

哈尔滨工业大学

<<数据库系统>>

实验报告二

(2024 年度秋季学期)

姓名:	刘子康
学号:	2022113416
学院:	计算学部
教师:	李东博

实验二

一、实验目的

在熟练掌握MySQL基本命令、SQL语言以及用C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

Windows 11 操作系统、MySQL 8.0.40 版本，Python 3.10，PyCharm 2022 社区版

三、实验过程及结果

此次实验开发了一个教务信息数据库系统，包含相关信息查询、学生信息修改、创建视图&索引等功能。

3.1 构建概念数据库

实体（下划线表示主键）：

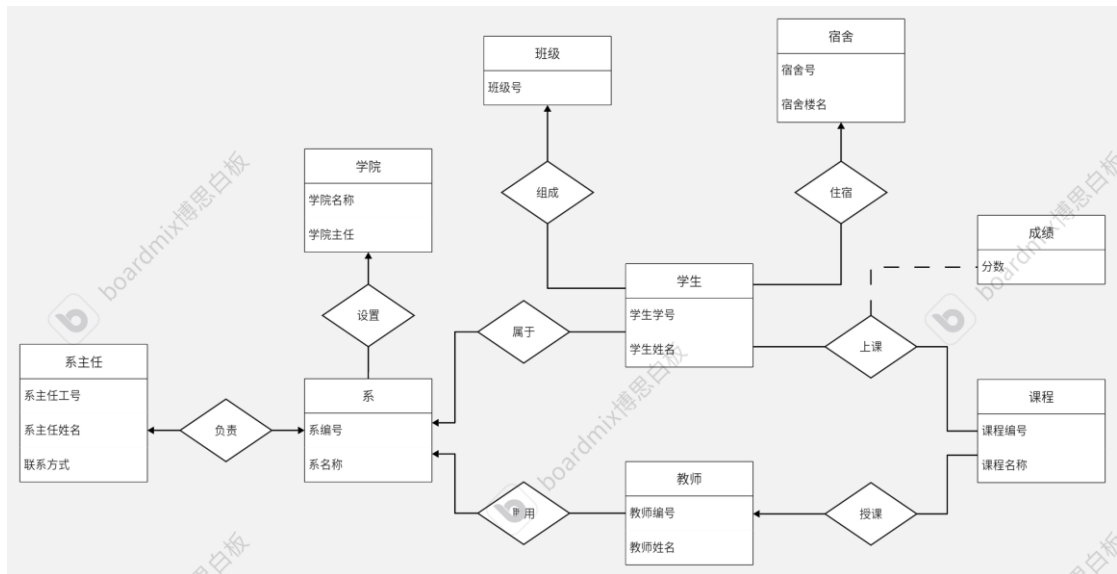
- (1) 学院（学院名，学院主任）
- (2) 系（系名，系编号，学院名）
- (3) 系主任（工号，姓名，联系方式，系编号）
- (4) 学生（学号，姓名，系编号，班级号，宿舍号）
- (5) 教师（教师工号，姓名，系编号）
- (6) 班级（班级号）
- (7) 宿舍（宿舍编号，宿舍名）
- (8) 课程（编号，课程名，授课教师工号）
- (9) 成绩（课程编号，学号，分数）

联系：

- (1) 设置：一个系由一个学院设置，一个学院可以设置多个系
- (2) 负责：一个系有且仅有一个系主任
- (3) 属于：一个学生属于一个系，一个系有多个学生
- (4) 聘用：一个系聘用多个教师，一个教师由一个系设置
- (5) 组成：一个班级由多个学生组成，一个学生属于一个班级
- (6) 住宿：一个学生住在一个宿舍，一个宿舍有多个学生
- (7) 上课：一个学生可以选多门课，一门课可有多个学生选
- (8) 授课：一个老师教授多门课，一门课只能有一个老师

3.2 绘制 E-R 图

根据以上实体和联系可绘制以下 E-R 图：



3.3 设计逻辑数据库

根据绘制的 E-R 图，对每个普通实体集构造关系 S_i ，分析实体间联系，采用构造新关系的方式处理联系，对每个联系构造关系 T_i ，得到初始关系数据库模式。经确定关系上的函数依赖集，定义关系的完整性约束，以及关系模式规范化后，得到最终关系数据库模式如下（均满足 3NF 范式）：

(1) 学院 college(coname, cdirector)

属性	含义	约束	数据类型
Coname	学院名称	非空，主键	Varchar
Cdirector	学院主任	非空	Varchar

(2) 系 department(dename, deno, coname)

