哈尔滨工业大学

<<数据库系统>>

实验报告二

(2023 年度春季学期)

姓名：	刘天瑞
学号：	7203610121
学院：	计算学部
教师：	李东博

实验一

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 关系数据库管理系统、MinGW 编译器、GUI 编程语言 Python 3.9.7 PyQt5 库

三、实验过程及结果

一共有学院、系、系主任、学生、教师、班级、住宿、课程 8 个实体集，还有成绩这一个联系集。其中系主任和系是一对一的关系，学生和课程是多对多的关系，剩余皆为一对多的关系，如一门课只由一位老师教，一个老师可以教多门课。

实体：主码加粗标红表示：

（1）学院：**学院名**、学院领导姓名

（2）系：**系名**、系主任工号、所属学院名

（3）系主任：**系主任工号**、系主任姓名、手机号

（4）学生：**学号**、学生姓名、所在系名、所在班号、所在宿舍号

（5）教师：**工号**、教师姓名、所在系名

（6）宿舍：**宿舍号**、所在公寓

（7）班级：**班号**

（8）课程：**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）一个学院可以设置多个系，但是每个系都只能属于一个学院；

（2）一个系主任只能负责一个系，而且一个系里也只能有一个系主任；

（3）一个系里有多个学生，每个学生都只能属于一个系；

（4）一个系可以聘用多个教师，每个教师只能被一个系聘用；

（5）一个学生只能住宿在一个宿舍，每个宿舍可以住宿多个学生；

（6）一个班级由多个学生组成，每个学生都只能属于一个班级；

（7）每个学生可以选择学习多门课程，每门课程也可以由多个学生学习；

（8）每位教师可以授课多门课程，每门课程也可以由多位教师来教授。

已设计好的关系表：

学生和课程是多对多关系，所以转换成逻辑数据库的时候添加了联系集成绩。其余的一对多关系，均在 n 的关系表中加入了 1 的主码，如在学生的表中添加了班级的主码班号，宿舍的主码宿舍号。具体如下：

学院：college(name, dname) // dname 为学院领导姓名

系：department(name, did, cname) // did 为系主任工号，cname 为学院名

系主任：director(id, name, phone)

学生：student(sid, sname, de_name, class, dorm) // de_name 为所在系名

教师：teacher(tid, tname, de_name) // de_name 为所在系名

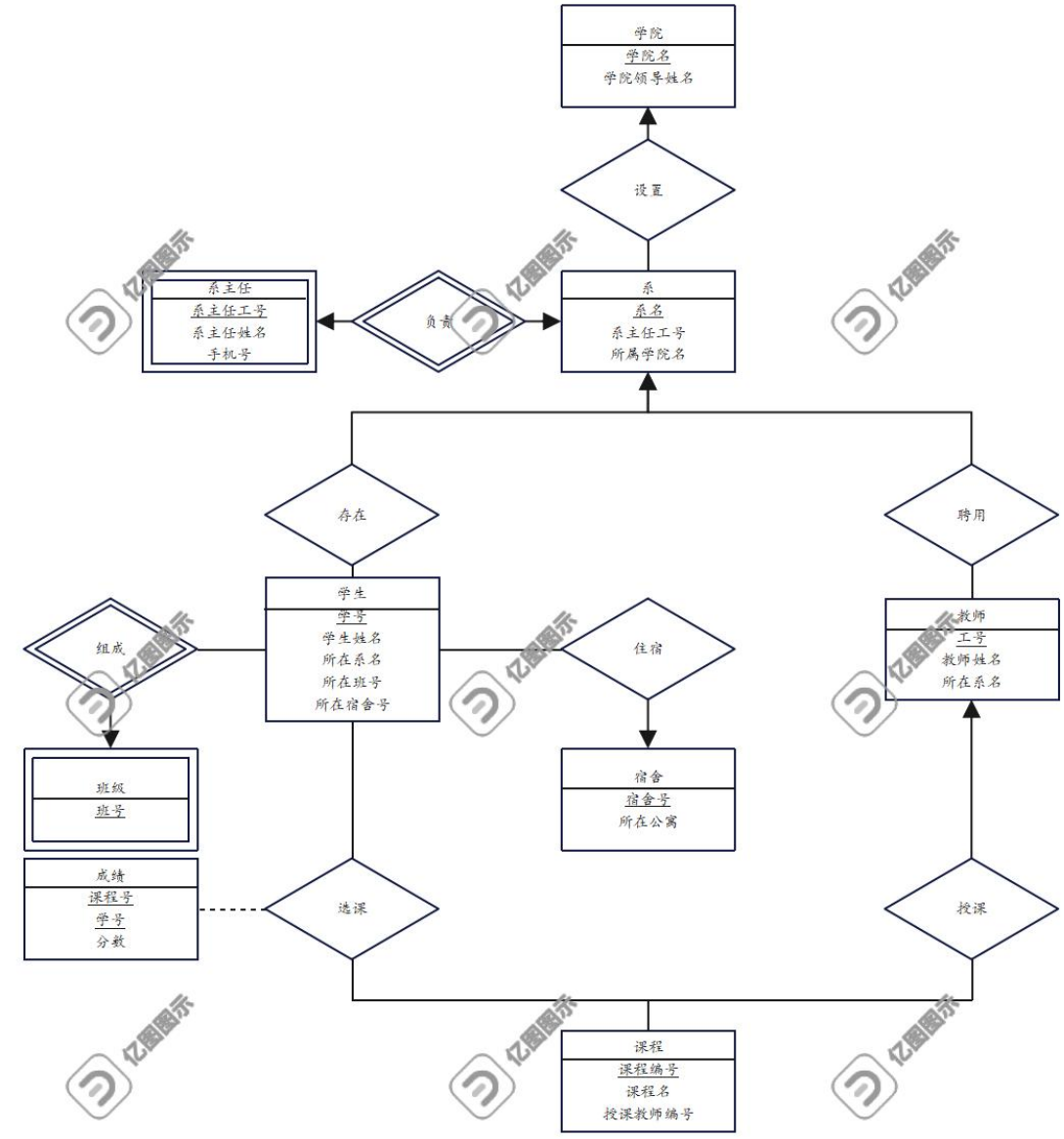
宿舍：dorm(id, place) // id 为宿舍号，place 为所在公寓

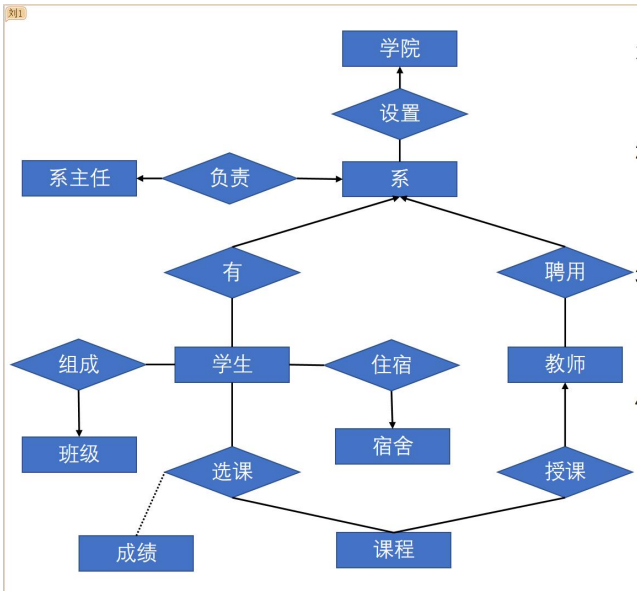
班级：class(id)

课程：course(cid, cname, tid) // tid 为授课教师工号

成绩：grade(cid, sid, score)

已绘制好的 E-R 图如下图所示（存在弱实体：系和班级；即也分别存在弱联系：负责和组成；虚线表示描述性属性，用于联系集；长方形表示主体；菱形表示关系）：





1. College（学院）
    1. Name（学院名）
    2. Dname(学院领导姓名)
2. Department（系）
    1. Name（系名）
    2. Did(系主任工号)
    3. Cname(所属学院名)
3. Director（系主任）
    1. Id（系主任工号）
    2. Name（系主任姓名）
    3. phone（手机号）
4. Student（学生）
    1. Sid（学号）
    2. Sname（学生姓名）
    3. De_name(所在系名)
    4. Class(所在班号)
    5. Dorm(所在宿舍号)
5. Teacher（教师）
    1. Tid（工号）
    2. Tname（教师姓名）
    3. De_name（所在系名）
6. Dorm（宿舍）
    1. Id（宿舍号）
    2. place（所在公寓）
7. Class（班级）
    1. Id（班号）
8. Course（课程）
    1. cid（课程编号）
    2. Cname（课程名）
    3. Tid(授课教师工号)
9. Grade（成绩）
    1. cid（课程号）
    2. Sid（学号）
    3. score（分数）

设计逻辑数据库：

确定关系上的函数依赖集；关系模式规范化（总共 16 个关系满足 BCNF）；并且无需进行关系模式优化。

关系的完整性约束：

主键约束如以上关系表以及 E-R 图主体属性的划线部分已经给出。各表中主键如下图截图命令行所示：

```
mysql> describe student;
+---------+--------------+------+-----+---------+-------+
| Field   | Type         | Null | Key | Default | Extra |
+---------+--------------+------+-----+---------+-------+
| sid     | int          | NO   | PRI | NULL    |       |
| sname   | varchar(255) | YES  | MUL | NULL    |       |
| de_name | varchar(255) | YES  | MUL | NULL    |       |
| class   | varchar(255) | YES  | MUL | NULL    |       |
| dorm    | int          | YES  | MUL | NULL    |       |
+---------+--------------+------+-----+---------+-------+
```

```
mysql> describe college;
+-------+--------------+------+-----+---------+-------+
| Field | Type         | Null | Key | Default | Extra |
+-------+--------------+------+-----+---------+-------+
| name  | varchar(255) | NO   | PRI | NULL    |       |
| dname | varchar(255) | YES  |     | NULL    |       |
+-------+--------------+------+-----+---------+-------+
```

```
mysql> describe department;
+-------+--------------+------+-----+---------+-------+
| Field | Type         | Null | Key | Default | Extra |
+-------+--------------+------+-----+---------+-------+
| name  | varchar(255) | NO   | PRI | NULL    |       |
| did   | int          | YES  | MUL | NULL    |       |
| cname | varchar(255) | YES  | MUL | NULL    |       |
+-------+--------------+------+-----+---------+-------+
```

```
mysql> describe director;
+-------+--------------+------+-----+---------+-------+
| Field | Type         | Null | Key | Default | Extra |
+-------+--------------+------+-----+---------+-------+
| id    | int          | NO   | PRI | NULL    |       |
| name  | varchar(255) | YES  |     | NULL    |       |
| phone | varchar(255) | YES  |     | NULL    |       |
+-------+--------------+------+-----+---------+-------+
```

```
mysql> describe teacher;
+---------+--------------+------+-----+---------+-------+
| Field   | Type         | Null | Key | Default | Extra |
+---------+--------------+------+-----+---------+-------+
| tid     | int          | NO   | PRI | NULL    |       |
| tname   | varchar(255) | YES  |     | NULL    |       |
| de_name | varchar(255) | YES  | MUL | NULL    |       |
+---------+--------------+------+-----+---------+-------+
```

```
mysql> describe dorm;
+-------+--------------+------+-----+---------+-------+
| Field | Type         | Null | Key | Default | Extra |
+-------+--------------+------+-----+---------+-------+
| id    | int          | NO   | PRI | NULL    |       |
| place | varchar(255) | YES  |     | NULL    |       |
+-------+--------------+------+-----+---------+-------+
```

```
mysql> describe class;
+-------+--------------+------+-----+---------+-------+
| Field | Type         | Null | Key | Default | Extra |
+-------+--------------+------+-----+---------+-------+
| id    | varchar(255) | NO   | PRI | NULL    |       |
+-------+--------------+------+-----+---------+-------+
```

```
mysql> describe course;
+-------+--------------+------+-----+---------+-------+
| Field | Type         | Null | Key | Default | Extra |
+-------+--------------+------+-----+---------+-------+
| cid   | int          | NO   | PRI | NULL    |       |
| cname | varchar(255) | YES  | MUL | NULL    |       |
| tid   | int          | YES  | MUL | NULL    |       |
+-------+--------------+------+-----+---------+-------+
```

```
mysql> describe grade;
+-------+-------+------+-----+---------+-------+
| Field | Type  | Null | Key | Default | Extra |
+-------+-------+------+-----+---------+-------+
| cid   | int   | NO   | PRI | NULL    |       |
| sid   | int   | NO   | PRI | NULL    |       |
| score | float | YES  |     | NULL    |       |
+-------+-------+------+-----+---------+-------+
```

