

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：李乐禛

学生学号：PB21111716

完成时间：2024年5月5日

需求分析

本次实验我选择的课题是学籍管理系统。

在各个学校对学籍管理的场景下，可以应用这套学籍管理系统。

所需的功能如下：

- 1.添加删除学生，录入基本信息，包括学生相片。
- 2.专业变更等基本信息变更。
- 3.奖惩情况管理，为学生添加删除奖项。
- 4.添加删除课程。
- 5.所选课程管理，添加或删除某名学生的选课。
- 6.课程成绩管理，修改选课成绩
- 7.查询以上内容。

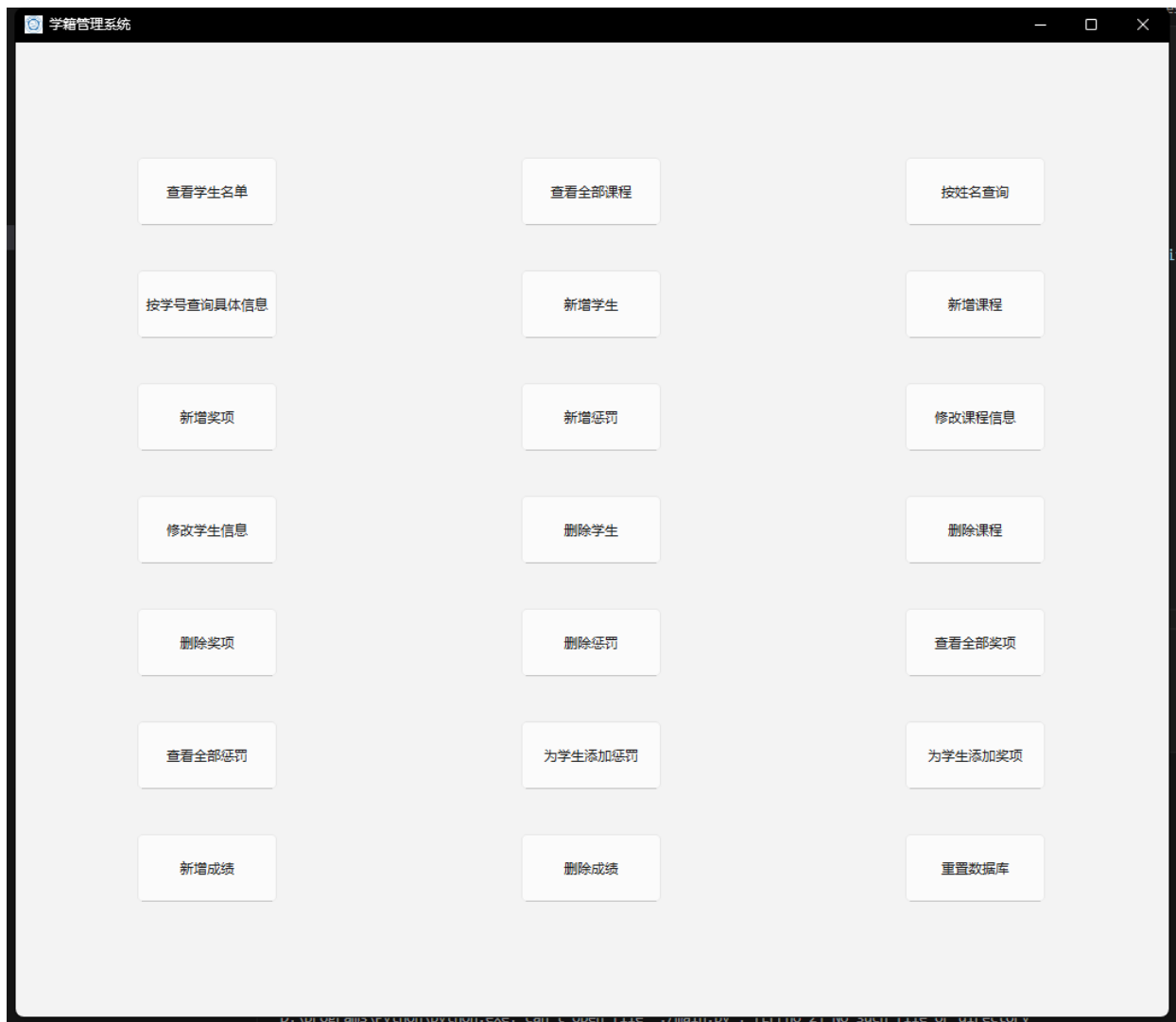
总体设计