属性	含义	约束	数据类型
Dename	系名	非空	Varchar
Deno	系编号	非空，主键	Char
Coname	所属学院	非空，外键	Varchar

其中 coname 参照 college.coname;

(3) 系主任 director(dino, diname, contact, deno)

属性	含义	约束	数据类型
Dino	系主任编号	非空，主键	Char
Diname	系主任名	非空	Varchar
Contact	联系方式	非空	Char
Deno	所属系	非空，外键	Char

其中 deno 参照 department.deno;

(4) 学生 student(sno, sname, deno, clno, dono)

属性	含义	约束	数据类型
Sno	学号	非空，主键	Char
Sname	姓名	非空	Varchar
Deno	所属系	非空，外键	Char
Clno	所属班级	非空，外键	Char
Dono	所属宿舍	非空，外键	Char

其中 deno 参照 department.deno，clno 参照 class.cln，dono 参照 dorm.dono；

(5) 教师 teacher(tno, tname, deno)

属性	含义	约束	数据类型
Tno	教师编号	非空，主键	Char
Tname	教师名称	非空	Varchar
Deno	所属系	非空，外键	Char

其中 deno 参照 department.deno；

(6) 班级 class(clno)

属性	含义	约束	数据类型
Clno	班级号	非空，主键	Char

(7) 宿舍 dorm(dono, doname)

属性	含义	约束	数据类型
Dono	宿舍编号	非空，主键	Char
Doname	宿舍名	非空	Varchar

(8) 课程 course(cno, cname, tno)

属性	含义	约束	数据类型
Cno	课程编号	非空，主键	Char
Cname	课程名	非空	Varchar
Tno	本课程教师	非空，外键	Char

其中 tno 参照 teacher.tno；

(9) 成绩 grade(cno, sno, gno)

属性	含义	约束	数据类型
Cno	课程编号	非空，主键，外键	Char
Sno	学生编号	非空，主键，外键	Char
Gno	成绩	非空	Int

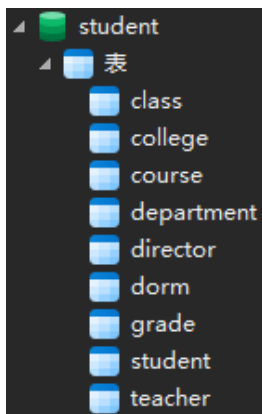
其中 cno 参照 course.cno，sno 参照 student.sno。

3.4 数据库系统实现

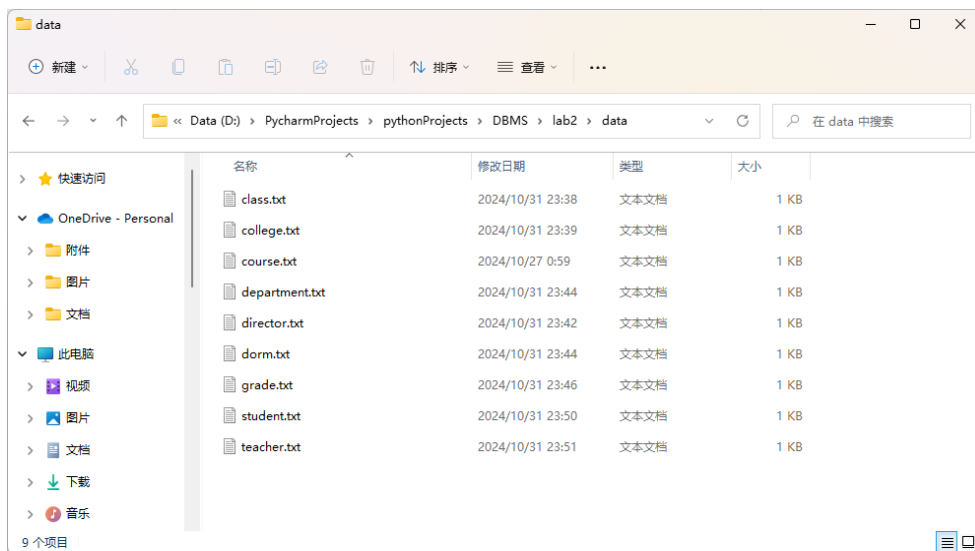
此次实验采用 MySQL+Python+PyQt5 实现数据库系统功能和可视化。

