_major_input = QLineEdit()
    self.student_major_input.setPlaceholderText('专业')
    # 创建上传照片按钮
    self.chooseButton = QPushButton('选择要上传的照片', self)
    self.chooseButton.clicked.connect(self.choosePhoto)
    self.add_student_button = QPushButton('添加学生')
    self.add_student_button.clicked.connect(self.add_student)

    self.inputLayout.addWidget(QLabel('学号:'))
    self.inputLayout.addWidget(self.student_id_input)
    self.inputLayout.addWidget(QLabel('姓名:'))
    self.inputLayout.addWidget(self.student_name_input)
    self.inputLayout.addWidget(QLabel('年龄:'))
    self.inputLayout.addWidget(self.student_age_input)
    self.inputLayout.addWidget(QLabel('专业:'))
    self.inputLayout.addWidget(self.student_major_input)
    self.inputLayout.addWidget(self.chooseButton)
    self.inputLayout.addWidget(self.add_student_button)


    self.inputLayout1 = QHBoxLayout()
    self.student_id_input1 = QLineEdit()
    self.student_id_input1.setPlaceholderText('学号')
    self.del_student_button = QPushButton('删除学生')
    self.del_student_button.clicked.connect(self.delete_student)
    self.change_student_Button = QPushButton('修改学生信息', self)
    self.change_student_Button.clicked.connect(self.change_student)


    self.inputLayout1.addWidget(QLabel('学号:'))
    self.inputLayout1.addWidget(self.student_id_input1)
    self.inputLayout1.addWidget(self.del_student_button)
    self.inputLayout1.addWidget(self.change_student_Button)
```

```
        layout = QVBoxLayout()
        layout.addWidget(self.table)
        layout.addLayout(self.inputLayout)
        layout.addLayout(self.inputLayout1)
        self.setLayout(layout)
```

```
# ./src/students.py
 def delete_student(self):
        student_id = self.student_id_input1.text()
        #print(student_id)

        if not (student_id):
            QMessageBox.warning(self, '输入为空')
            return

        # 将学生信息插入数据库
        try:
            students.delete_student(self.db, self.cursor, student_id)
            self.load_data()
            widget = QWidget()
            QMessageBox.information(widget, '信息', '删除成功')  # 触发的事件时弹出
会话框

        except Exception as e:
            print(f"发生错误: {e}")
            return

 def add_student(self):
        student_id = self.student_id_input.text()
        student_name = self.student_name_input.text()
        student_age = self.student_age_input.text()
        student_major = self.student_major_input.text()
        student_photo = None
        if (self.photo):
            student_photo = self.photo

        if not (student_id and student_name and student_age and student_major):
            QMessageBox.warning(self, '输入错误', '所有字段都是必填的。')
            return

            # 将学生信息插入数据库
        try:
            students.add_student(self.db, self.cursor, student_id,
student_name, student_age, student_major, student_photo)
            self.load_data()
            widget = QWidget()
            QMessageBox.information(widget, '信息', '添加成功')  # 触发的事件时弹出
会话框
        except Exception as e:
            print(f"发生错误: {e}")
            return
```

添加删改后端代码

非主键属性 直接进行修改

主键属性 利用存储过程进行修改 用触发器进行删除

```
# ./src/students.py
def add_student(db, cursor, ID, name, Age, major, image=None):
 sql1 = "INSERT INTO Students (StudentID, Name, Age, Major, Photo) VALUES
(%s, %s, %s, %s, %s)"
 sql2 = "INSERT INTO users (username, password) VALUES (%s, %s)"
 try:
     cursor.execute(sql1, (ID, name, Age, major, image))
     cursor.execute(sql2, (ID, ID))
     db.commit()
 except Exception as e:
     print(f"发生错误: {e}")
     db.rollback()
     return


# 对学生的属性进行修改
def change_student(db, cursor, ID, num, newinf):
 if (num == 1):
     sql = "UPDATE Students SET Name = %s WHERE StudentID = %s"
 if (num == 2):
     sql = "UPDATE Students SET Age = %s WHERE StudentID = %s"
 if (num == 3):
     sql = "UPDATE Students SET Major = %s WHERE StudentID = %s"
 if (num == 4):
     sql = "UPDATE Students SET Photo = %s WHERE StudentID = %s"
 try:
     cursor.execute(sql, (newinf, ID))
     db.commit()
 except Exception as e:
     print(f"发生错误: {e}")
     db.rollback()
     return