外键约束：系 department 中 cname，did 是外键。学生 student 中 de_name，class，dorm 是外键。教师 teacher 中 de_name 是外键。课程 course 中 tid 为外键。

具体每个表的外键约束内容如下图所示：

对象 course @lab2 (mysql) - 表

保存 添加外键 删除外键

字段 索引 外键 检查 触发器 选项 注释 SQL 预览

名	字段	被引用的模式	被引用的表（父）	被引用的字段	删除时	更新时
tid	tid	lab2	teacher	tid	RESTRICT	RESTRICT

对象 department @lab2 (mysql) - 表

保存 添加外键 删除外键

字段 索引 外键 检查 触发器 选项 注释 SQL 预览

名	字段	被引用的模式	被引用的表（父）	被引用的字段	删除时	更新时
cname	cname	lab2	college	name	RESTRICT	RESTRICT
did	did	lab2	director	id	RESTRICT	RESTRICT

对象 student @lab2 (mysql) - 表

保存 添加外键 删除外键

字段 索引 外键 检查 触发器 选项 注释 SQL 预览

名	字段	被引用的模式	被引用的表（父）	被引用的字段	删除时	更新时
class	class	lab2	class	id	RESTRICT	RESTRICT
dname	de_name	lab2	teacher	de_name	RESTRICT	RESTRICT
dorm	dorm	lab2	dorm	id	RESTRICT	RESTRICT

对象 teacher @lab2 (mysql) - 表

保存 添加外键 删除外键

字段 索引 外键 检查 触发器 选项 注释 SQL 预览

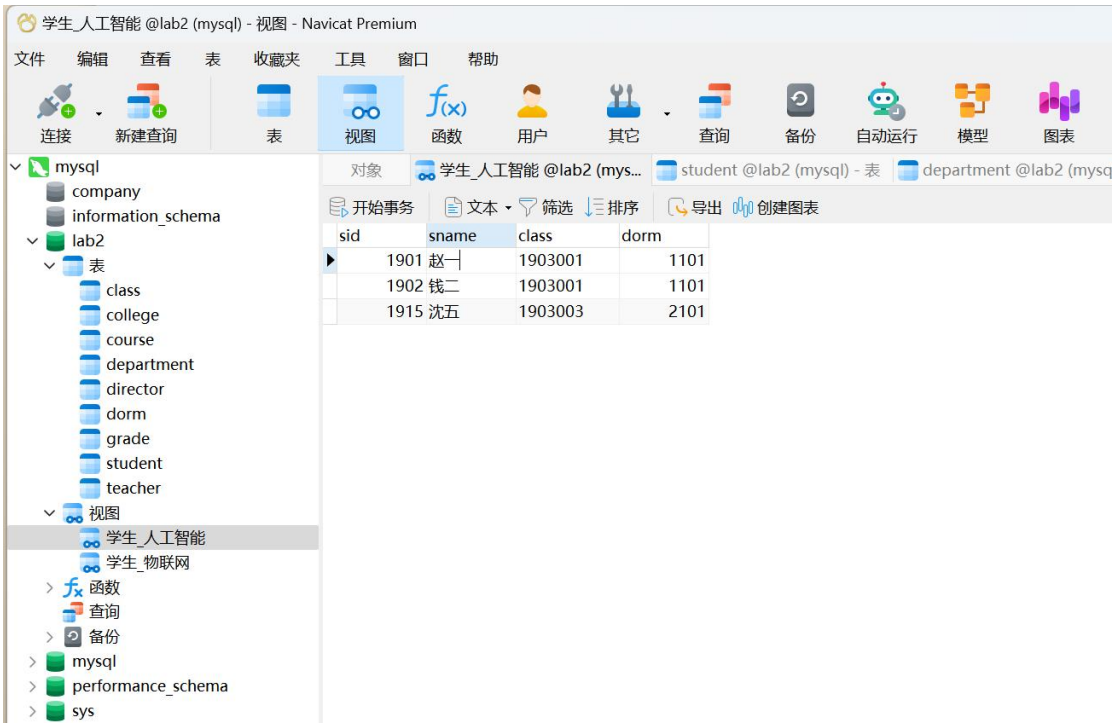
名	字段	被引用的模式	被引用的表（父）	被引用的字段	删除时	更新时
de_name	de_name	lab2	department	name	RESTRICT	RESTRICT

空值约束全为非空属性（Not Null），具体约束方法会在后面程序中会体现。

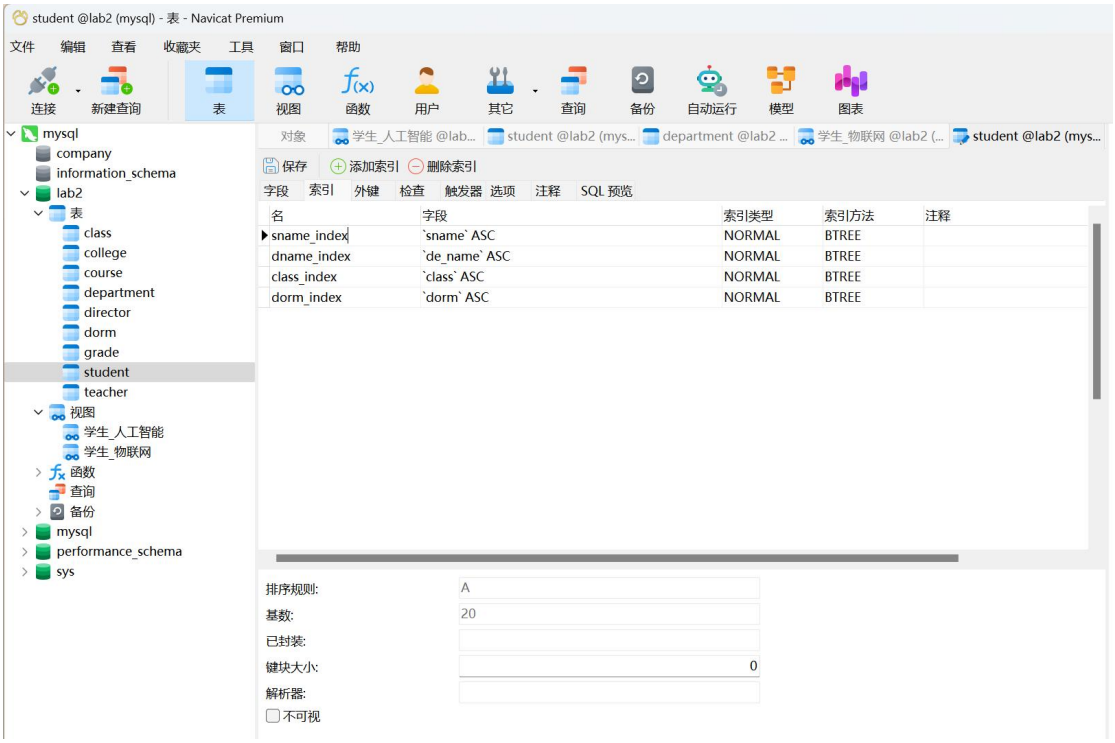
子模式定义略。

常用查询的视图和常用属性的索引：

我为同一院系的同学建立了视图，建立的方法在后续程序中会体现。如人工智能院系的学生：



对于非主键的属性索引，我就以学生为例，为除了主键学号的其他属性都建立了索引，如下图所示：

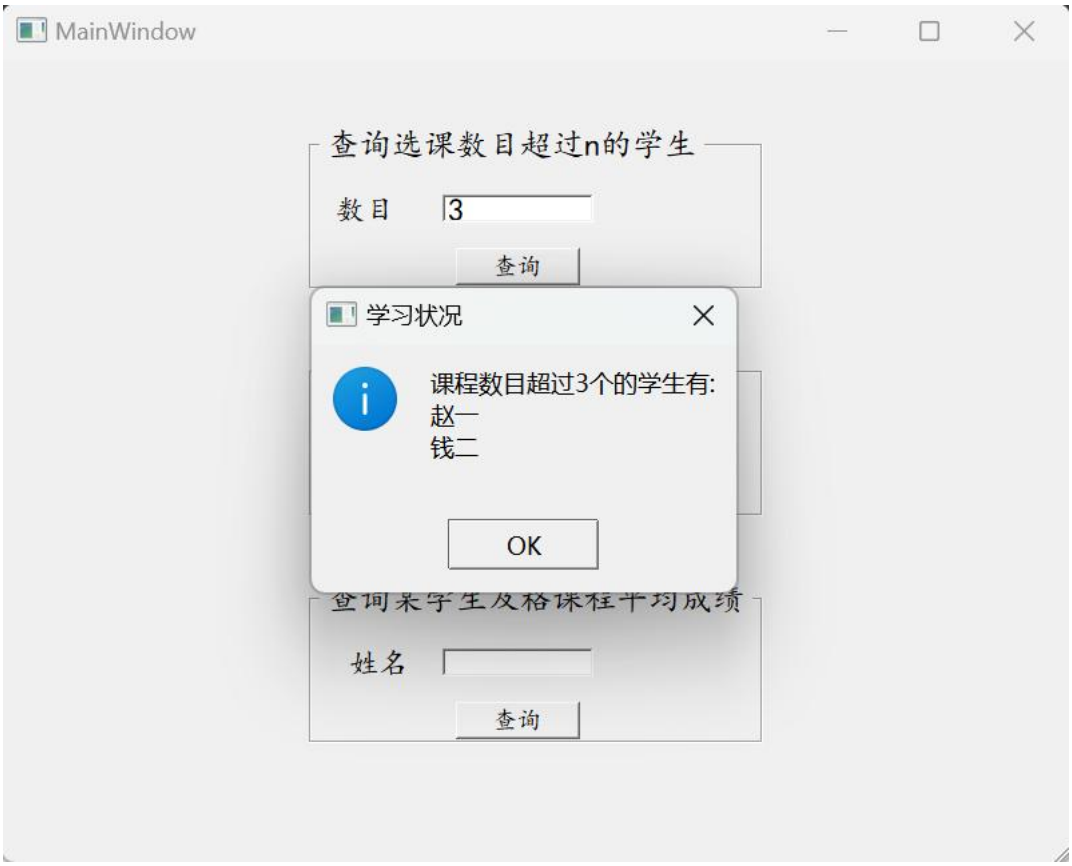


```
mysql> show index from student;
+---------+------------+---------------+--------------+-------------+-----------+-------------+----------+--------+------+------
------+---------+---------------+---------+------------+
| Table   | Non_unique | Key_name      | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index
_type | Comment | Index_comment | Visible | Expression |
+---------+------------+---------------+--------------+-------------+-----------+-------------+----------+--------+------+------
------+---------+---------------+---------+------------+
| student |          0 | PRIMARY       |            1 | sid         | A         |           0 | NULL     | NULL   |      | BTREE
     |         |               | YES     | NULL       |
| student |          1 | sname_index   |            1 | sname       | A         |          20 | NULL     | NULL   | YES  | BTREE
     |         |               | YES     | NULL       |
| student |          1 | dname         |            1 | de_name     | A         |           7 | NULL     | NULL   | YES  | BTREE
     |         |               | YES     | NULL       |
| student |          1 | class         |            1 | class       | A         |           4 | NULL     | NULL   | YES  | BTREE
     |         |               | YES     | NULL       |
| student |          1 | dorm          |            1 | dorm        | A         |           7 | NULL     | NULL   | YES  | BTREE
     |         |               | YES     | NULL       |
| student |          1 | de_name_index |            1 | de_name     | A         |           7 | NULL     | NULL   | YES  | BTREE
     |         |               | YES     | NULL       |
+---------+------------+---------------+--------------+-------------+-----------+-------------+----------+--------+------+------
------+---------+---------------+---------+------------+
6 rows in set (0.03 sec)
```