系统模块结构

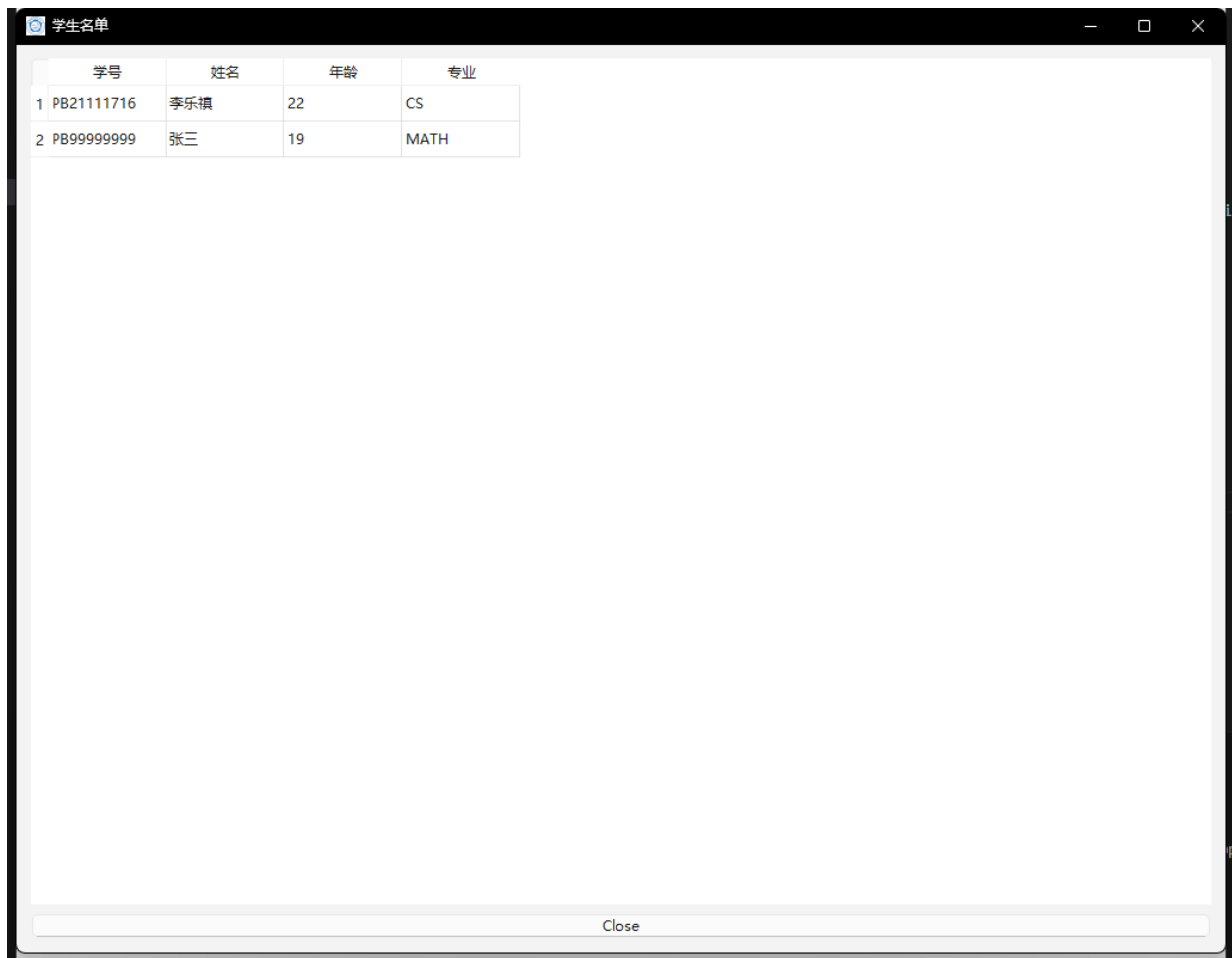
本次实验我的代码分前端和后端两大模块。结构是以前端为主的两层式结构。

系统工作流程

打开主窗口，有21个选项可选。



在查看全部的几个选项时，会直接出现结果表格。



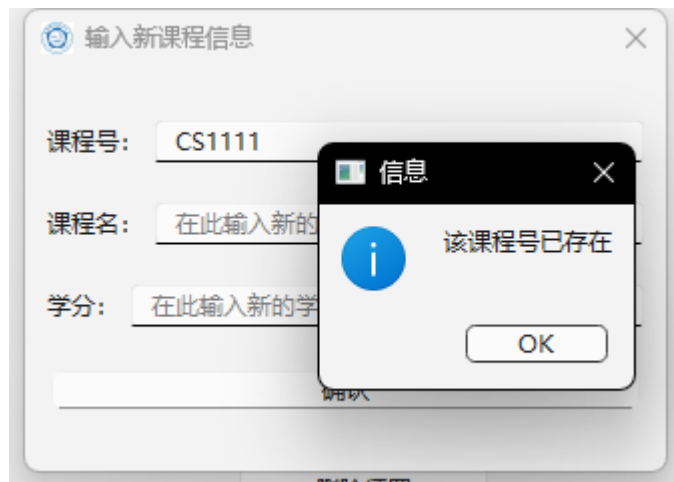
选择要输入内容的选项时，则会弹出弹窗。

The screenshot shows a dialog box titled "输入新课程信息" (Input New Course Information). It contains three input fields and a confirmation button:

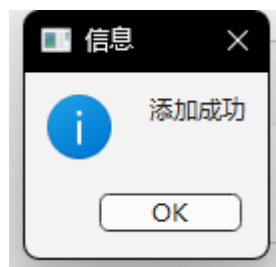
- 课程号: 在此输入新的课程号...
- 课程名: 在此输入新的课程名...
- 学分: 在此输入新的学分...

At the bottom of the dialog box, there is a button labeled "确认" (Confirm).

如果有内容不符则会报错。



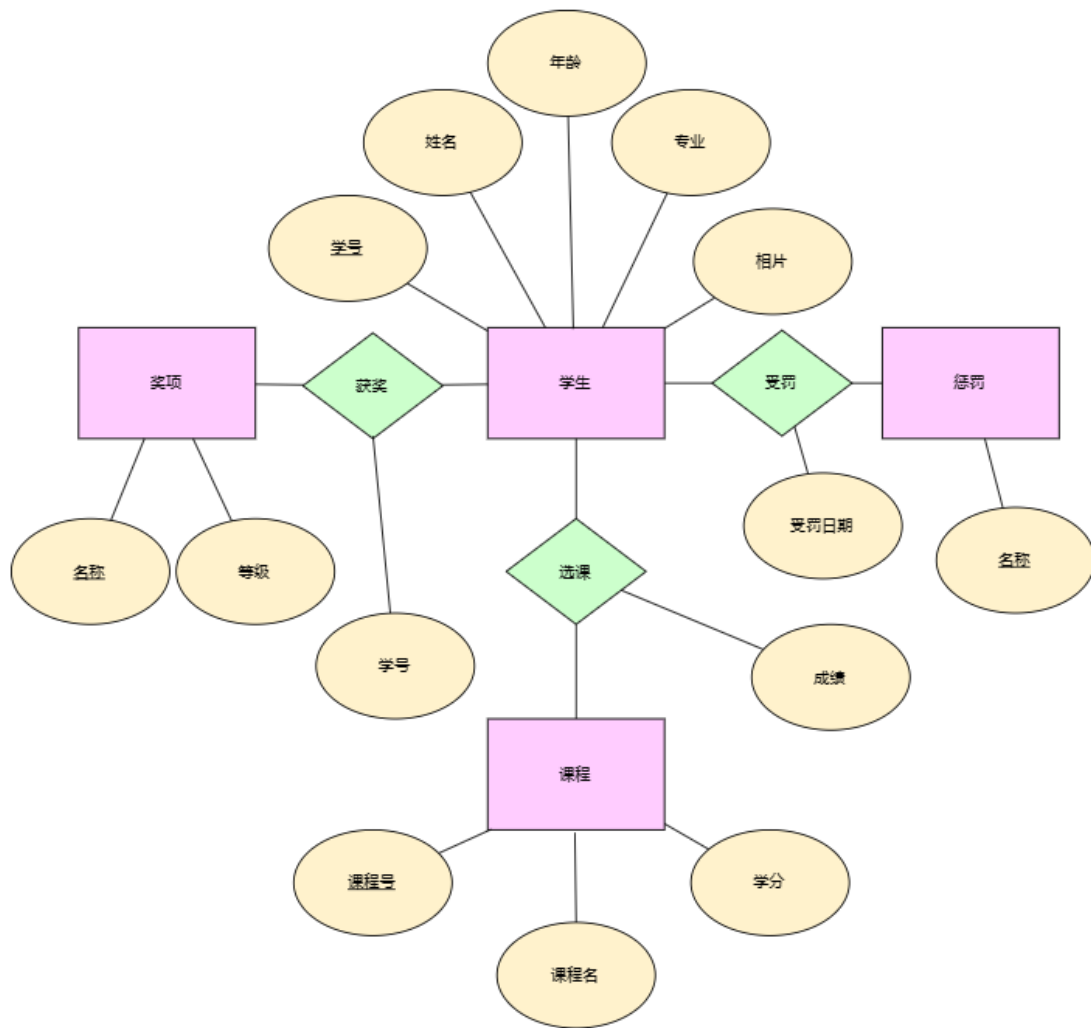
成功执行后会弹窗。



选择重置数据库时会DROP所有表格和函数然后重新生成，方便测试。

数据库设计

ER图如下：



模式分解如下：

学生表（学号，姓名，年龄，专业，相片）

课程表（课程号，课程名，学分）

奖项表（奖项名称，等级）

惩罚表（惩罚名称，受罚日期）

成绩表（学号，课程号，成绩）

获奖日期表（学号，奖项名称，日期）

受惩罚日期表（学号，惩罚名称，日期）

存储过程设计思路：

本实验我设计了两个存储过程，用于对学号，课程号两个有外键约束的主键的修改。

触发器设计思路：

本实验我设计了数个触发器，用于删除的实现，保证外键约束不被破坏。

函数设计思路：

本实验我设计了一个函数，用于计算学生当前已获得的总学分。

核心代码解析

仓库地址

[ustc2024-db_lab2/README.md at main · llz200/ustc2024-db_lab2 \(github.com\)](https://github.com/llz200/ustc2024-db_lab2)

目录

```
C:.\
|  README.md
|  实验报告.md
|
|--picture
|    11z.png
|    logo.png
|
|--src
|    classes.py          -----有关课程的后端代码
|    init_db.py          -----初始化数据库
|    main.py             -----主代码，链接数据库，创建游标对象，显示主窗口
|    prizes.py           -----有关奖项的后端代码
|    punishments.py      -----有关惩罚的后端代码
|    selects.py          -----有关查询的后端代码
|    students.py         -----有关学生的后端代码
|    test.py             -----测试用文件，主函数不调用
|    window.py           -----全部前端代码
|
|--__pycache__
|    classes.cpython-38.pyc
|    init_db.cpython-38.pyc
|    prizes.cpython-38.pyc
|    punishments.cpython-38.pyc
|    selects.cpython-38.pyc
|    students.cpython-38.pyc
|    window.cpython-38.pyc