# 删除学生
def delete_student(db, cursor, ID):
 sql = "DELETE FROM Students WHERE StudentID = %s"
 sql2 = "DELETE FROM users WHERE username = %s"
 try:
     cursor.execute(sql, (ID, ))
     cursor.execute(sql2, (ID,))
     db.commit()
 except Exception as e:
     print(f"发生错误: {e}")
     db.rollback()
     return
```

查询某个学生的个人信息和成绩 前端

合并了修改功能 方便学生修改个人信息

```
# ./src/search_stu.py
 class ResultWindow(QWidget):
     def __init__(self, db, cursor, id):
         super().__init__()
         self.db = db
         self.cursor = cursor
         self.id = id
         info = selects.select_student_baseinfo(self.db, self.cursor,
self.id)
         self.name = info[1]
         self.age = info[2]
         self.major = info[3]
         self.photo = info[4]
         self.setWindowTitle('学生详细信息')
         self.resize(720, 720)
         self.init_ui()

     def init_ui(self):
         scoreinfo = selects.select_student_score(self.db, self.cursor,
self.id)
         punishinfo = selects.select_student_punish(self.db, self.cursor,
self.id)
         prizeinfo = selects.select_student_prizes(self.db, self.cursor,
self.id)
         if (len(scoreinfo) != 0):
             total_points_info = scores.get_total_points(self.db,
self.cursor, self.id)
             total_points = total_points_info[0]
         else:
             total_points = 0

         # 创建 QLabel 对象
         self.label1 = QLabel(f"学号：{self.id}")
         self.label2 = QLabel(f"姓名：{self.name}")
         self.label3 = QLabel(f"年龄：{self.age}")
         self.label4 = QLabel(f"专业：{self.major}")
         self.label5 = QLabel(f"当前已获总学分：{total_points}")
         try:
             if(self.photo):
                 image = QImage.fromData(QByteArray(self.photo))
                 pixmap = QPixmap(image)

                 # 将图片大小限制为 200x200 像素
                 fixed_size = QSize(200, 200)
                 scaled_pixmap = pixmap.scaled(fixed_size,
Qt.AspectRatioMode.KeepAspectRatio,

Qt.TransformationMode.SmoothTransformation)

                 self.label6 = QLabel(self)
                 self.label6.setPixmap(scaled_pixmap)
```

```
            else:
                self.label6 = None
        except Exception as e:
            print(f"发生错误: {e}")
            return

        # 创建表格
        self.table1 = QTableWidget(len(scoreinfo), 3)
        self.table1.setHorizontalHeaderLabels(['课程号', '学分', '成绩'])
        # 填充表格数据
        for i in range(len(scoreinfo)):
            for j in range(len(scoreinfo[i])):
                item = QTableWidgetItem(str(scoreinfo[i][j]))
                self.table1.setItem(i, j, item)
        # 创建表格
        self.table2 = QTableWidget(len(prizeinfo), 2)
        self.table2.setHorizontalHeaderLabels(['奖项', '时间'])
        # 填充表格数据
        for i in range(len(prizeinfo)):
            for j in range(1,3):
                item = QTableWidgetItem(str(prizeinfo[i][j]))
                self.table2.setItem(i, j-1, item)

        # 创建表格
        self.table3 = QTableWidget(len(punishinfo), 2)
        self.table3.setHorizontalHeaderLabels(['惩罚', '时间'])
        # 填充表格数据
        for i in range(len(punishinfo)):
            for j in range(1,3):
                item = QTableWidgetItem(str(punishinfo[i][j]))
                self.table3.setItem(i, j-1, item)

        # 创建关闭按钮
        self.closeButton = QPushButton('修改信息', self)
        self.closeButton.clicked.connect(self.change)

        # 设置布局
        layout = QVBoxLayout()
        layout_base = QHBoxLayout()
        layout_base1 = QVBoxLayout()


        layout_base.addWidget(self.label6)
        layout.addLayout(layout_base)

        layout1 = QHBoxLayout()
        layout1.addWidget(self.label1)
        layout1.addWidget(self.label2)
        layout2 = QHBoxLayout()
        layout2.addWidget(self.label3)
        layout2.addWidget(self.label4)
        layout3 = QHBoxLayout()
        layout3.addWidget(self.label5)
        layout_base1.addLayout(layout1)
        layout_base1.addLayout(layout2)
```