查询操作：

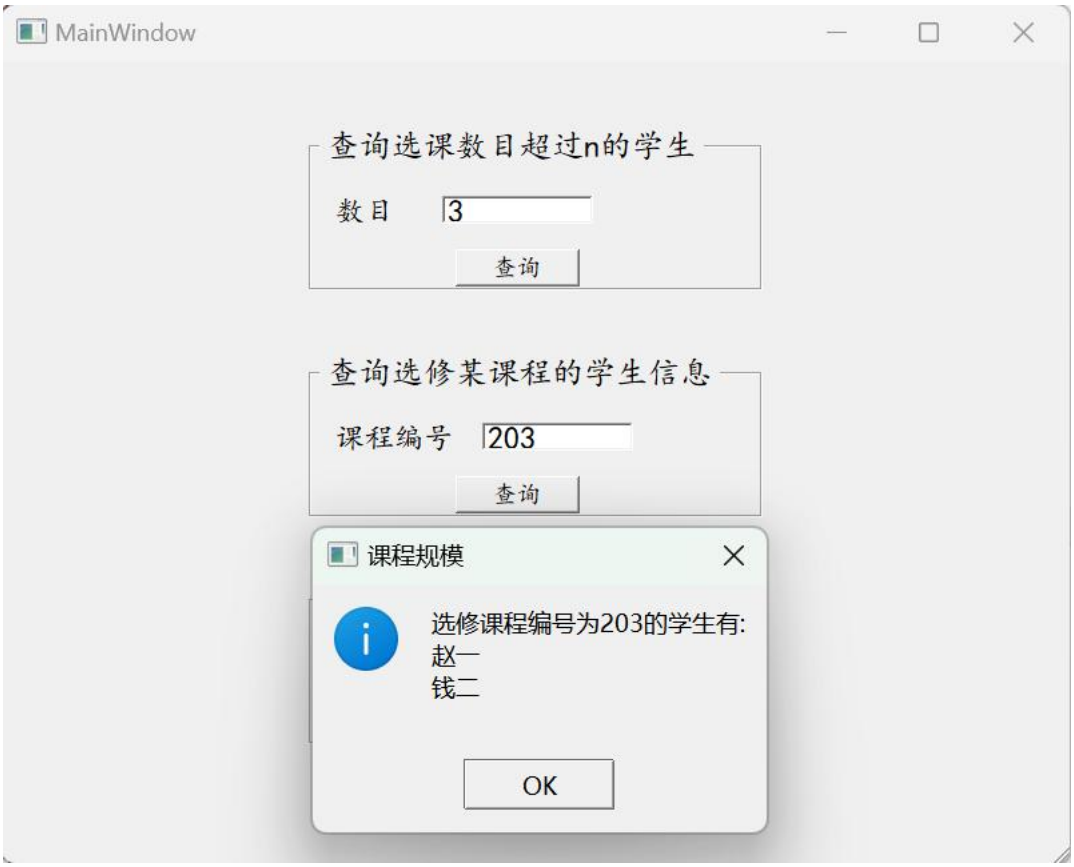
如果查询选课数目超过 n 的学生姓名。首先进行了输入 n 的检查，以及是否为空值，错误值并且给予提示。在具体查询中的 sql 语句使用了关系的连接和分组操作，并且使用 having 语句进行筛选，下图中的 query。部分运行的关键代码如下图所示：

```
128     def query_stu(self):
129         text = self.lineEdit.text()
130         if not text:
131             QMessageBox.warning(self, '警告！', '请输入学生数目(禁止为空)！')
132         elif int(text)<0:
133             QMessageBox.warning(self, '警告！', '请输入正确的学生数目！')
134         else:
135             con = pymysql.connect(host='localhost', port=3306, user='root', password='691179', charset='utf8',
136                                   database='lab2')  # 连接mysql数据库
137             cur = con.cursor()  # 执行sql语句的游标
138             # 连接查询，体现分组，having
139             query = 'select sname from student natural join grade group by grade.sid having count(*) > %s'
140             cur.execute(query, [text])
141             curr = cur.fetchall()
142             if len(curr)>0:
143                 msg = '课程数目超过{num}个的学生有:\n'.format(num=text)
144                 for item in curr:
145                     msg += item[0]+'\n'
146             else:
147                 msg = '数目过大，无结果！'
148             QMessageBox.information(self, '学习状况', msg)
149             con.close()
150             cur.close()
```



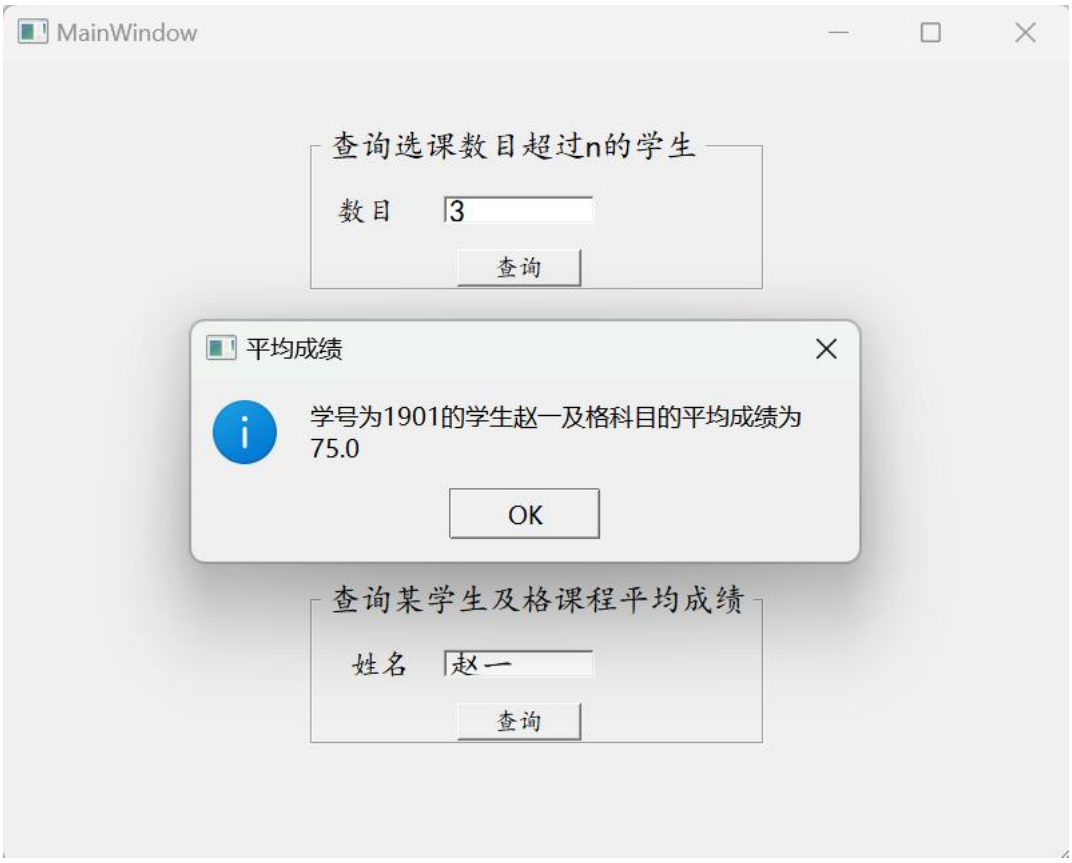
如果查询选修了某特定课程号的课程的学生姓名。这里面的 sql 语句使用到了嵌套查询，如下图所示中的 query 部分运行关键代码：

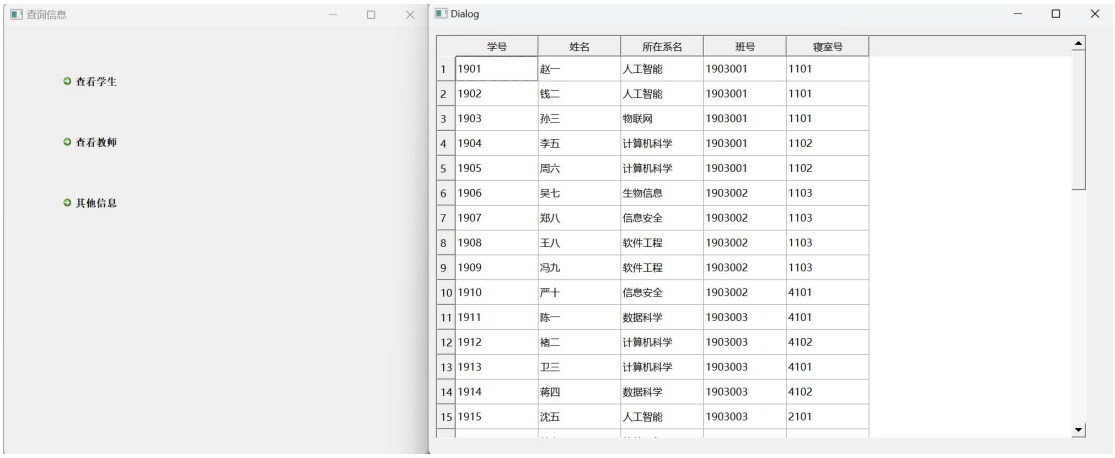
```
153    def query_stu_sc(self):
154        c_num = self.lineEdit_2.text()
155
156        if not c_num:
157            QMessageBox.warning(self, '警告！', '请输入课程编号（禁止为空）！')
158        elif len(c_num) != 3:
159            QMessageBox.warning(self, '警告！', '课程号错误，应为3位数字！')
160        else:
161            con = pymysql.connect(host='localhost', port=3306, user='root', password='691179', charset='utf8',
162                                  database='lab2')  # 连接数据库
163            cur = con.cursor()  # 执行sql语句的游标
164            query = 'select sname from student where sid in (select sid from grade where cid = %s)' # 嵌套查询
165            cur.execute(query, [c_num])
166            curr = cur.fetchall()
167            if len(curr) > 0:
168                msg = '选修课程编号为{num}的学生有:\n'.format(num=c_num)
169                for item in curr:
170                    msg += item[0] + '\n'
171            else:
172                msg = '编号错误，无结果！'
173            QMessageBox.information(self, '课程规模', msg)
174            con.close()
175            cur.close()
```