```

建模代码

```
# ./src/init_db
sql = """
CREATE TABLE Scores (
    StudentID CHAR(10),
    ClassID CHAR(10),
    Score INT,
    PRIMARY KEY (StudentID, ClassID),
    CONSTRAINT Scores_sforeign FOREIGN KEY (StudentID) REFERENCES
Students(StudentID),
    CONSTRAINT Scores_cforeign FOREIGN KEY (ClassID) REFERENCES Classes(ClassID)
)
"""
```

```

try:
    cursor.execute(sql)
    db.commit()
except Exception as e:
    print(f"发生错误: {e}")
    db.rollback()
    return

```

选取建模部分的成绩表作为代表讲解，先用sql存储接下来要运行的sql语句，然后在try中运行，成功后就提交事务，db.commit()，失败则db.rollback()。

根据ER图，成绩、获奖时间、受惩时间都有外键约束，需要学生表、课程表、奖项表、惩罚表中的数据。

```

# ./src/init_db
sql = """
CREATE TRIGGER trg_delete_class
BEFORE DELETE ON Classes
FOR EACH ROW
BEGIN
    SELECT COUNT(*) INTO @count FROM Scores WHERE ClassID = OLD.ClassID;

    IF @count > 0 THEN
        DELETE FROM Scores WHERE ClassID = OLD.ClassID;
    END IF;
END
"""
try:
    cursor.execute(sql)
    db.commit()
except Exception as e:
    print(f"发生错误: {e}")
    db.rollback()
    return

```

触发器的实现，其他过程函数同理，格式与上面建表相同，选择在删除前的时间点触发。

后端实现

```

# ./src/students
# 查询已获得总学分
def get_total_points(db, cursor, ID):
    sql = "SELECT GetTotalPoints(%s) as TotalCredits"
    try:
        cursor.execute(sql, (ID, ))
        result = cursor.fetchone()
        return result
    except Exception as e:
        print(f"发生错误: {e}")
        return