```
        layout_base1.addLayout(layout3)
        layout_base.addLayout(layout_base1)

        layout.addWidget(self.table1)
        table_layout = QHBoxLayout()
        table_layout.addWidget(self.table2)
        table_layout.addWidget(self.table3)
        layout.addLayout(table_layout)
        bottomLayout = QHBoxLayout()
        bottomLayout.addWidget(self.closeButton)
        layout.addLayout(bottomLayout)
        self.setLayout(layout)
```

查询某个学生的个人信息和成绩 后端

```
# ./src/selects.py
# 查询某个学生的基本信息
def select_student_baseinfo(db, cursor, ID):
    sql = "SELECT * FROM Students WHERE StudentID = %s"
    try:
        cursor.execute(sql, (ID, ))
        student_info = cursor.fetchone()
        return student_info
    except Exception as e:
        print(f"发生错误: {e}")
        return

# 查询某个学生获得的奖项
def select_student_prizes(db, cursor, ID):
    sql = "SELECT * FROM Prizetime WHERE StudentID = %s"
    try:
        cursor.execute(sql, (ID, ))
        student_info = cursor.fetchall()
        return student_info
    except Exception as e:
        print(f"发生错误: {e}")
        return

# 查询某个学生获得的惩罚
def select_student_punish(db, cursor, ID):
    sql = "SELECT * FROM Punishtime WHERE StudentID = %s"
    try:
        cursor.execute(sql, (ID, ))
        student_info = cursor.fetchall()
        return student_info
    except Exception as e:
        print(f"发生错误: {e}")
        return

# 查询某个学生的全部成绩
def select_student_score(db, cursor, ID):
    sql = "SELECT c.ClassId, c.name, c.Point, s.Score FROM classes c JOIN
scores s ON c.ClassId = s.ClassId WHERE s.StudentID = %s"
    try:
```