除此之外，我还提供了其他查询，例如查询所有学生信息，所有教师信息以及某位学生的所有及格课程的平均分。query 部分运行关键代码如下图所示：

```
178    def avg_score(self):
179        name = self.lineEdit_3.text()
180        if not name:
181            QMessageBox.warning(self, '警告！', '请输入学生姓名（禁止为空）！')
182        else:
183            con = pymysql.connect(host='localhost', port=3306, user='root', password='691179', charset='utf8',
184                                  database='lab2')  # 连接数据库
185            cur = con.cursor()  # 执行sql语句的游标
186            res=[]
187            query = 'select sid, avg(score) from grade where sid in (select sid from student where sname = %s)  and score >= 60 group by sid'
188            cur.execute(query, [name])
189            for item in cur.fetchall():
190                res.append('学号为'+str(item[0])+'的学生{name}及格科目的平均成绩为'.format(name=name) + '\t' + str(item[1]))
191                print(item[0])
192            QMessageBox.information(self, '平均成绩', '\n'.join(res) if len(res) != 0 else '输入错误，无结果！')
```


MainWindow
查询选课数目超过n的学生
数目
3
查询
平均成绩
学号为1901的学生赵一及格科目的平均成绩为75.0
OK
查询某学生及格课程平均成绩
姓名
赵一
查询

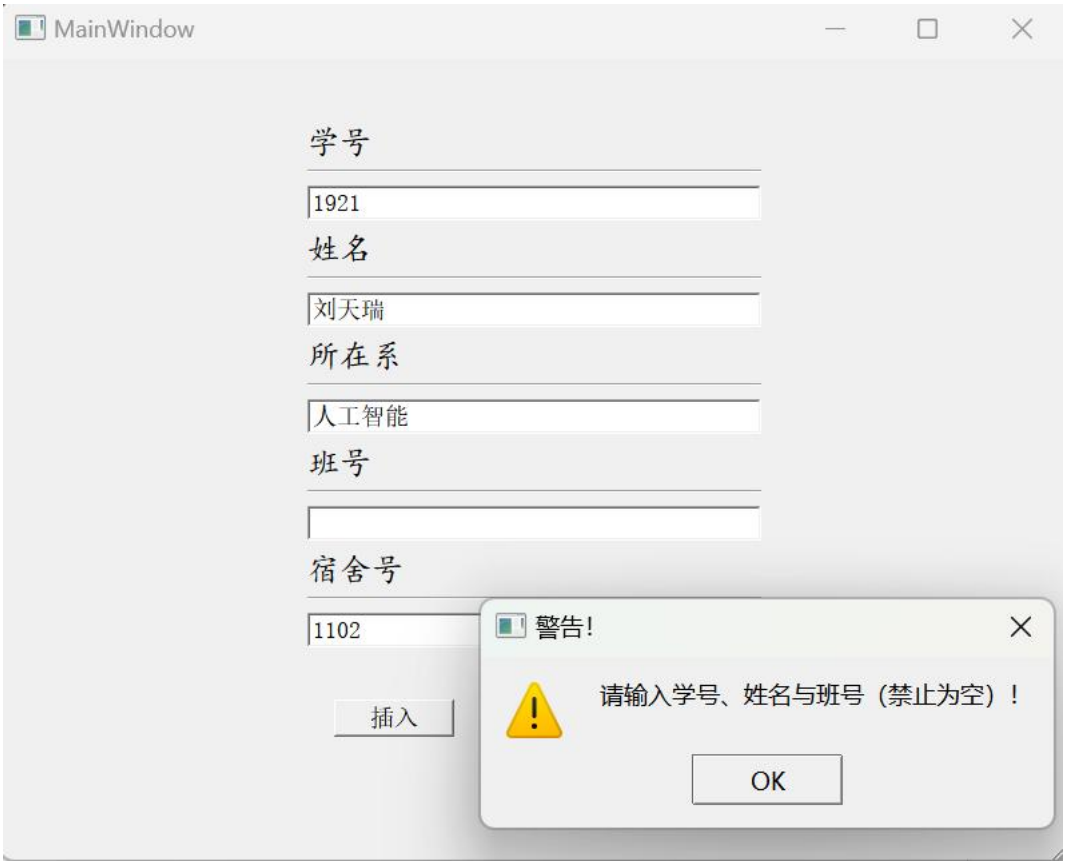

查询信息
查看学生
查看教师
其他信息
Dialog
学号 姓名 所在系名 班号 寝室号
1 1901 赵一 人工智能 1903001 1101
2 1902 钱二 人工智能 1903001 1101
3 1903 孙三 物联网 1903001 1101
4 1904 李五 计算机科学 1903001 1102
5 1905 周六 计算机科学 1903001 1102
6 1906 吴七 生物信息 1903002 1103
7 1907 郑八 信息安全 1903002 1103
8 1908 王八 软件工程 1903002 1103
9 1909 冯九 软件工程 1903002 1103
10 1910 严十 信息安全 1903002 4101
11 1911 陈一 数据科学 1903003 4101
12 1912 褚二 计算机科学 1903003 4102
13 1913 卫三 计算机科学 1903003 4101
14 1914 蒋四 数据科学 1903003 4102
15 1915 沈五 人工智能 1903003 2101

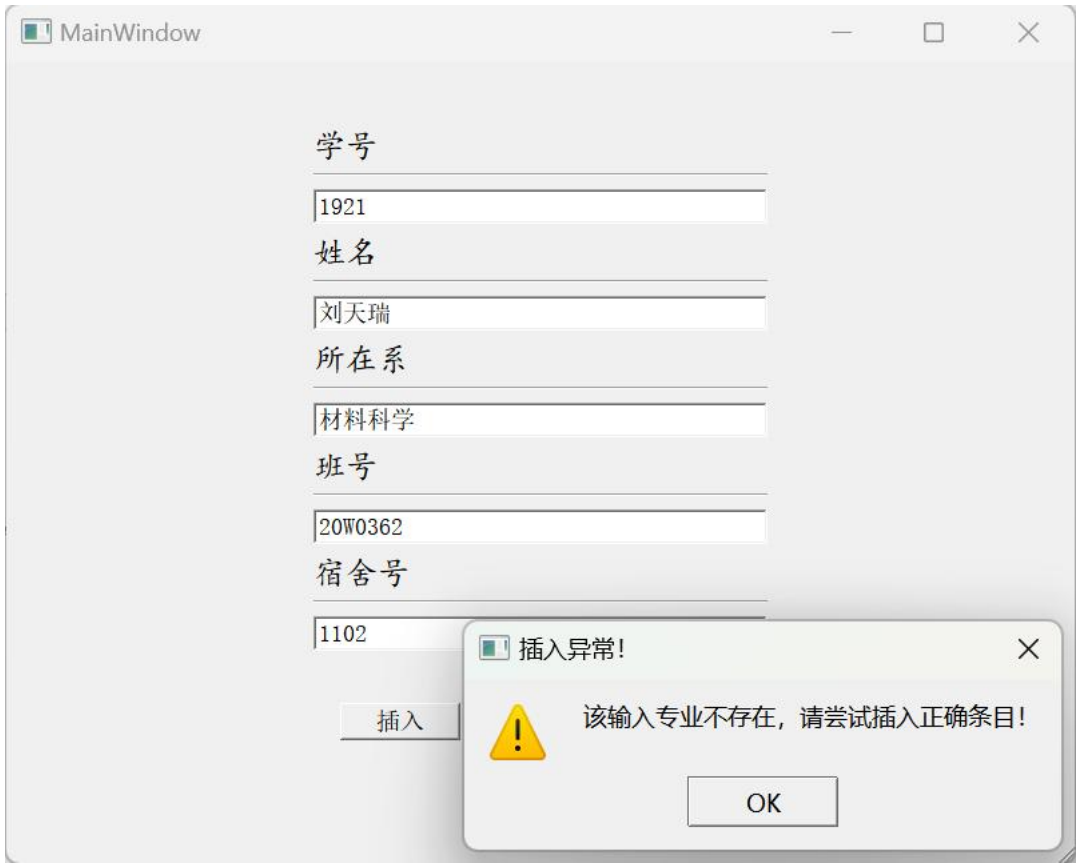

查询信息
查看学生
查看教师
其他信息
Dialog
教工号 姓名 所在系名
1 101 吕一 计算机科学
2 102 孔二 计算机科学
3 103 张三 人工智能
4 104 李四 人工智能
5 105 王五 软件工程
6 106 方六 软件工程
7 107 曹七 信息安全
8 108 华八 信息安全
9 109 何九 数据科学
10 110 苏十 数据科学
11 111 苗一 物联网
12 112 任二 物联网
13 113 黄三 生物信息
14 114 孟四 生物信息
15