```

选取与学生相关的后端实现作为代表讲解，格式同上，使用了%s来传递要用的ID。

前端实现

```
# ./src/window.py
class Main_Window(QWidget):
    def __init__(self, db, cursor):
        super().__init__()
        self.db = db
        self.cursor = cursor
        self.setWindowTitle('学籍管理系统')
        self.resize(1024, 864)
        self.setWindowIcon(QIcon('./picture/logo.png'))
        self.btn1 = QPushButton('查看学生名单', self)
        self.btn2 = QPushButton('查看全部课程', self)
        self.btn3 = QPushButton('按姓名查询', self)
        # 省略...
        self.btn20 = QPushButton('删除成绩', self)
        self.btn21 = QPushButton('重置数据库', self)
        self.init_ui()

    def init_ui(self):
        self.btn1.resize(128, 64)
        self.btn1.move(106, 100)    #按钮的位置
        self.btn1.clicked.connect(self.check_all_students) #使用connect绑定事件, 点击
按钮时触发
        # 省略...

        self.btn21.resize(128, 64)
        self.btn21.move(788, 700)   #按钮的位置
        self.btn21.clicked.connect(self.reset_db) #使用connect绑定事件, 点击按钮时触发

    def check_all_students(self):
        self.tablewindow = select_allTable(self.db, self.cursor, 0, '')
        self.tablewindow.show()

    # 省略...

    def check_student_info(self):
        self.newwindow = Check_StudentWindow(self.db, self.cursor)
        if self.newwindow.exec():
            newID = self.newwindow.get_entered_ID()
            newName = self.newwindow.get_entered_Name()
            newAge = self.newwindow.get_entered_Age()
            newMajor = self.newwindow.get_entered_Major()
            newPhoto = self.newwindow.get_entered_Photo()
            self.resultwindow = ResultWindow(self.db, self.cursor, newID, newName,
newAge, newMajor, newPhoto)
            self.resultwindow.show()

    def add_student(self):
        self.newwindow = New_StudentWindow(self.db, self.cursor)
        if self.newwindow.exec():
            newID = self.newwindow.get_entered_ID()
            newName = self.newwindow.get_entered_Name()
```

```

newAge = self.newwindow.get_entered_Age()
newMajor = self.newwindow.get_entered_Major()
newPhoto = self.newwindow.get_entered_Photo()
students.add_student(self.db, self.cursor, newID, newName, newAge,
newMajor, newPhoto)
widget = QWidget()
QMessageBox.information(widget, '信息', '添加成功') #触发的事件时弹出会话框

# 省略...

def change_student(self):
    getID = EnterText('修改', '学生的学号')
    if getID.exec():
        # 如果用户点击确认按钮, 则获取输入的文本
        oldID = getID.get_entered_text()
        oldinfo = selects.select_student_baseinfo(self.db, self.cursor, oldID)
        if oldinfo == None:
            widget = QWidget()
            QMessageBox.information(widget, '信息', '学生不存在')
        else:
            self.changewindow = Change_Studentwindow(self.db, self.cursor,
oldinfo)

            if self.changewindow.exec():
                newID = self.changewindow.get_entered_ID()
                newName = self.changewindow.get_entered_Name()
                newAge = self.changewindow.get_entered_Age()
                newMajor = self.changewindow.get_entered_Major()
                newPhoto = self.changewindow.get_entered_Photo()
                students.change_student(self.db, self.cursor, oldID, 1,
newName)

                students.change_student(self.db, self.cursor, oldID, 2,
newAge)

                students.change_student(self.db, self.cursor, oldID, 3,
newMajor)

                students.change_student(self.db, self.cursor, oldID, 4,
newPhoto)

                students.changeid_student(self.db, self.cursor, oldID, newID)
                widget = QWidget()
                QMessageBox.information(widget, '信息', '修改成功') #触发的事件时
弹出会话框

def delete_student(self):
    getID = EnterText('删除', '学生的学号')
    if getID.exec():
        oldID = getID.get_entered_text()
        oldinfo = selects.select_student_baseinfo(self.db, self.cursor, oldID)
        if oldinfo == None:
            widget = QWidget()
            QMessageBox.information(widget, '信息', '学生不存在')
        else:
            students.delete_student(self.db, self.cursor, oldID)
            widget = QWidget()

```

```

        QMessageBox.information(widget, '信息', '删除成功') #触发的事件时弹出会
话框

# 省略...

def reset_db(self):
    init_db.init(self.db, self.cursor)
    widget = QWidget()
    QMessageBox.information(widget, '信息', '成功重置数据库') #触发的事件时弹出对话框

```

增删改查的接口各选择了一个进行讲解。

首先主窗口使用QPushButton来实现选项，使用connect连接事件。

增添部分的接口生成一个New_StudentWindow类，然后通过exec()来判断是否结束，结束后将输入内容读入，作为参数调用students.py中的后端代码。

查询和增添部分基本一致。

修改和删除部分用oldinfo来判断输入是否合法，并且显示出来。

```

class Check_StudentWindow(QDialog):
    def __init__(self, db, cursor):
        super().__init__()
        self.db = db
        self.cursor = cursor
        self.ID = ''
        self.name = ''
        self.age = 0
        self.major = ''
        self.photo = ''
        self.setWindowTitle('输入需要查询的学生学号')
        self.resize(320, 200)
        self.setWindowIcon(QIcon('./picture/logo.png'))
        self.init_ui()

    def init_ui(self):

        # 创建 QLineEdit 对象
        self.input_box1 = QLineEdit(self)
        self.input_box1.setPlaceholderText("在此输入学号...")

        # 创建确认按钮
        self.closeButton = QPushButton('确认', self)
        self.closeButton.clicked.connect(self.get_text)

        # 设置布局
        layout = QVBoxLayout()
        layout1 = QHBoxLayout()
        layout1.addWidget(self.input_box1)
        bottomLayout = QHBoxLayout()
        bottomLayout.addWidget(self.closeButton)
        layout.addLayout(layout1)