```
        cursor.execute(sql, (ID, ))
        Scores = cursor.fetchall()
        return Scores
    except Exception as e:
        print(f"发生错误: {e}")
        return
```

实验与测试

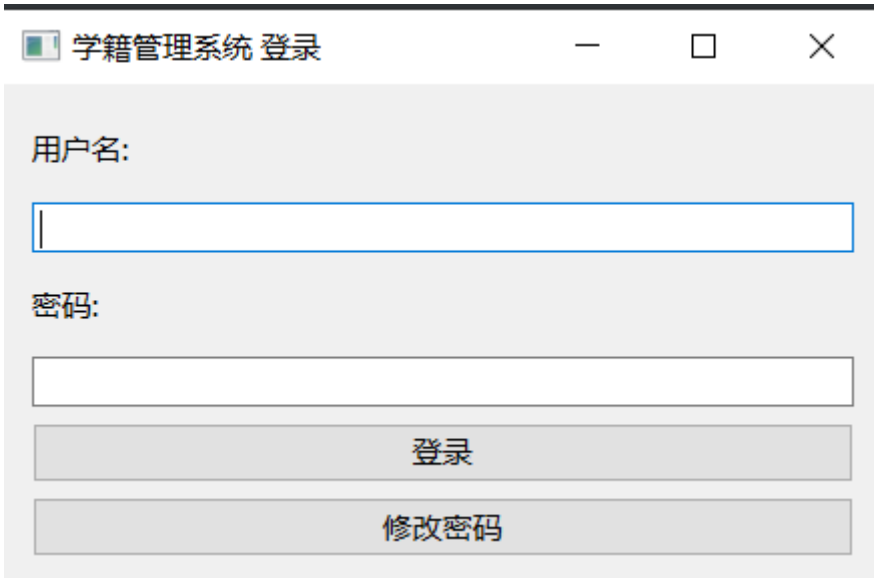
依赖

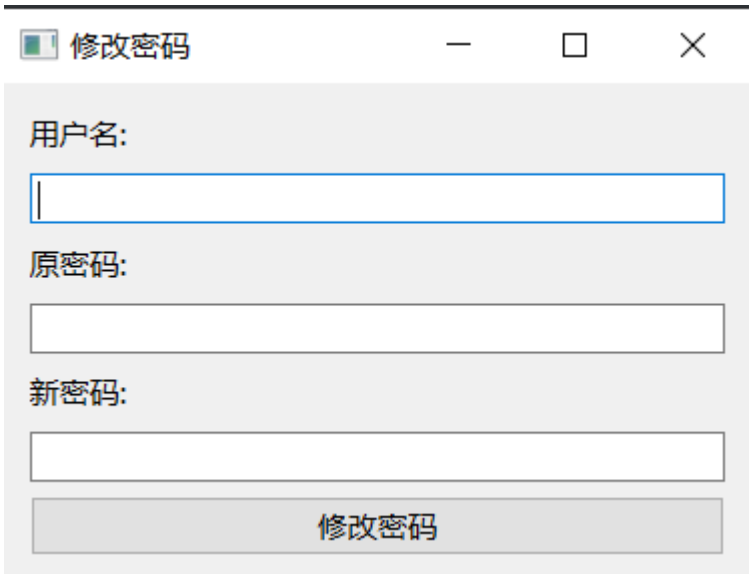
> pyqt6、mysql、python3.10

部署

> 打开数据库后运行main.exe
>
> 或者命令行