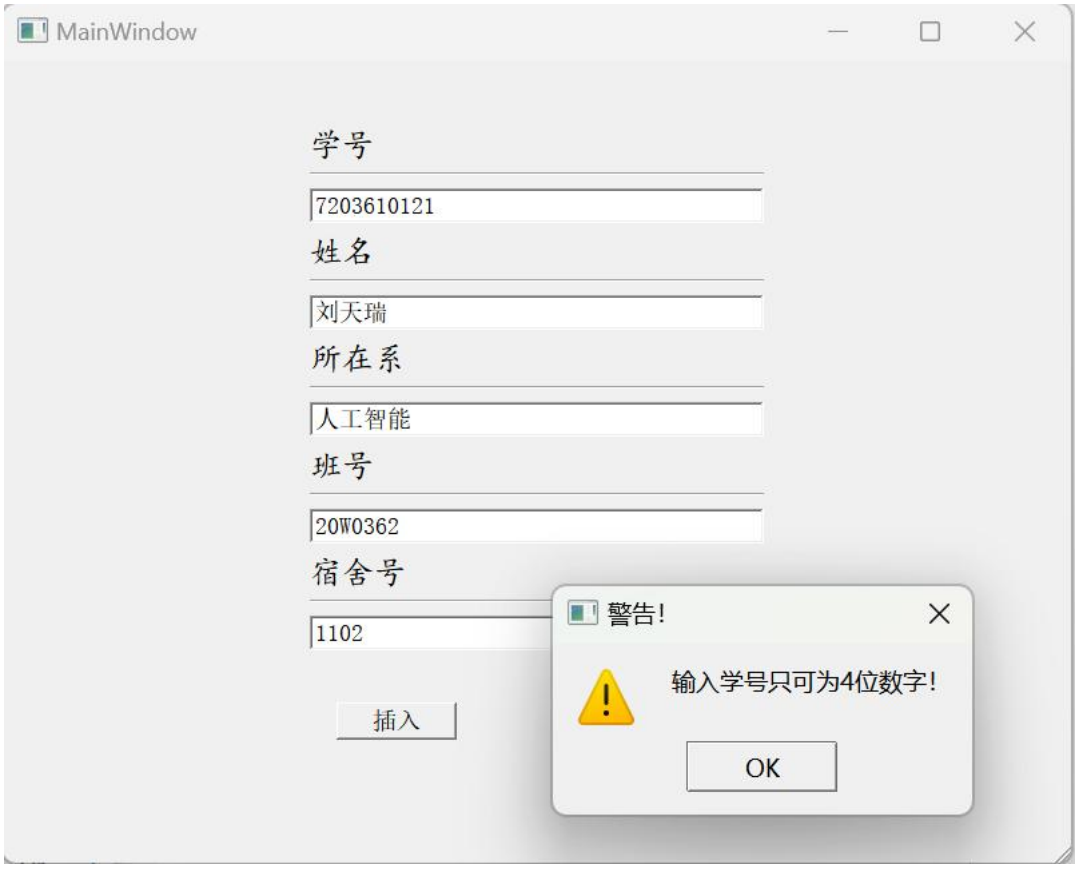
插入操作：

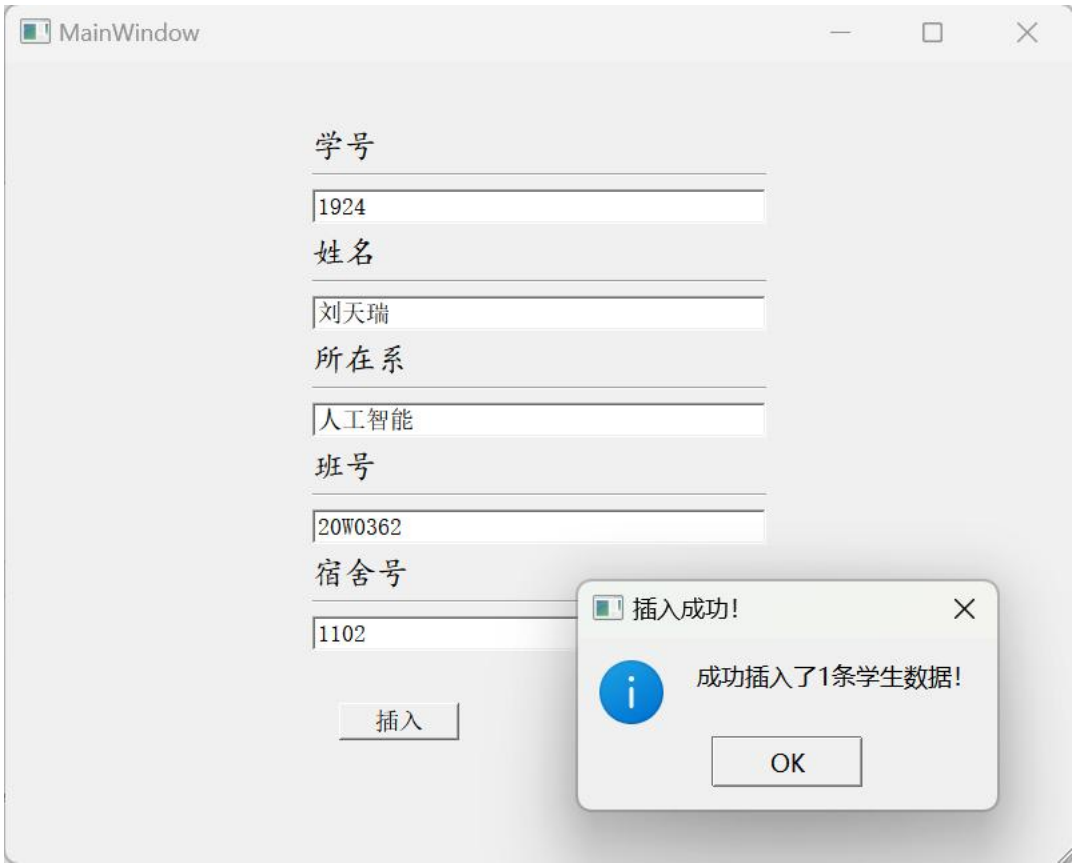
我在这里提供了学生、课程还有成绩的插入操作。不妨先以学生为例：插入操作首先需要判断是否所有的值都为非空的，如果缺少了一条信息，则会提供相关提示。除此之外，还需要通过学生 student 表的主键学号来判断插入的信息是否为重复，如果重复则也会给予提示。最后，如果输入的该专业并未出现在院系 department 表中，则也不会允许被正确地插入进去。部分具体的运行代码如下图所示：

```
160    def insert(self):
161        num, name, de_name, classid, dorm = self.lineEdit.text(), self.lineEdit_2.text(), self.lineEdit_3.text(), \
162                                            self.lineEdit_4.text(), self.lineEdit_5.text()
163        if not num or not name or not de_name or not classid or not dorm: # 插入错误值，空值，提醒
164            QMessageBox.warning(self, '警告！', '请输入学号、姓名与班号（禁止为空）！') # 空值判断
165        elif len(num) != 4:
166            QMessageBox.warning(self, '警告！', '输入学号只可为4位数字！') # 错误值判断
167        else:
168            con = pymysql.connect(host='localhost', port=3306, user='root', password='691179', charset='utf8',
169                                  database='lab2')  # 连接数据库
170            cur = con.cursor()  # 执行sql语句的游标
171            query = 'select * from student where sid=%s'
172            if cur.execute(query, [num]):
173                QMessageBox.warning(self, '插入异常！', '该输入学号已存在，请重新输入！') # 重复值提示（重复信息）
174            elif not cur.execute('select * from department where name = %s', [de_name]):
175                QMessageBox.warning(self, '插入异常！', '该输入专业不存在，请尝试插入正确条目！') # 外键约束
176            else:
177                QMessageBox.information(self, '插入成功！', '成功插入了1条学生数据！')
178                query = 'insert into student(sid, sname, de_name, class, dorm) values (%s, %s, %s, %s, %s)'
179                cur.execute(query, [num, name, de_name, classid, dorm])
180                con.commit()
```




MainWindow
学号
1921
姓名
刘天瑞
所在系
材料科学
班号
20W0362
宿舍号
1102
插入
插入异常!
该输入专业不存在，请尝试插入正确条目!
OK

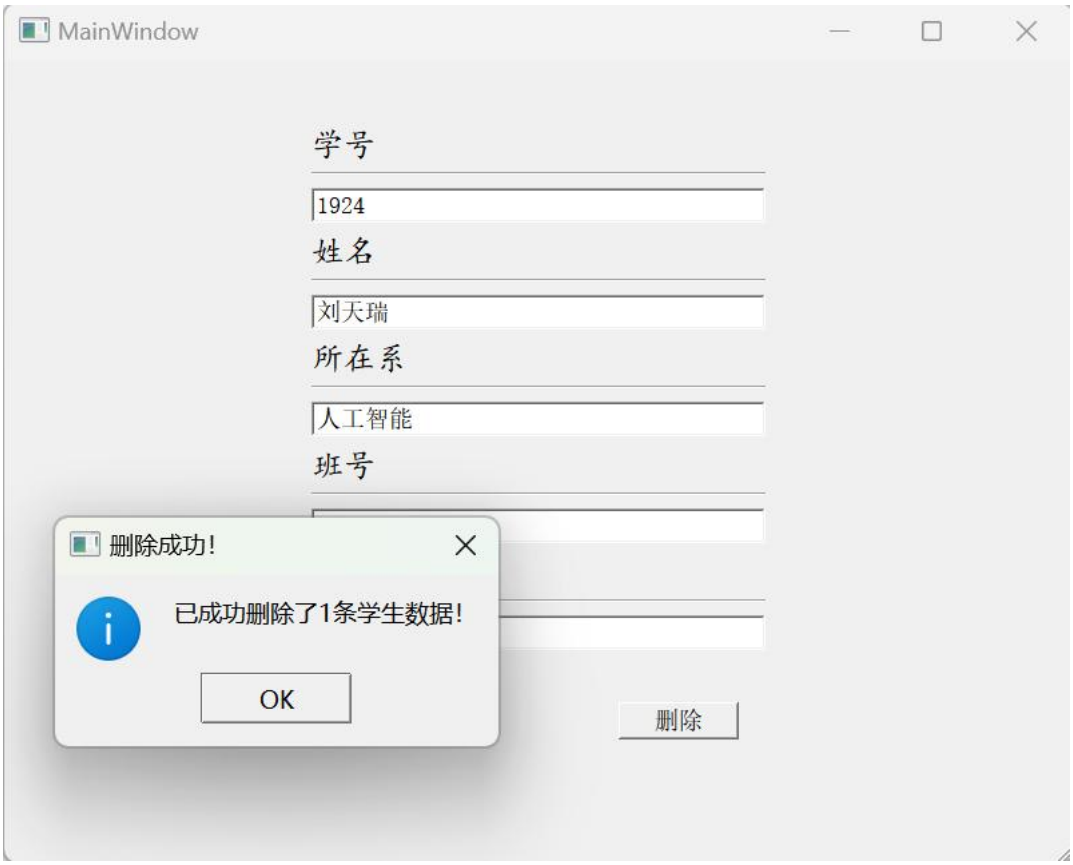

MainWindow
学号
7203610121
姓名
刘天瑞
所在系
人工智能
班号
20W0362
宿舍号
1102
插入
警告!
输入学号只可为4位数字!
OK