```

```

        layout.addLayout(bottomLayout)
        self.setLayout(layout)

    def get_text(self):
        self.acceptflag = False
        if (len(self.input_box1.text()) != 0):
            self.ID = self.input_box1.text()
            info = selects.select_student_baseinfo(self.db, self.cursor, self.ID)
            if info == None:
                widget = QWidget()
                QMessageBox.information(widget, '信息', '学生不存在') #触发的事件时弹出
            else:
                self.acceptflag = True
                self.name = info[1]
                self.age = info[2]
                self.major = info[3]
                self.photo = info[4]

        if self.acceptflag:
            self.accept()

    def get_entered_ID(self):
        return self.ID

    def get_entered_Name(self):
        return self.name

    def get_entered_Age(self):
        return self.age

    def get_entered_Major(self):
        return self.major

    def get_entered_Photo(self):
        return self.photo

class ResultWindow(QWidget):
    def __init__(self, db, cursor, id, name, age, major, photo):
        super().__init__()
        self.db = db
        self.cursor = cursor
        self.id = id
        self.name = name
        self.age = age
        self.major = major
        self.photo = photo
        self.setWindowTitle('学生详细信息')
        self.resize(1024, 768)
        self.setWindowIcon(QIcon('./picture/logo.png'))
        self.init_ui()

    def init_ui(self):

```

会话框

```

scoreinfo = selects.select_student_score(self.db, self.cursor, self.id)
punishinfo = selects.select_student_prizes(self.db, self.cursor, self.id)
prizeinfo = selects.select_student_punish(self.db, self.cursor, self.id)
if (len(scoreinfo)!=0):
    total_points_info = students.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:
    image = QImage.fromData(QByteArray(self.photo))
    pixmap = QPixmap(image)
    self.label6 = QLabel(self)
    self.label6.setPixmap(pixmap)
except Exception as e:
    print(f"发生错误: {e}")
    return

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

# 创建表格
self.table2 = QTableWidget(len(prizeinfo), 2)
self.table2.setHorizontalHeaderLabels(['奖项', '等级'])
# 填充表格数据
for i in range(len(prizeinfo)):
    for j in range(2):
        item = QTableWidgetItem(str(prizeinfo[i][j]))
        self.table2.setItem(i, j, item)

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

# 创建关闭按钮
self.closeButton = QPushButton('Close', self)

```

```
self.closeButton.clicked.connect(self.close)
```

```
# 设置布局
```

```
layout = QVBoxLayout()
layout_base = QHBoxLayout()
layout_base1 = QVBoxLayout()
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_base.addWidget(self.label6)
layout.addLayout(layout_base)
layout.addWidget(self.table1)
layout.addWidget(self.table2)
layout.addWidget(self.table3)
bottomLayout = QHBoxLayout()
bottomLayout.addWidget(self.closeButton)
layout.addLayout(bottomLayout)
self.setLayout(layout)
```

前端代码选取了查询具体信息的部分进行讲解。

该部分先是一个输入学号窗口，点击确认后跳转到查询结果窗口。

使用QLineEdit来获取文本输入，QTableWidget来输出表格，QPixmap和QImage用于BLOB格式的图像输出。

最后使用layout设置布局。

主函数

```
# ./src/main.py
app = QApplication(sys.argv)
# 打开数据库连接
try:
    db = pymysql.connect(host='localhost', user='root', passwd='123456',
port=3306, autocommit=False)
    #print('连接成功! ')
except:
    print('something wrong!')

# 使用 cursor() 方法创建一个游标对象 cursor
cursor = db.cursor()

# 切换database到lab2
```

```
sql = "USE lab2"
try:
    cursor.execute(sql)
except Exception as e:
    print(f"发生错误: {e}")

window = window.Main_window(db, cursor)
window.show()
sys.exit(app.exec())
```