>
> ```
> mysql -u root -p
> python ./src/main.py
> ```

实验结果

> 登录
>
> 
>
>
> 修改密码

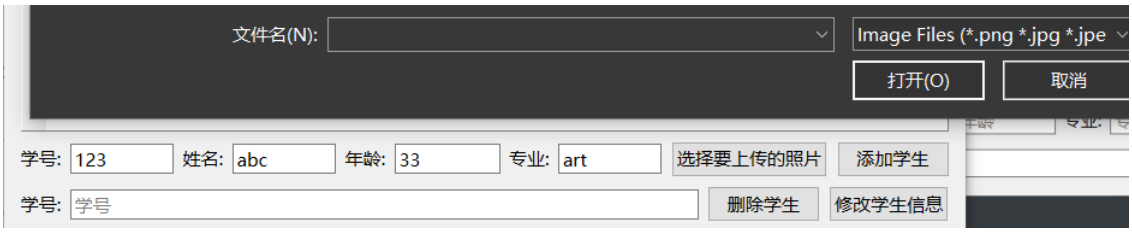

修改密码
用户名:
原密码:
新密码:
修改密码

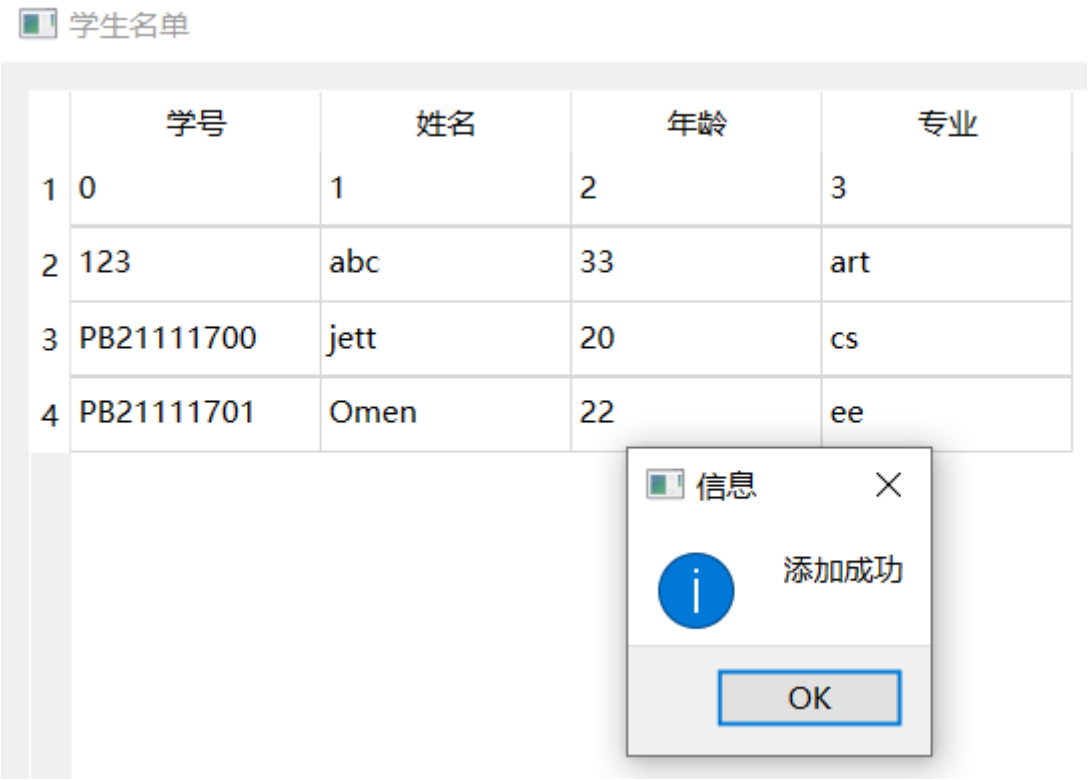
管理员主界面

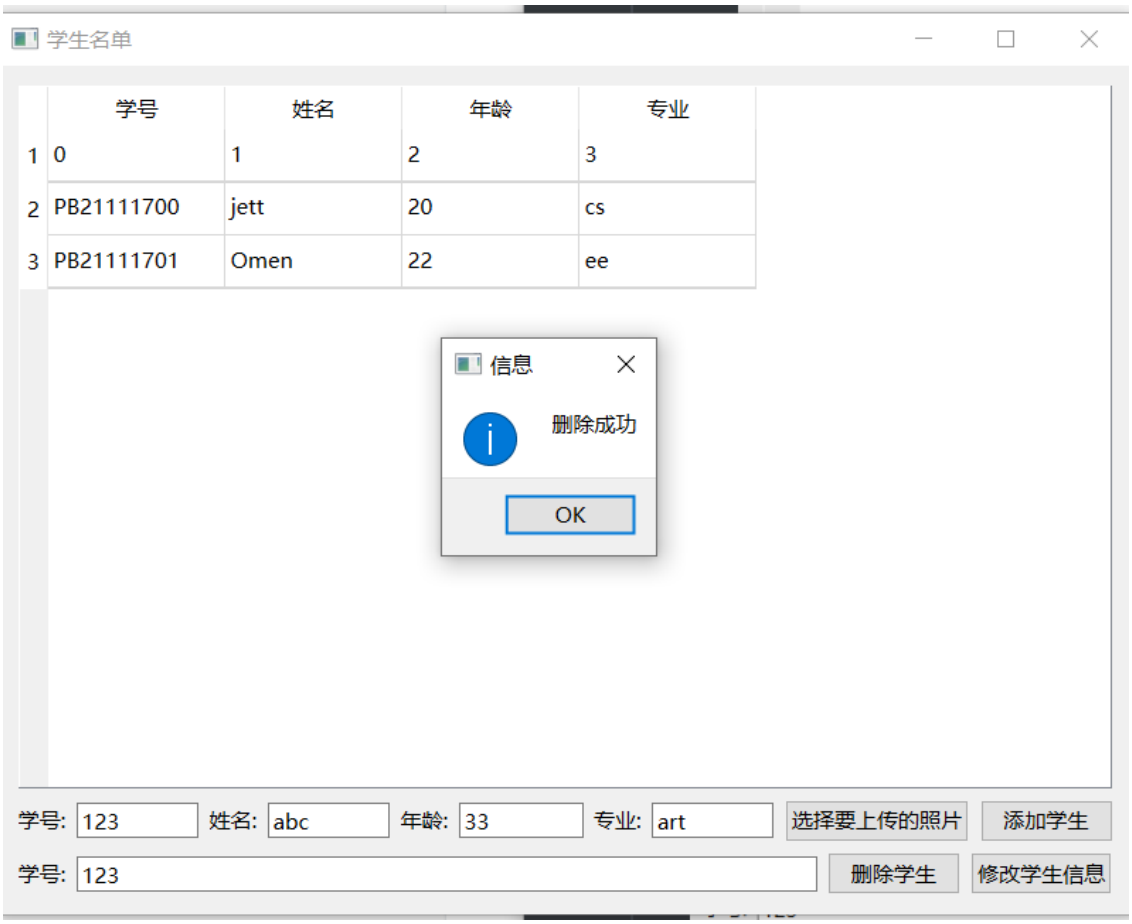

学籍管理系统 管理员界面
查看全部学生
查看全部奖项
查看成绩信息
查看全部课程
查看全部惩罚
查询学生信息

增删改查


学生名单
学号
姓名