1. 在 MySQL 中创建数据库

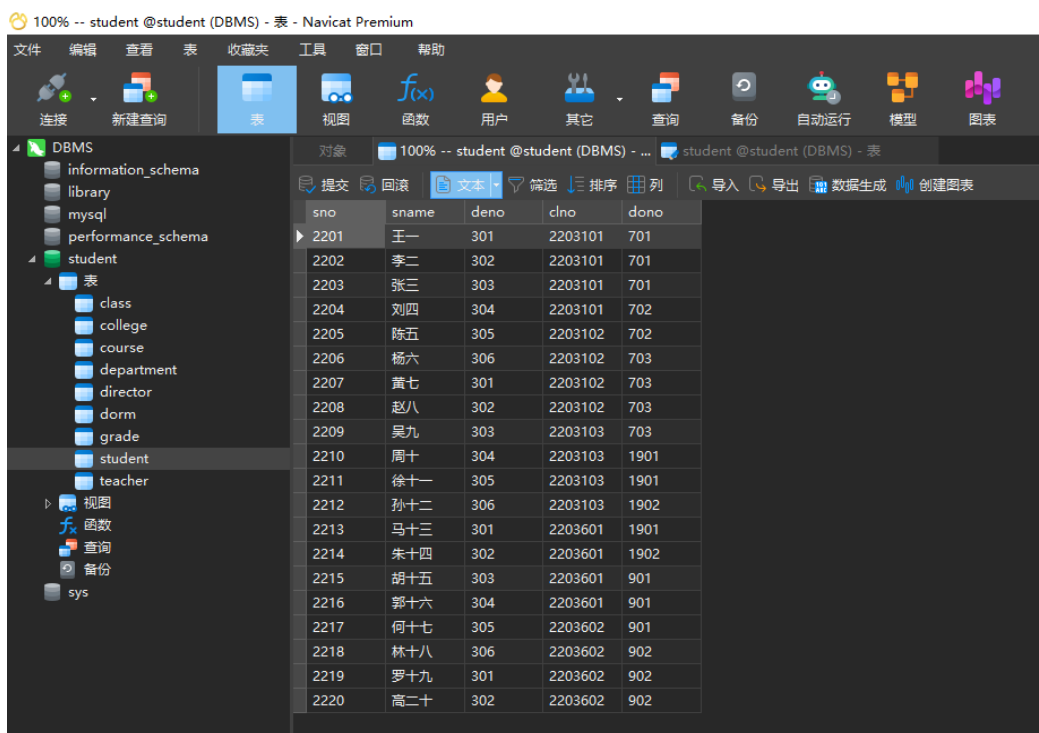
这里使用 Navicat 客户端管理 MySQL 数据库。创建 student 数据库，根据上述关系模式创建表，设置字段名、类型、长度、能否为 null、是否为主键、是否使用外键等。