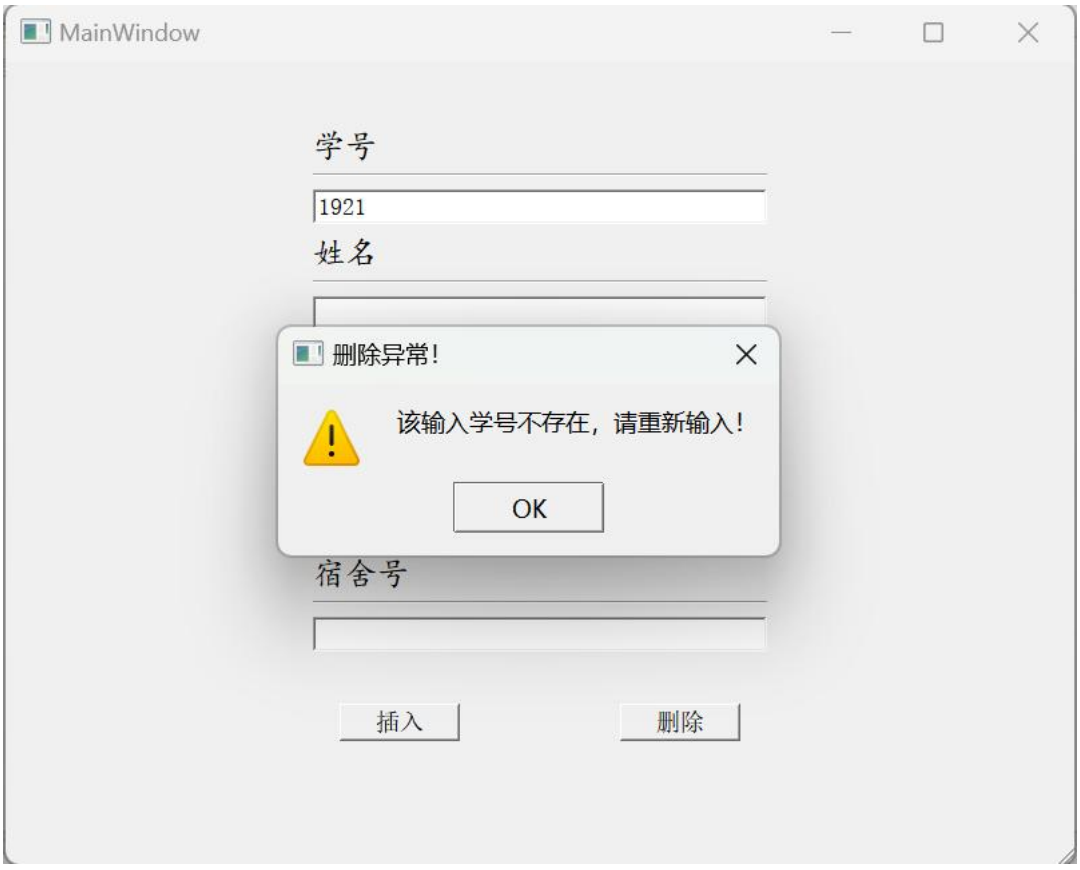


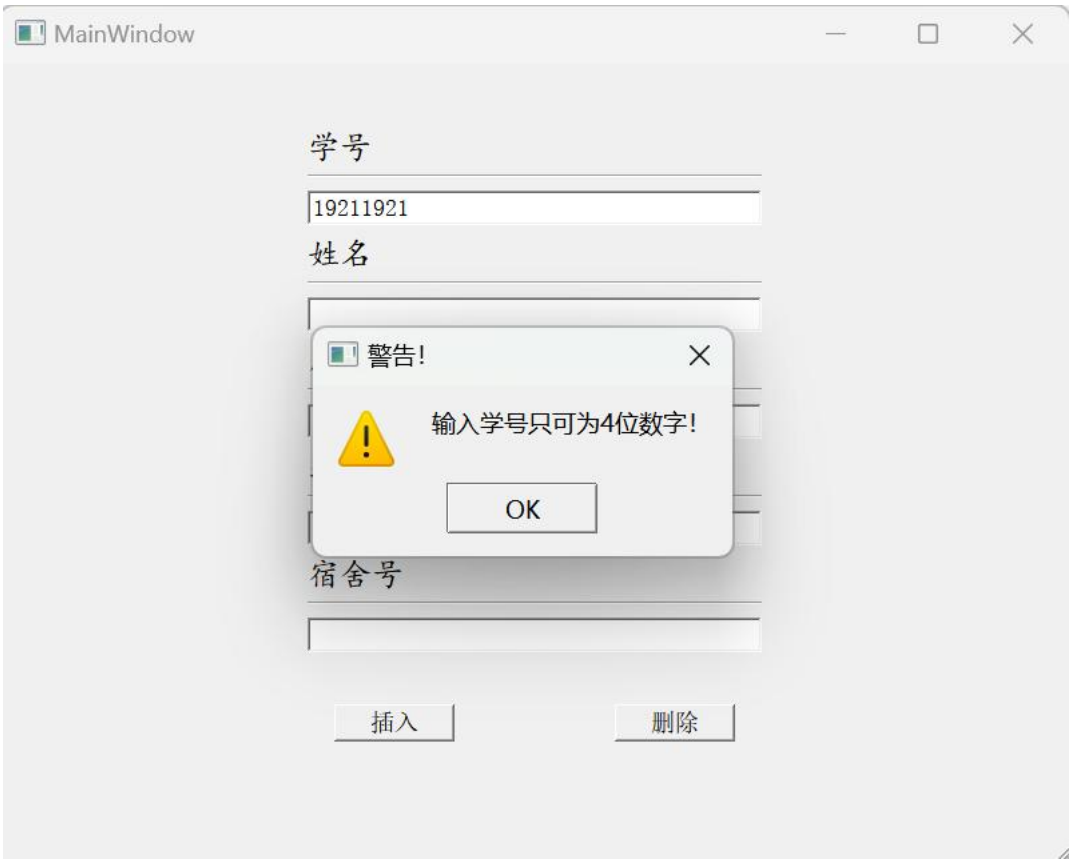
删除操作：

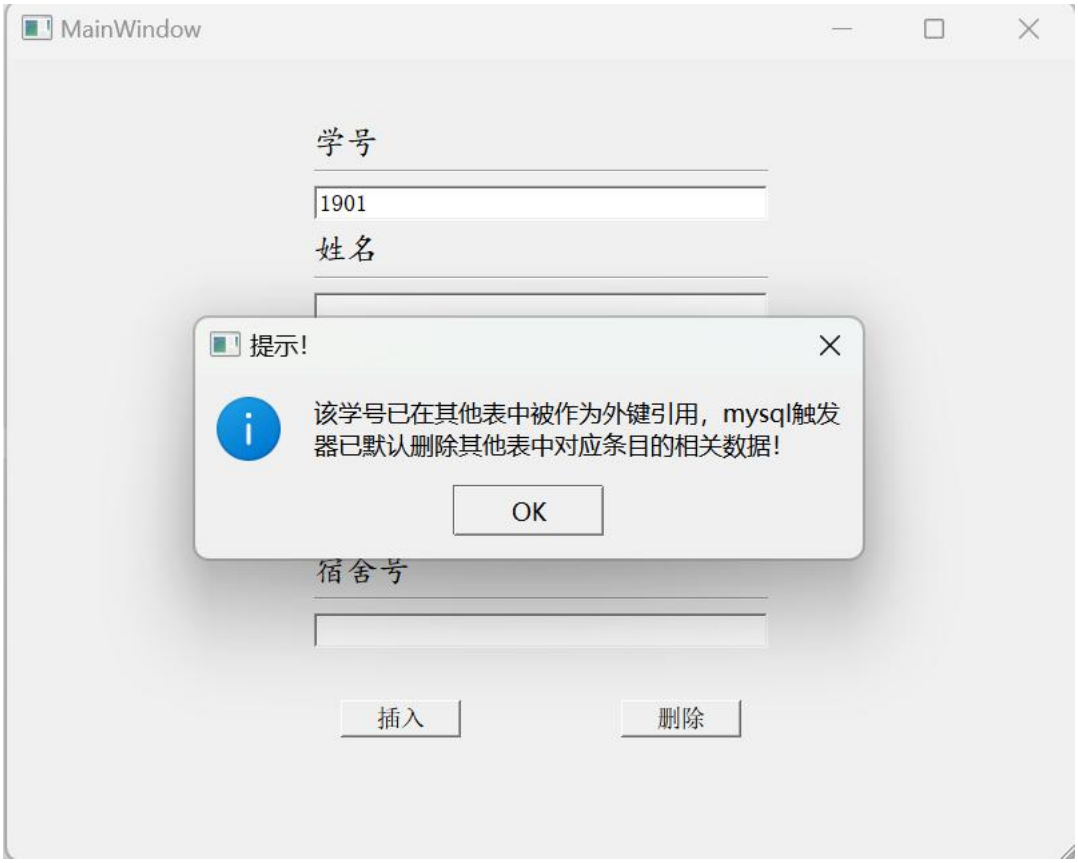
同样的，我以删除学生信息为例：在删除操作中只需要提供正确的学号就能够进行删除操作。同时，我这里进行了 mysql 触发器功能，即如果检测到该学号在成绩 grade 表中也出现了，则将会一并删去了其在成绩 grade 表中的元组。同理，对于删除课程信息也有可能删除成绩 grade 表中的元组，但是值得注意的是删除成绩信息则反而并不会影响到学生和课程信息的完整性。其部分运行的具体关键代码和截图如下图所示：

```
183    def delete(self):
184        num= self.lineEdit.text()
185        if not num:
186            QMessageBox.warning(self, '警告！', '输入学号禁止为空！')
187        elif not num.isdigit() or len(num) != 4:
188            QMessageBox.warning(self, '警告！', '输入学号只可为4位数字！')
189            # 空值与错误判断
190        else:
191            con = pymysql.connect(host='localhost', port=3306, user='root', password='691179', charset='utf8',
192                                  database='lab2')  # 连接数据库
193            cur = con.cursor()  # 执行sql语句的游标
194            query = 'select * from student where sid=%s'
195            if not cur.execute(query, [num]):
196                QMessageBox.warning(self, "删除异常！", "该输入学号不存在，请重新输入！") # 外键约束
197            else:
198                if cur.execute('select * from grade where sid=%s', [num]):
199                    QMessageBox.information(self, '提示！', '该学号已在其他表中被作为外键引用，mysql触发器已默认删除其他表中对应条目的相关数据！')
200                    query = 'delete from grade where sid=%s'
201                    cur.execute(query, [num]) # 触发器：删除成绩grade表中的元组
202                QMessageBox.information(self, '删除成功！', '已成功删除了1条学生数据！')
203                query = 'delete from student where sid=%s'
204                cur.execute(query, [num])
205                con.commit()
```