使用pymysql库，在主函数部分打开数据库连接，创建游标对象，切换database，打开主窗口。

实验与测试

依赖

PyQt6库、pymysql库、python3.8.8

部署

```
mysql -u root -p
```

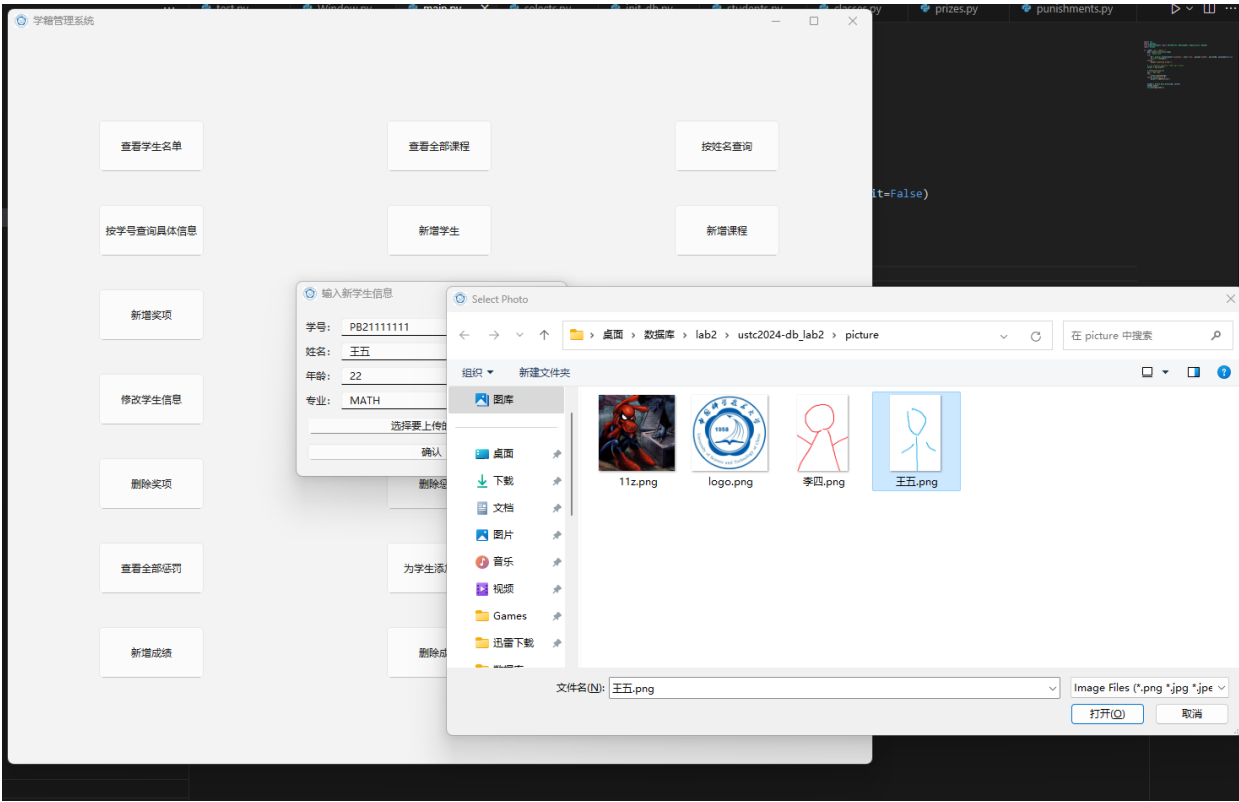
在cmd中打开数据库

```
python ./src/main.py
```