然后通过 txt 文本文件导入数据：



最终结果如下图所示（部分）：



2. 可视化 GUI 实现

共有两种代码文件：

- Filename.py: 界面布局和组件设置的代码文件
- Filename_.py: 具体执行 SQL 语句操作的代码文件

主要可视化界面如下图所示：

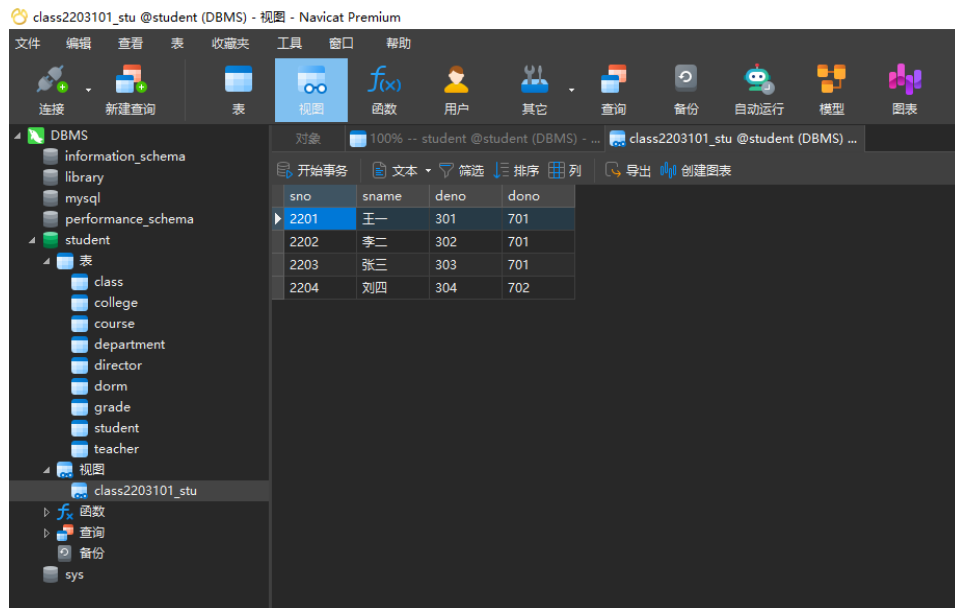


3. 检查点实现

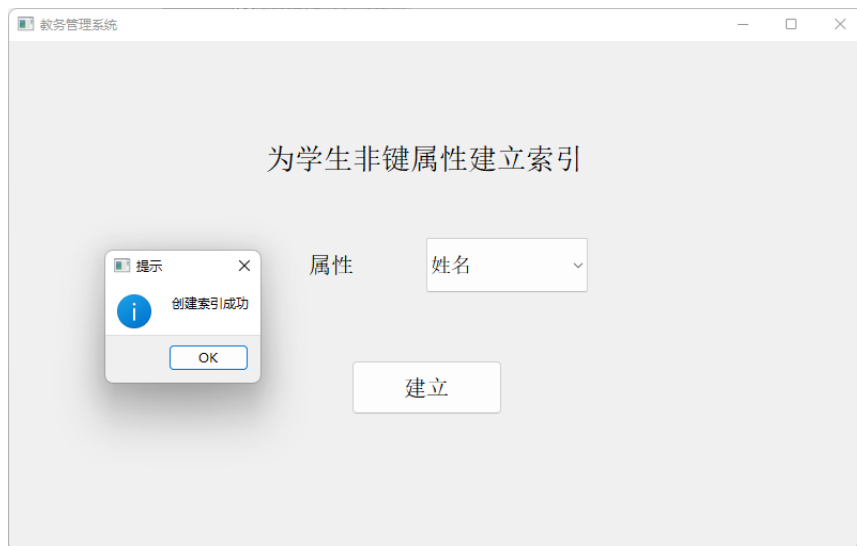
(1) 对常用查询（某班级所有学生信息）创建视图，如下图所示：



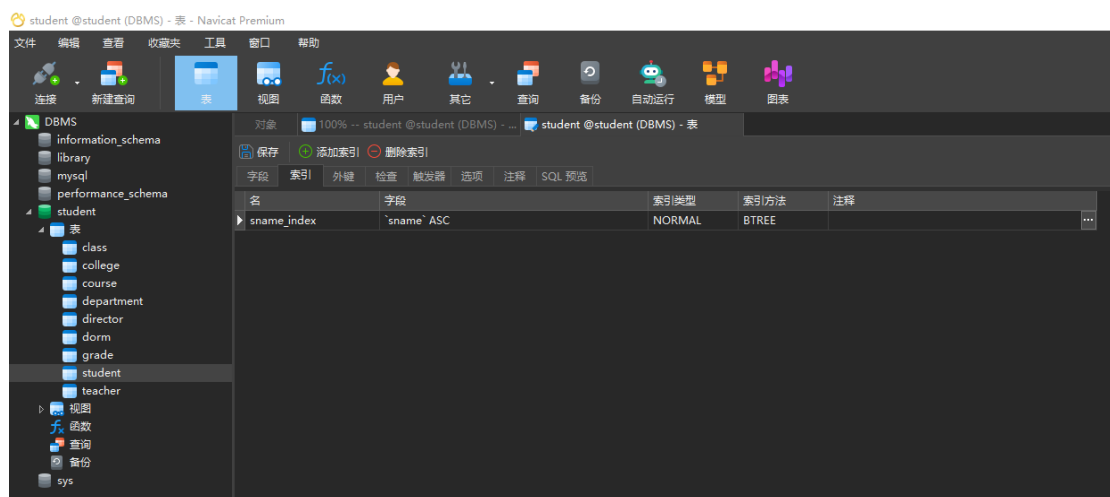
创建视图结果如下图所示：



(2) 为 student 的常用属性（非主键）建立索引



建立索引结果如下图所示：



(3) 使用 SQL 语句进行插入操作（添加学生信息）

教务管理系统

添加学生信息

学号: 2221

姓名: 潘二一

系编号: 303

班级: 2203602

宿舍号: 902

提交

提示: 插入成功

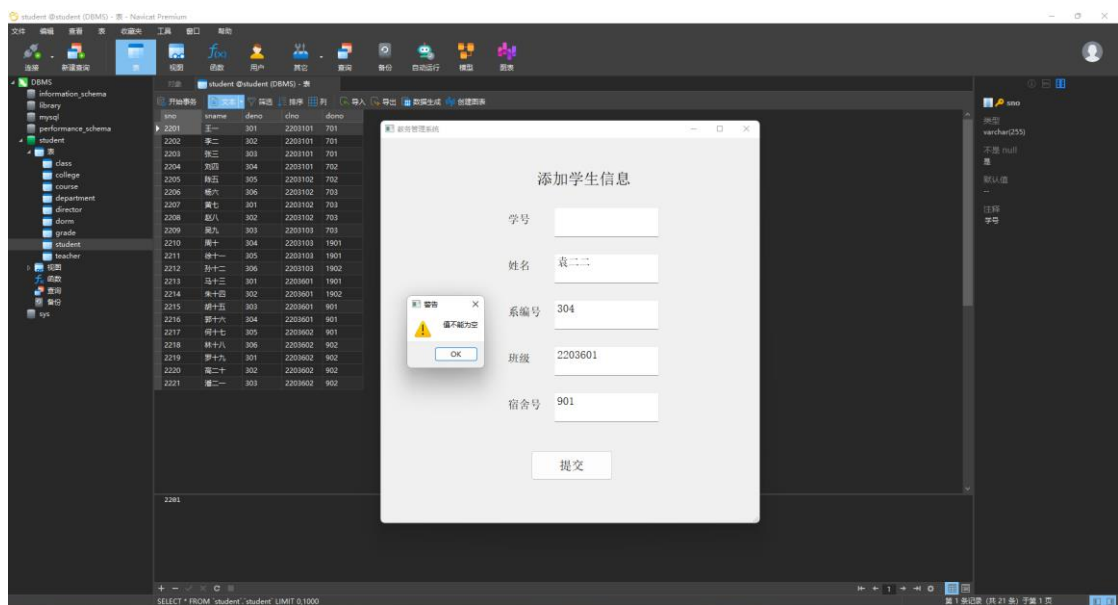
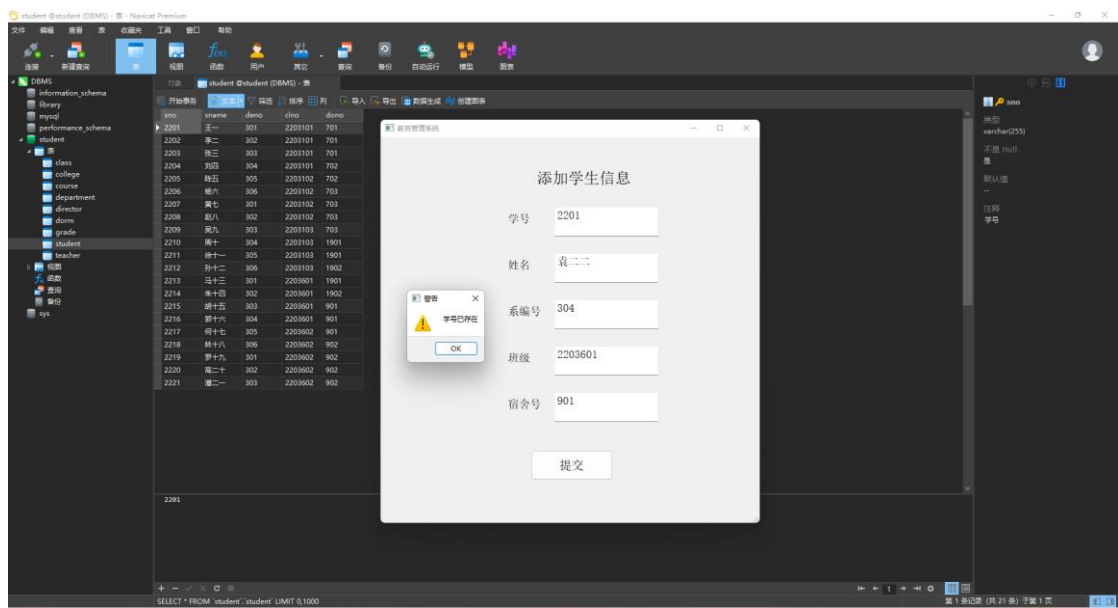
插入结果如下图所示:

student @student (DBMS) - 表 - Navicat Premium

sno	sname	deno	clno	dono
2201	王一	301	2203101	701
2202	李二	302	2203101	701
2203	张三	303	2203101	701
2204	刘四	304	2203101	702
2205	陈五	305	2203102	702
2206	杨六	306	2203102	703
2207	黄七	301	2203102	703
2208	赵八	302	2203102	703
2209	吴九	303	2203103	703
2210	周十	304	2203103	1901
2211	徐十一	305	2203103	1901
2212	孙十二	306	2203103	1902
2213	马十三	301	2203601	1901
2214	朱十四	302	2203601	1902
2215	胡十五	303	2203601	901
2216	郭十六	304	2203601	901
2217	何十七	305	2203602	901
2218	林十八	306	2203602	902
2219	罗十九	301	2203602	902
2220	高二十	302	2203602	902
2221	潘二一	303	2203602	902