MainWindow
学号
1924
姓名
刘天瑞
所在系
人工智能
班号
删除成功!
已成功删除了1条学生数据!
OK
删除


MainWindow
学号
1921
姓名
删除异常!
该输入学号不存在，请重新输入!
OK
宿舍号
插入
删除

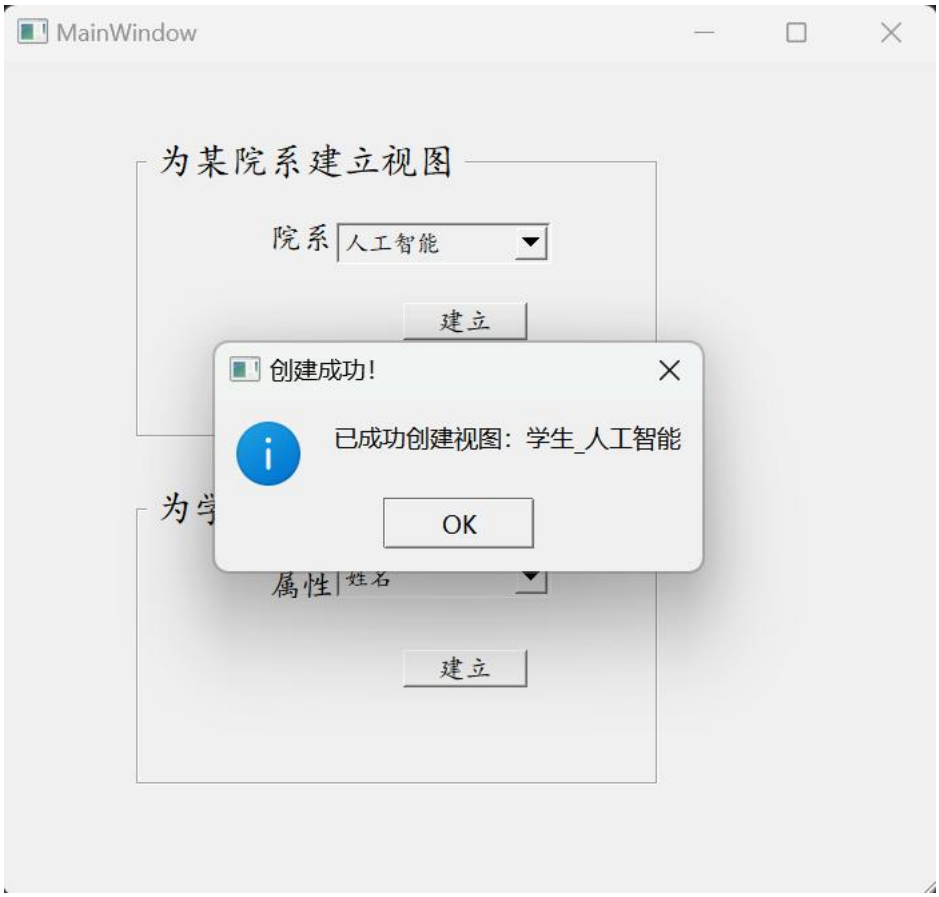

MainWindow
学号
19211921
姓名
警告!
输入学号只可为4位数字!
OK
宿舍号
插入
删除

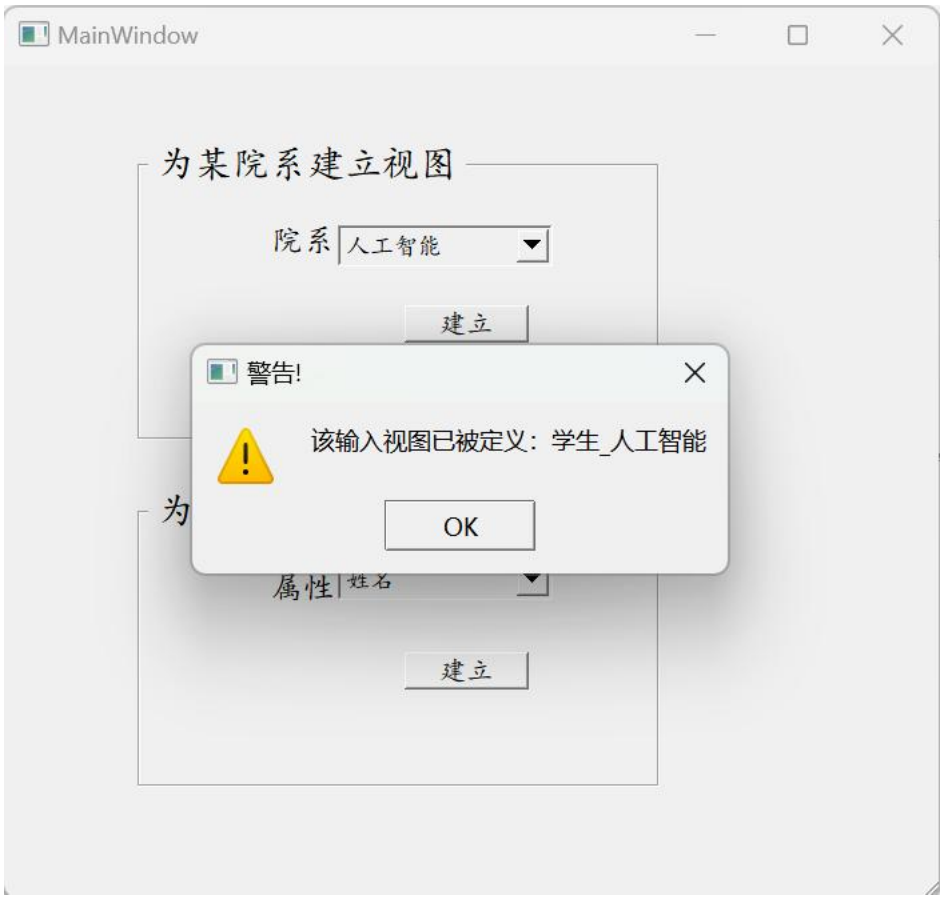

MainWindow
学号
1901
姓名
提示!
该学号已在其他表中被作为外键引用，mysql触发器已默认删除其他表中对应条目的相关数据!
OK
宿舍号
插入
删除

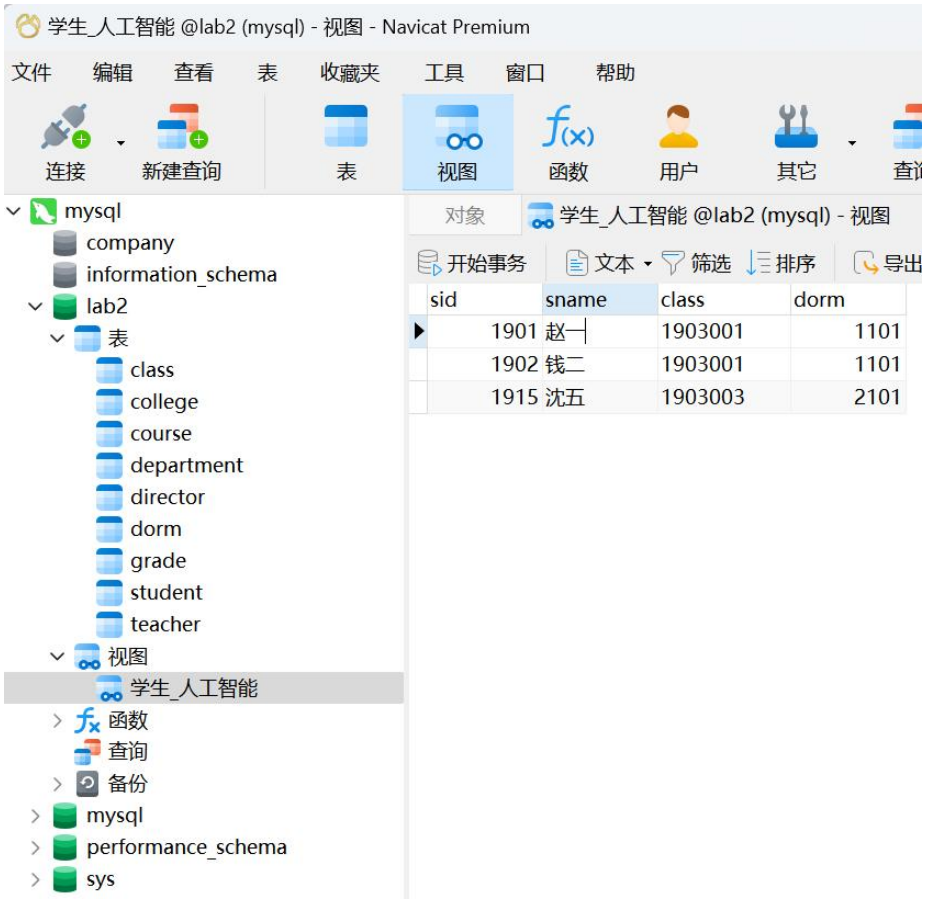
建立视图和索引：

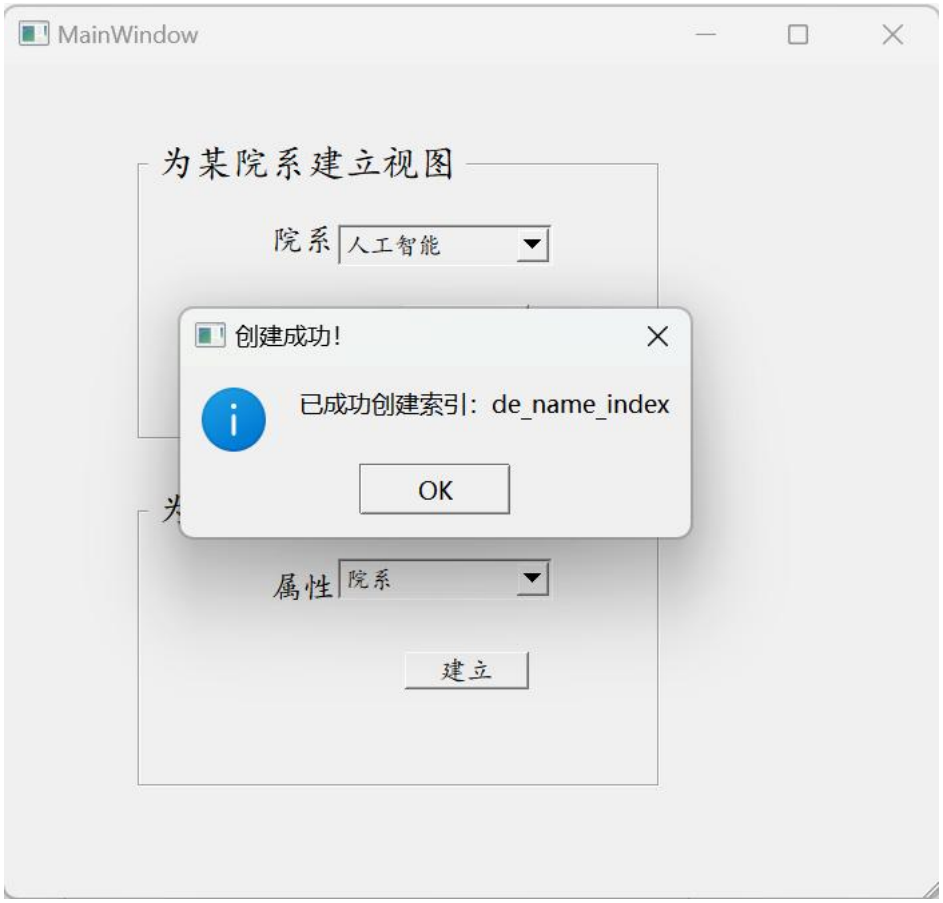
在本次实验中，我还添加了建立视图与建立索引这两项重要的功能。并且建立视图是以学生的院系为分组标准，为相同系的学生建立视图。这里采用了选框的组件，所以并不需要判断空值和错误值。但是需要判断该视图是否已经被创建。其具体运行部分关键代码和结果截图如下图所示：

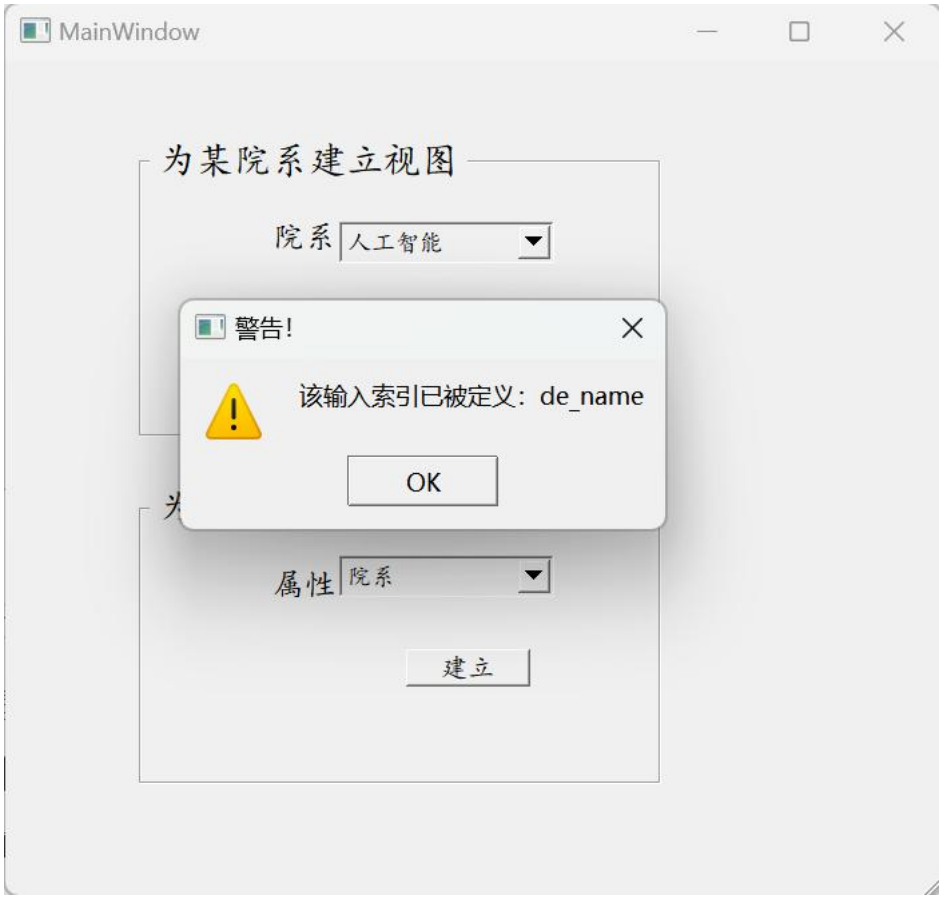
```
def create_department_view(self):
    d_name = self.comboBox.currentText()
    view_name = '学生_' + d_name
    con = pymysql.connect(host='localhost', port=3306, user='root', password='691179', charset='utf8',
                          database='lab2')  # 连接数据库
    cur = con.cursor()
    query = 'select count(*) from information_schema.VIEWS where TABLE_SCHEMA="lab2" and TABLE_NAME=%s'
    cur.execute(query, [view_name])  # 先查询视图是否已被定义
    if cur.fetchone()[0] == 1:
        QMessageBox.warning(self, '警告!', '该输入视图已被定义：' + view_name) # 判断该视图是否被重复创建
    else:
        query = 'create view ' + view_name + ' as select sid ,sname, class, dorm from student where de_name = %s' # 创建语句
        cur.execute(query, [d_name])
        QMessageBox.information(self, '创建成功！', '已成功创建视图：' + view_name)
        cur.close()
        con.close()
```