年龄
专业
1 0 1 2 3
2 PB21111700 jett 20 cs
3 PB21111701 Omen 22 ee
学号: 学号
姓名: 姓名
年龄: 年龄
专业: 专业
选择要上传的照片
添加学生
学号: 学号
删除学生
修改学生信息


文件名(N):
Image Files (*.png *.jpg *.jpe
打开(O)
取消
学号: 123
姓名: abc
年龄: 33
专业: art
选择要上传的照片
添加学生
学号: 学号
删除学生
修改学生信息

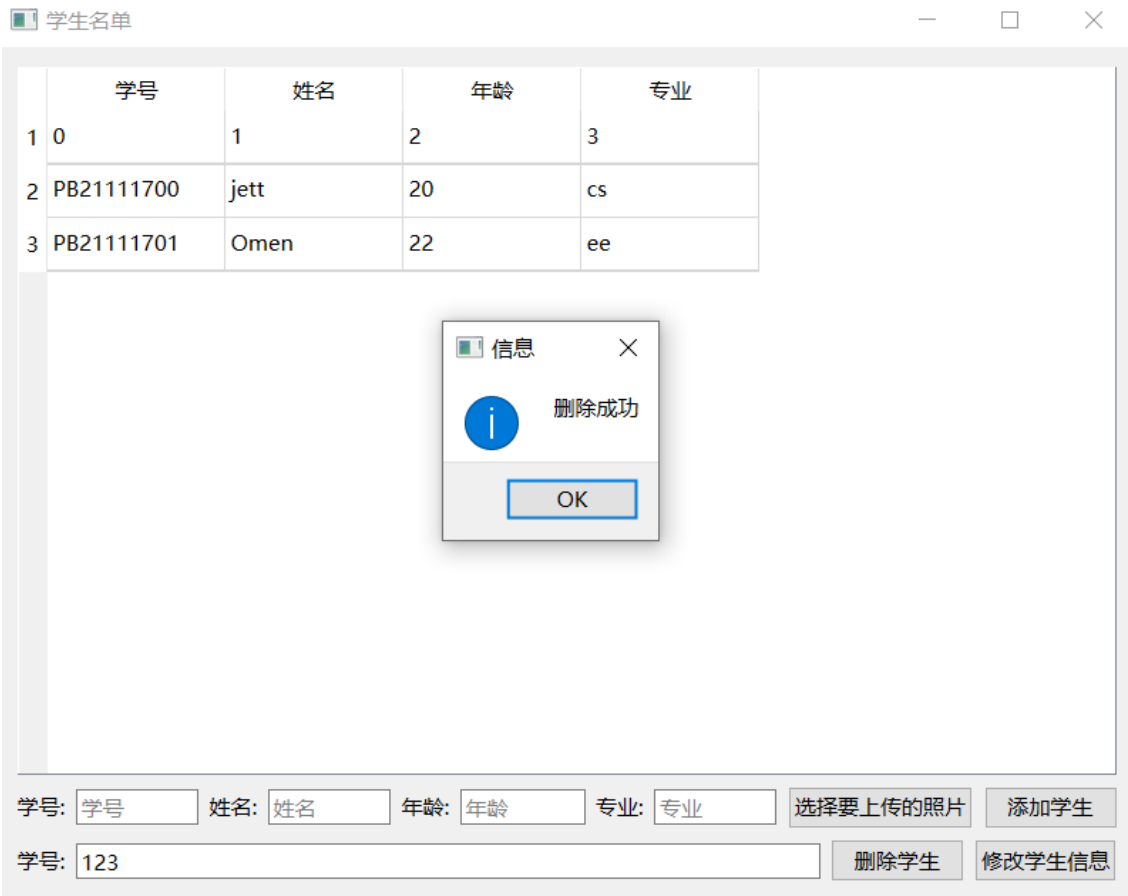

学生名单
学号
姓名
年龄
专业
1 0 1 2 3
2 123 abc 33 art