当试图插入空值或重复值时，均给出警告提示（如下图所示：），体现了关系表的完整性约束。

<<数据库系统>>实验报告

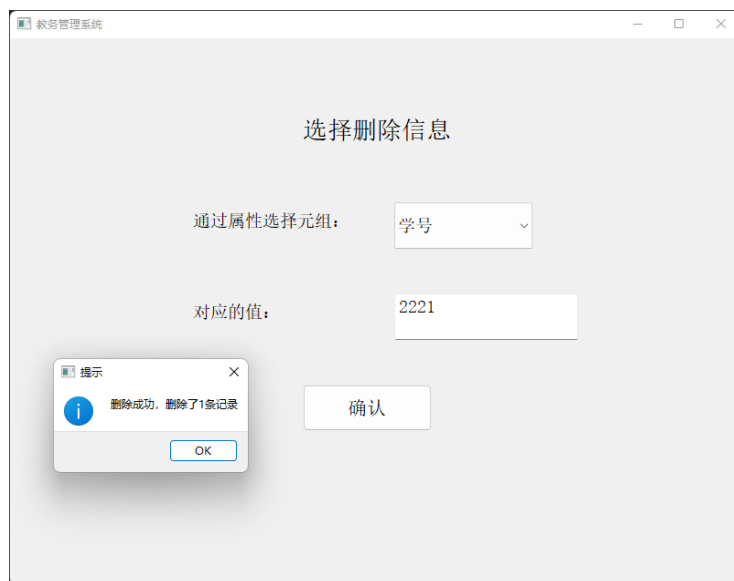


关键代码如下图所示：

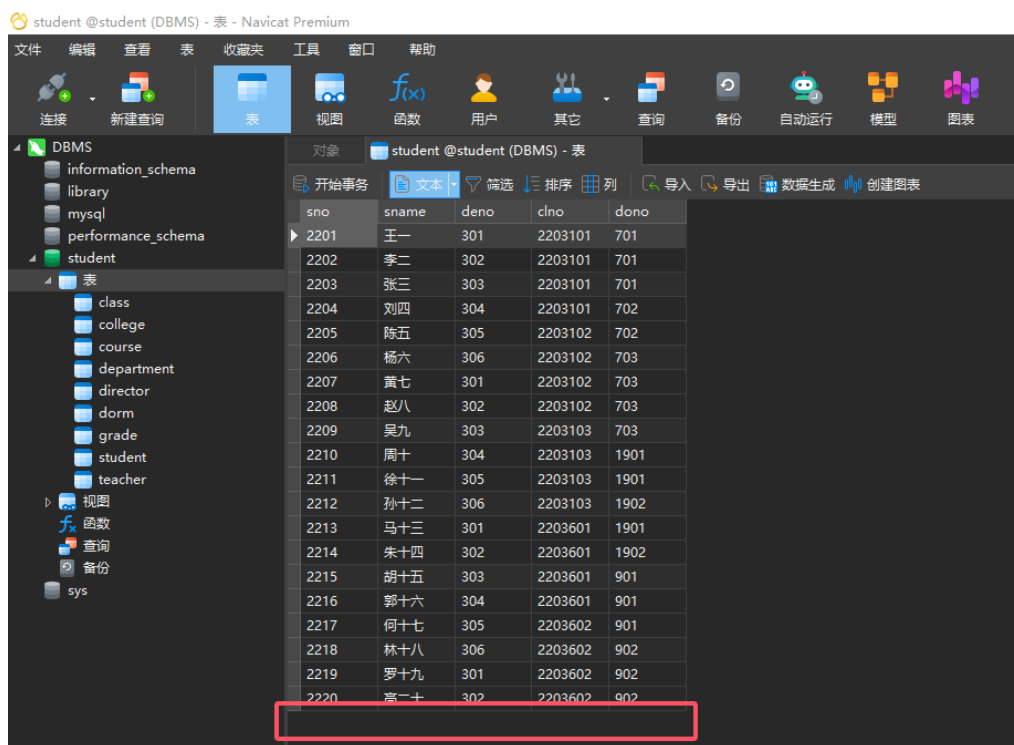
```
19         sno = self.textEdit.toPlainText()
20         sql = 'select sno from student'
21         cursor.execute(sql)
22         snos = cursor.fetchall()
23         for s in snos:
24             if s[0] == sno:
25                 QMessageBox.warning(self, '警告', '学号已存在')
26                 return

62         args = [sno, sname, deno, clno, dono]
63         if not sno or not sname or not deno or not clno or not dono:
64             QMessageBox.warning(self, '警告', '值不能为空')
65             return
66         print(args)
67         sql = 'insert into student values(%s, %s, %s, %s, %s)'
68         cnt = cursor.execute(sql, args)
69         print(cnt)
70         QMessageBox.information(self, '提示', '插入成功')
71         db.commit()
72         db.close()
```