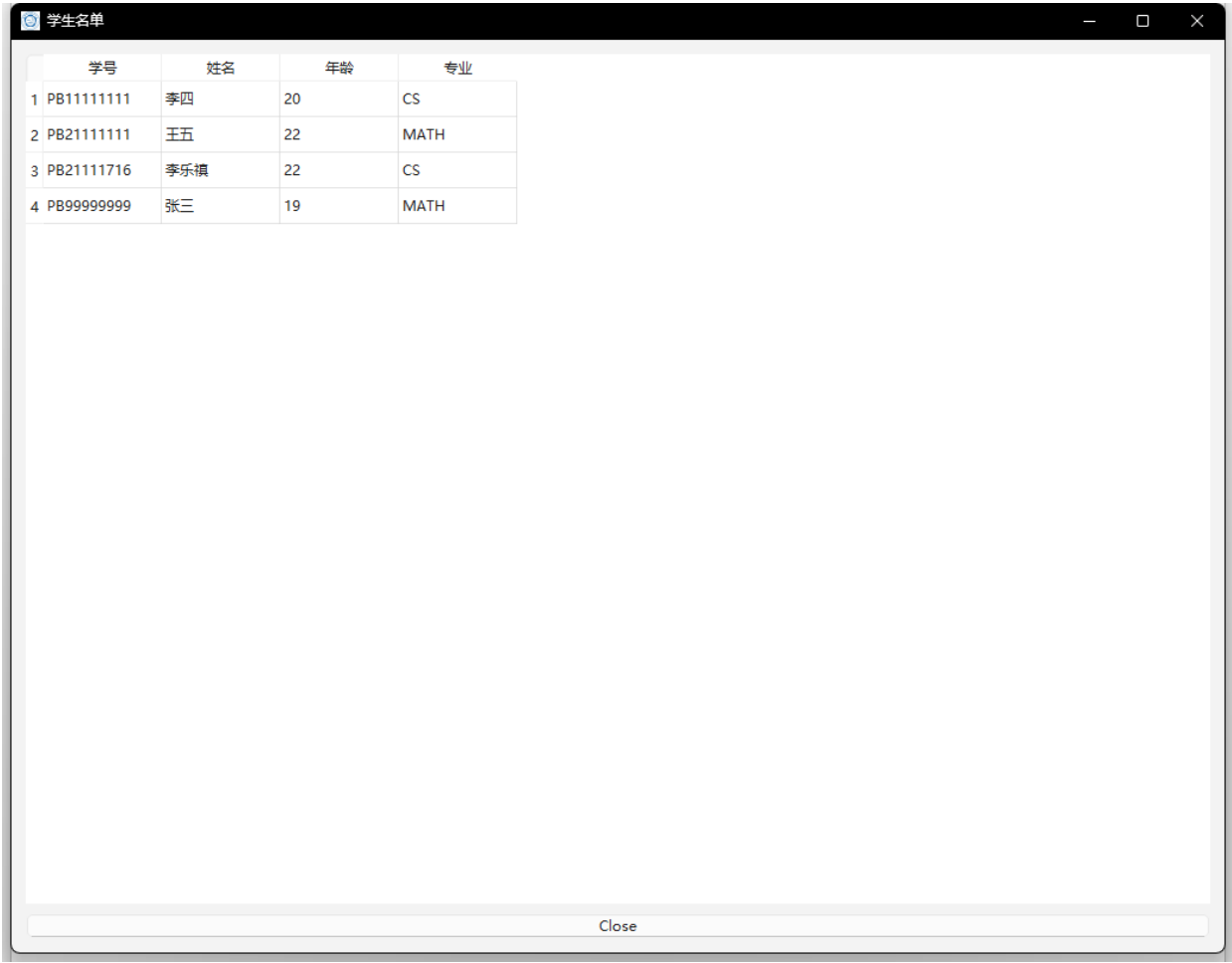
使用python语句直接运行main.py即可

实验结果

添加学生效果如下：



添加后：

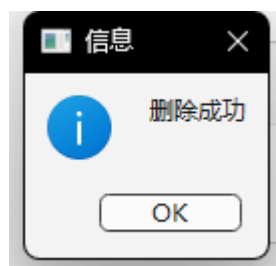


删除学生效果如下：

请输入要删除的学生的学号

PB11111111

确认



删除后：

学生名单			
学号	姓名	年龄	专业
1 PB21111111	王五	22	MATH
2 PB21111716	李乐祺	22	CS
3 PB99999999	张三	19	MATH

Close

修改学生效果如下：

请输入要修改的学生的学号

PB99999999

确认

请输入要修改的内容，不需要修改则留空

从PB99999999修改为：

在此输入新的学号...

从张三修改为：

在此输入新的姓名...

从19修改为：

20

从MATH修改为：

在此输入新的专业...

选择要上传的照片

确认

信息

i

修改成功

OK

修改后：

学生名单

	学号	姓名	年龄	专业
1	PB21111111	王五	22	MATH
2	PB21111716	李乐祺	22	CS
3	PB99999999	张三	20	MATH

Close

验证存储过程：

本次实验两个修改过程写的是修改学号，效果如下：

请输入要修改的内容，不需要修改则留空

从PB21111111修改为：

PB33333333

从王五修改为：

在此输入新的姓名...

从22修改为：

在此输入新的年龄...

从MATH修改为：

在此输入新的专业...

选择要上传的照片

确认

修改后：

学生名单

	学号	姓名	年龄	专业
1	PB21111716	李乐祺	22	CS
2	PB33333333	王五	22	MATH
3	PB99999999	张三	20	MATH

Close

验证函数：

函数为统计学生已获得的总学分，效果如下：

学生详细信息

学号: PB99999999

姓名: 张三

年龄: 20

专业: MATH

当前已获总学分: 9

课程号	成绩
1 C8181	60
2 CS1111	90

奖项	等级
----	----

惩罚

Close

课程		
课程号	课程名	学分
1 C8181	程序设计	6
2 CS1111	数据库	3

Close

验证触发器：

删除课程程序设计，相关联的成绩也会被删除，效果如下：

课程		
课程号	课程名	学分
1 CS1111	数据库	3

Close

学生详细信息

学号: PB99999999

姓名: 张三

年龄: 20

专业: MATH

当前已获总学分: 3

课程号	成绩
1 CS1111	90

奖项	等级
----	----

惩罚

Close

验证文件管理:

如上新增部分所示, 使用查询单个学生可以显示图像

学生详细信息


学号: PB21111111

姓名: 王五

年龄: 22

专业: MATH

当前已获总学分: 0



课程号	成绩

奖项	等级

惩罚

Close

参考

[PyQt6 实战: PyQt6 布局详细用法, 构建邮件发送应用程序实例, 轻松管理电子邮件 - 知乎 \(zhihu.com\)](#)

[PyMySQL documentation — PyMySQL 0.7.2 documentation](#)

[PyQt6 \(unogeeks.com\)](#)