3 PB21111700 jett 20 cs
4 PB21111701 Omen 22 ee
信息
添加成功
OK



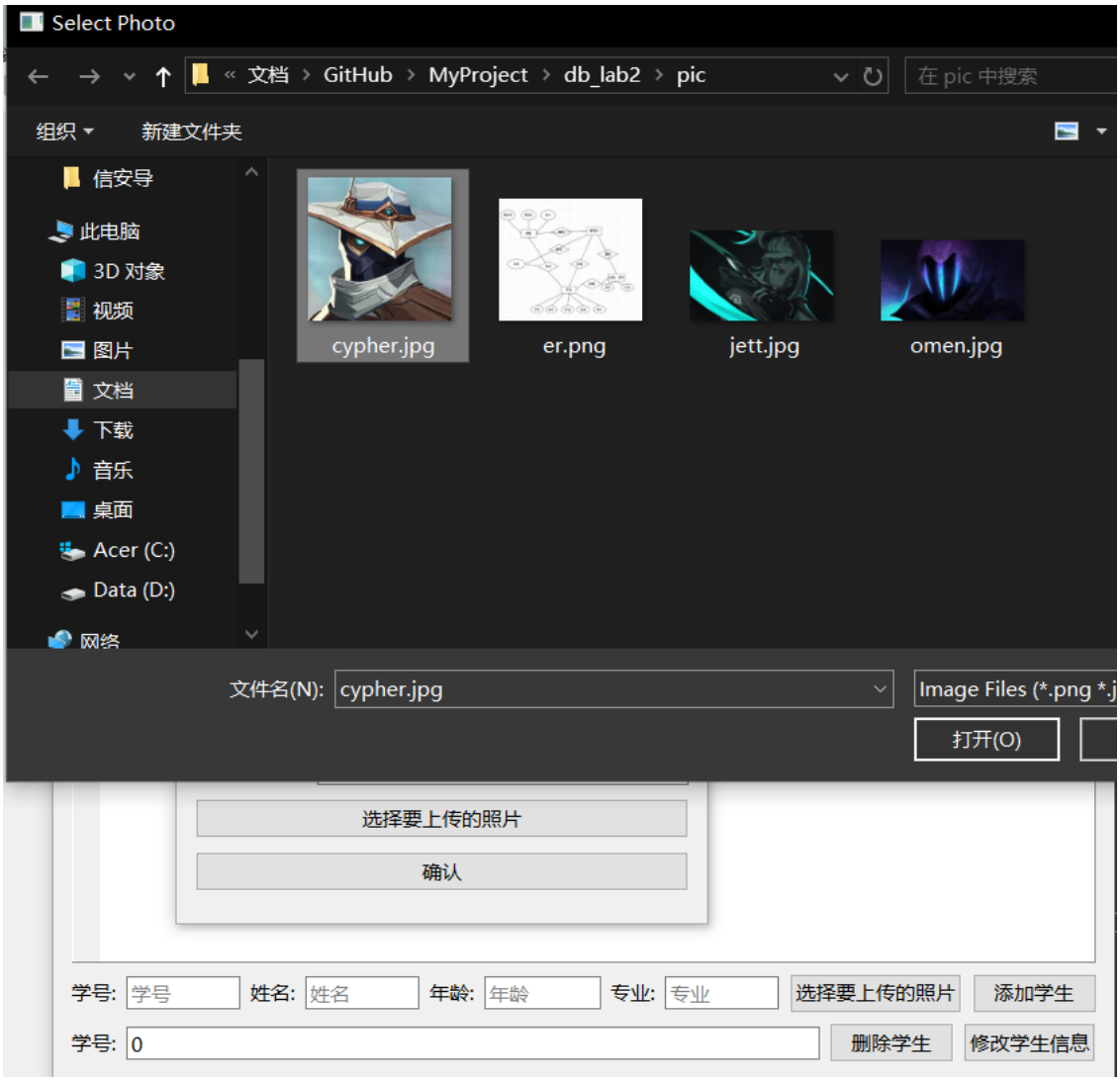
验证存储过程：修改有外键约束的学号，同时修改学生表和成绩表、奖惩表。

	学号	课程号	成绩
1	0	cs186	66
2	12345	cs152	77

触发器：删除学生时触发，删除该学生在其他表的记录。



文件管理：实现了图片管理



查询该学生的信息

平均分和学分两个**函数**统计正常

参考

[maicss/PyQt-Chinese-tutorial: PyQt6中文教程 (github.com)]