(4) 使用 SQL 语句进行删除操作（删除学生信息）



删除结果如下图所示：



关键代码如下图所示：

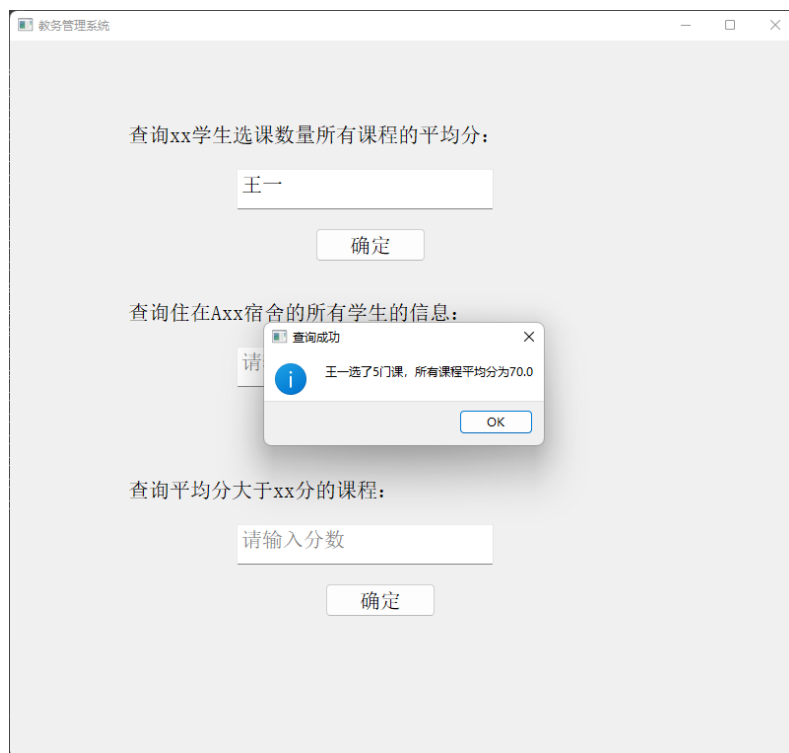
```

35         if value not in values:
36             QMessageBox.warning(self, '警告', '数据库中不存在此值')
37             return
38         sql = 'delete from student where ' + key + '=%s'
39         cnt = cursor.execute(sql, value)
40         QMessageBox.information(self, '提示', f'删除成功, 删除了{cnt}条记录')
41         db.commit()
42         db.close()

```

(5) 使用 SQL 语句进行连接查询

以查询学生选课数量以及平均分的功能为例，查询结果如下图所示：



关键代码如下图所示：

```
19 def search_1(self):
20     name = self.textEdit.toPlainText()
21     print(name)
22     if not name:
23         QMessageBox.warning(self, '警告', '请输入姓名')
24     else:
25         db = pymysql.connect(host='localhost', user='root', passwd='123456', database='student')
26         cursor = db.cursor()
27         sql = f'select avg(gno), count(*) from student natural join grade where sname=%s'
28         cursor.execute(sql, name)
29         result = cursor.fetchall()
30         if result[0][0] is None:
31             QMessageBox.warning(self, '警告', '查无此人')
32             return
33         print(result)
34         str_avg = name + f'选了{result[0][1]}门课, 所有课程平均分为{round(result[0][0], 2)}'
35         print(str_avg)
36         QMessageBox.information(self, '查询成功', str_avg)
```