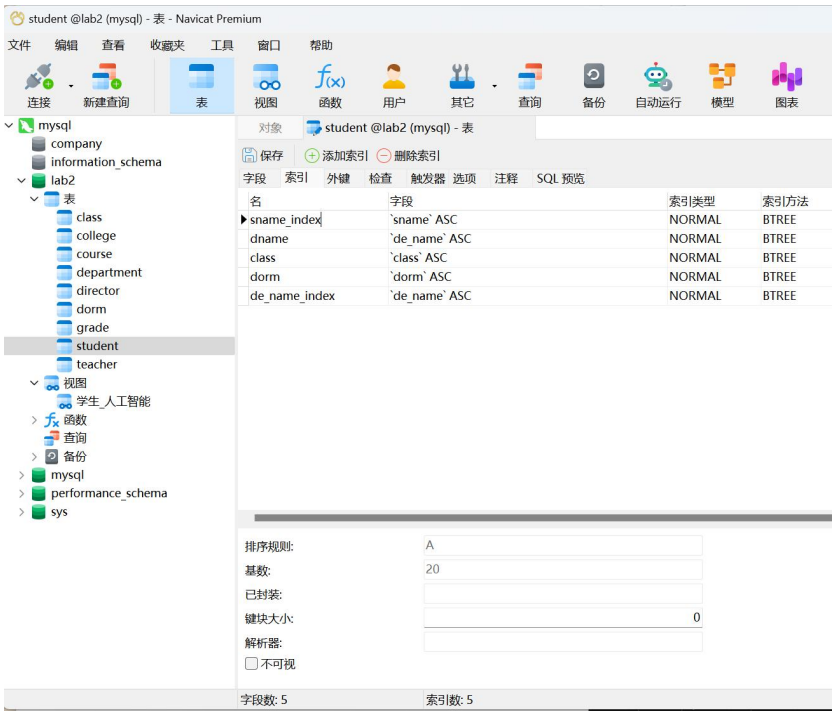



MainWindow
为某院系建立视图
院系
人工智能
建立
警告!
该输入视图已被定义：学生_人工智能
OK
为
属性
姓名
建立


学生_人工智能 @lab2 (mysql) - 视图 - Navicat Premium
文件 编辑 查看 表 收藏夹 工具 窗口 帮助
连接 新建查询 表 视图 函数 用户 其它
mysql
company
information_schema
lab2
表
class
college
course
department
director
dorm
grade
student
teacher
视图
学生_人工智能
函数
查询
备份
mysql
performance_schema
sys
对象
学生_人工智能 @lab2 (mysql) - 视图
开始事务 文本 筛选 排序 导出
sid sname class dorm
1901 赵一 1903001 1101
1902 钱二 1903001 1101
1915 沈五 1903003 2101


MainWindow
为某院系建立视图
院系
人工智能
创建成功!
已成功创建索引：de_name_index
OK
属性
院系
建立

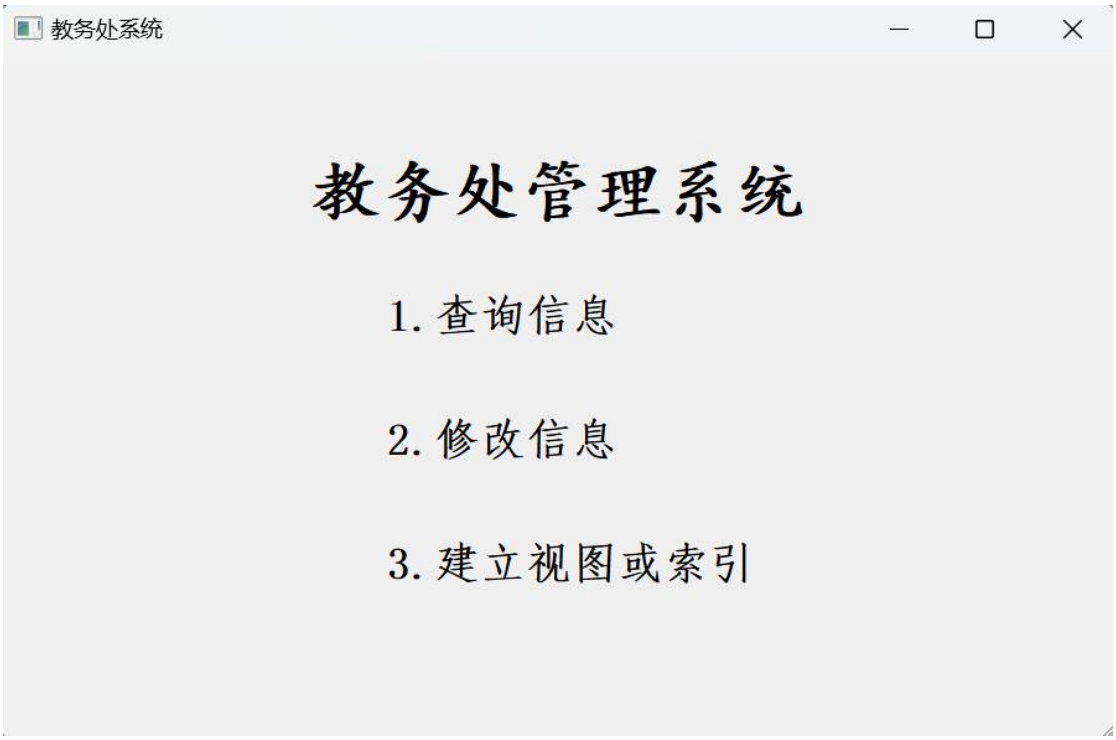

MainWindow
为某院系建立视图
院系
人工智能
警告!
该输入索引已被定义：de_name
OK
属性
院系
建立




```
mysql> show index from student;
+---------+------------+---------------+--------------+-------------+-----------+-------------+----------+--------+------+------
------+---------+---------------+---------+------------+
| Table   | Non_unique | Key_name      | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index
_type | Comment | Index_comment | Visible | Expression |
+---------+------------+---------------+--------------+-------------+-----------+-------------+----------+--------+------+------
------+---------+---------------+---------+------------+
| student |          0 | PRIMARY       |            1 | sid         | A         |           0 | NULL     | NULL   |      | BTREE
      |         |               | YES     | NULL       |
| student |          1 | sname_index   |            1 | sname       | A         |          20 | NULL     | NULL   | YES  | BTREE
      |         |               | YES     | NULL       |
| student |          1 | dname         |            1 | de_name     | A         |           7 | NULL     | NULL   | YES  | BTREE
      |         |               | YES     | NULL       |
| student |          1 | class         |            1 | class       | A         |           4 | NULL     | NULL   | YES  | BTREE
      |         |               | YES     | NULL       |
| student |          1 | dorm          |            1 | dorm        | A         |           7 | NULL     | NULL   | YES  | BTREE
      |         |               | YES     | NULL       |
| student |          1 | de_name_index |            1 | de_name     | A         |           7 | NULL     | NULL   | YES  | BTREE
      |         |               | YES     | NULL       |
+---------+------------+---------------+--------------+-------------+-----------+-------------+----------+--------+------+------
------+---------+---------------+---------+------------+
6 rows in set (0.03 sec)
```

GUI 界面设计：

本次实验的 ui 界面我是基于 Python 里的 PyQt5 库来进行设计和接入的。主页面的设计我将其为了三个主要功能：分别是查询信息功能，修改信息功能以及建立视图或索引的功能。在查询信息功能模块中又可以分为查看学生、查看教师以及其他查询的三个板块；在修改信息功能模块中，还可以分别对学生、课程以及成绩进行插入和删除操作。具体界面如下图所示：


教务处系统
教务处管理系统
1. 查询信息
2. 修改信息
3. 建立视图或索引


查询信息
查看学生
查看教师
其他信息

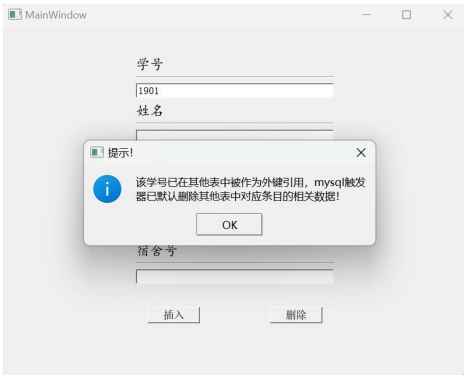

MainWindow
查询选课数目超过n的学生
数目
查询
查询选修某课程的学生信息
课程编号
查询
查询某学生及格课程平均成绩
姓名
查询


MainWindow
修改学生信息
修改课程信息
修改成绩信息


MainWindow
学号
姓名
所在系
班号
宿舍号
插入
删除

MySQL 触发器功能：

在编写进行删除各类信息操作的部分里我增加了 mysql 触发器功能，以删除学生信息为例：如果检测到该输入学生的学号在成绩 grade 表中也出现，则一并删去其在成绩 grade 表中的元组。同理，对于删除课程信息也有可能删除成绩中的元组，但是相反删除成绩信息则不会影响到学生和课程信息。



四、实验心得

列举遇到并解决的问题等。

答：

1. 关于 PyQt 库的使用和各个组件的设置与配置接口代码：因为我之前较少接触过 Python 里专门用来进行 GUI 界面设计的 PyQt 库，所以在这一部分我花了较多时间，搜索了较多的 csdn 博客与相关资料，在一步步地摸索中逐渐完善，感觉受益匪浅，最终将完整的界面设计出来；
2. 在进行插入修改操作的相关代码编写时，因为从数据库 mysql 中查询到的学

号是 int 整型，但是我将学号 item[0]作为输出的时候疏忽没有转换为字符串 str 类型，所以在这一部分一直出错。后来我继续添加了 str(item[0])后，就解决了这个问题：

```
189            for item in cur.fetchall():
190                res.append('学号为'+str(item[0])+'的学生{name}及格科目的平均成绩为'.format(name=name) + '\t' + str(item[1]))
191                print(item[0])
```

3. 还有一些问题就是我刚开始没有考虑足够全面：例如在学生 student 表中插入的新相关数据信息，如果该数据消息里的班号被发现不在班级 class 表中时，理论上此时应该也在班级 class 表中也同时插入新的班号，从而作为一个插入修改操作的 mysql 触发器。但是因为时间原因和相关代码繁琐我也就只做出了删除修改操作的 mysql 触发器功能，并且事务管理也并未完善；
4. 在 Visual Studio 2022 中进行数据库与高级语言的链接：因为我的编程语言为 Python，故首先应该在该环境下安装 pymysql 库，然后调用 connect（）连接函数，设置好本地主机、端口号、用户名、密码、字符格式以及该用户的数据库，再进行 SQL 语句与 python 语句的转换。

```
con = pymysql.connect(host='localhost', port=3306, user='root', password='691179', charset='utf8',
                      database='lab2')  # 连接数据库
```