(6) 使用 SQL 语句进行嵌套查询

以查询某宿舍所有学生信息的功能为例，查询结果如下图所示：



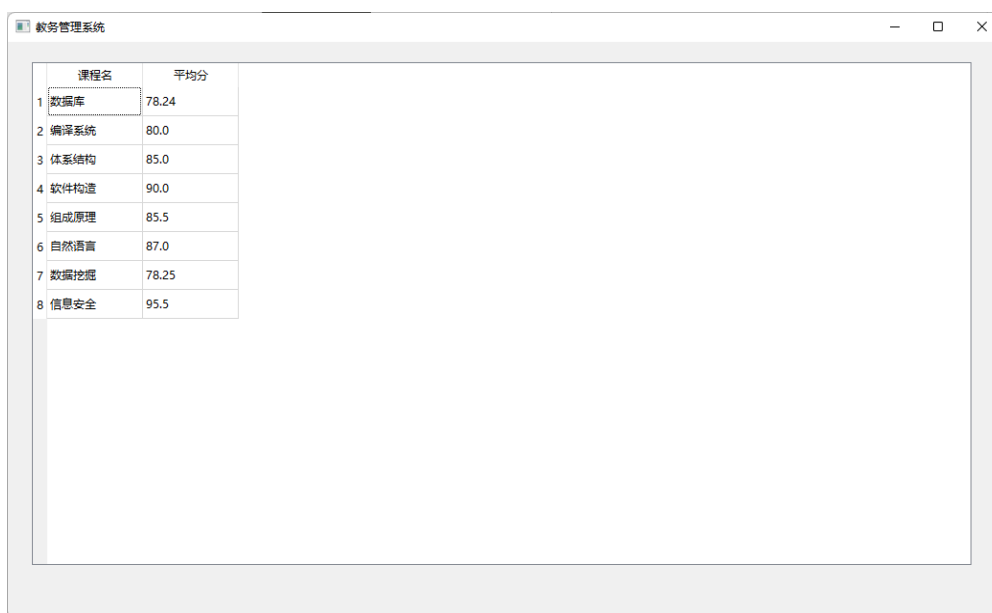
	学号	姓名	所在系名	班号	寝室号
1	2201	王一	计算机科学	2203101	701
2	2202	李二	软件工程	2203101	701
3	2203	张三	人工智能	2203101	701
4	2204	刘四	信息安全	2203101	702
5	2205	陈五	数据科学	2203102	702
6	2206	杨六	物联网	2203102	703
7	2207	黄七	计算机科学	2203102	703
8	2208	赵八	软件工程	2203102	703
9	2209	吴九	人工智能	2203103	703

关键代码如下图所示：

```
38 def search_2(self):
39     dorm = self.textEdit_2.toPlainText()
40     print(dorm)
41     if not dorm:
42         QMessageBox.warning(self, '警告', '请输入宿舍名')
43     else:
44         db = pymysql.connect(host='localhost', user='root', passwd='123456', database='student')
45         cursor = db.cursor()
46         sql = ('select sno, sname, dename, cno, dono from student natural join '
47              'department where dono in (select dono from dorm where noname=%s)')
48         cnt = cursor.execute(sql, dorm)
49         result = cursor.fetchall()
50         head = ['学号', '姓名', '所在系名', '班号', '寝室号']
51         if not result:
52             QMessageBox.warning(self, '警告', '查无此舍')
53         return
54         print(result)
55         self.table_1 = Table(head, result)
56         self.table_1.show()
```

(7) 使用 SQL 语句进行分组查询

以查询平均分大于指定分数（如 75 分）的所有课程的功能为例，查询结果如下图所示：



	课程名	平均分
1	数据库	78.24
2	编译系统	80.0
3	体系结构	85.0
4	软件构造	90.0
5	组成原理	85.5
6	自然语言	87.0
7	数据挖掘	78.25
8	信息安全	95.5

关键代码如下图所示：

```
58 def search_3(self):
59     score = self.textEdit_3.toPlainText()
60     print(score)
61     if not score:
62         QMessageBox.warning(self, '警告', '请输入阈值')
63     elif int(score) > 100 or int(score) < 0:
64         QMessageBox.warning(self, '警告', '请输入正确的阈值')
65     else:
66         db = pymysql.connect(host='localhost', user='root', passwd='123456', database='student')
67         cursor = db.cursor()
68         sql = 'select cname, avg(gno) from course natural join grade group by cno having avg(gno)>%s'
69         cnt = cursor.execute(sql, score)
70         result = cursor.fetchall()
71         head = ['课程名', '平均分']
72         self.table_2 = Table(head, result)
73         self.table_2.show()
74         print(result)
```

四、实验心得

(1) 完整地设计了一个数据库系统，从概念数据库的构建，到逻辑数据库的设计，再到具体关系数据库的实现，也熟练掌握了 E-R 图的设计方法以及 E-R 图转化为关系模式的步骤，对数据库系统有了更深层次的理解；

(2) 对于 SQL 语句的运用更加熟练，包括创建数据库，创建表，创建视图、索引以及在表上的插入、更新、查询删除等操作；

(3) 学会了使用 PyQt5 进行界面可视化，对未来的学习和项目编程有很大帮助，对于遇到的组件布局、按钮与函数绑定等问题，均通过自行查阅资料解决；

(4) 在设置表的外键时，遇到“键索引重复”、“外键约束失败”等无法添加外键的问题，通过检查表内数据类型是否一致，值是否匹配，键索引是否重复等得以解决。