哈尔滨工业大学

<<数据库系统>>

实验报告

(2020 年度春季学期)

姓名:	史纪元
学号:	1173300919
学院:	计算机科学与技术学院
教师:	史建焘

实验一 数据库应用系统的开发

一、实验目的

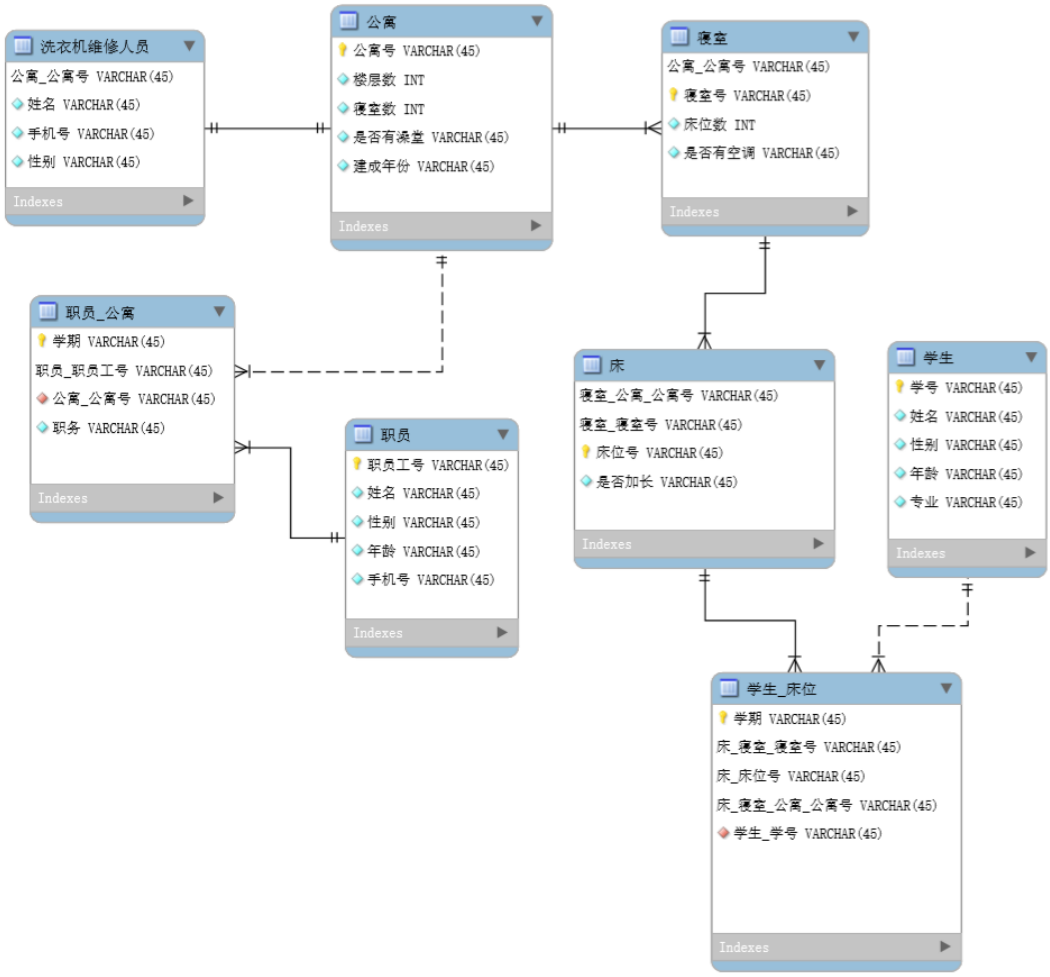
熟练掌握关系数据库系统的使用、SQL 语言；掌握在高级语言中通过嵌入式 SQL 对数据库进行操作，学习简单数据库系统的设计方法，包括数据库概要设计、逻辑设计。

二、实验环境

DBMS 使用 MySQL，系统开发语言使用 python 3.6。

三、实验过程及结果

学生公寓管理系统 E-R 图



实体与联系

在该系统中，每个学生所住公寓会随着学期发生变动，因此在学生和床位之间有一个相交实体“学生_床位”，同理有相交实体“职员_公寓”，学生-床位、职员-公寓之间是**多对多联系**。另外规定每个公寓有唯一的洗衣机维修人员，且每位洗衣机维修人员只能负责一栋公寓，因此洗衣机维修人员与公寓之间是**一对一联系**，其他实体之间例如公寓-寝室是**一对多联系**。

库表设计

定义表：

dorm 表：

列名	类型	完整性约束	解释
dorm_id	varchar	主键	公寓号
floors	int	非空	楼层数
rooms	int	非空	寝室数
has_bath	varchar	非空	是否有澡堂
built_year	varchar	非空	建成年份

room 表：

列名	类型	完整性约束	解释
dorm_id	varchar	主键、外键	公寓号
room_id	varchar	主键	寝室号
beds	int	非空	床位数
has_ac	varchar	非空	是否有空调
built_year	varchar	非空	建成年份

bed 表：

列名	类型	完整性约束	解释
dorm_id	varchar	主键、外键	公寓号
room_id	varchar	主键、外键	寝室号
bed_id	varchar	主键	床位号
is_long	varchar	非空	是否加长

student 表：

列名	类型	完整性约束	解释
Sid	varchar	主键	学号
Sname	varchar	非空	学生姓名
Sgender	varchar	非空	学生性别
Sage	int	非空	学生年龄
Smajor	varchar	非空	学生专业

student_bed 表：

列名	类型	完整性约束	解释
semester	varchar	主键	学期
dorm_id	varchar	主键、外键	公寓号
room_id	varchar	主键、外键	寝室号
bed_id	varchar	主键、外键	床位号
Sid	varchar	外键	学号

employee 表：

列名	类型	完整性约束	解释
Eid	varchar	主键	职员工号
Ename	varchar	非空	职员姓名
Egender	varchar	非空	职员性别
Eage	int	非空	职员年龄
E_phonenumber	varchar	非空	职员手机号

employee_dorm 表：

列名	类型	完整性约束	解释
semester	varchar	主键	学期
Eid	varchar	主键、外键	职员工号
dorm_id	varchar	外键	公寓号
job	varchar	非空	职务

maintenance_worker 表：

列名	类型	完整性约束	解释
dorm_id	varchar	主键、外键	公寓号
Mname	varchar	非空	维修人员姓名
M_phonenumber	varchar	非空	维修人员手机号
Mgender	varchar	非空	维修人员性别

创建数据表的 SQL 语句略。

创建视图：

为计算机系学生创建视图，SQL 语句为：

```
create view compstud as
(select * from student where Smajor='计算机');
```

为有澡堂的公寓创建视图，SQL 语句为：

```
create view dorm_with_bath as
(select * from dorm where has_bath='是');
```

建立索引：

在学生表中，学生姓名是常用属性且不是主键，因此为学生姓名建立索引，SQL 语句为：

```
create index idxSname on student(Sname);
```

在 MySQL workbench 查看建立的索引，可以看到建立成功：

```
4 •  show index from student;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_
student	0	PRIMARY	1	Sid	A	7	NULL
student	1	idxSname	1	Sname	A	7	NULL

系统功能

系统界面

其中前 8 个选项提供对具体某个表的插入、删除、查询操作，“更多功能”选项提供连接查询、嵌套查询、分组查询、对视图的查询等功能。

本系统的 GUI 界面用 QT designer 构建。

查询操作

用户输入任意查询条件的组合，系统会返回 MySQL 数据库中满足条件的元组信息，用户也可以输入'%'等通配符进行模糊查询。

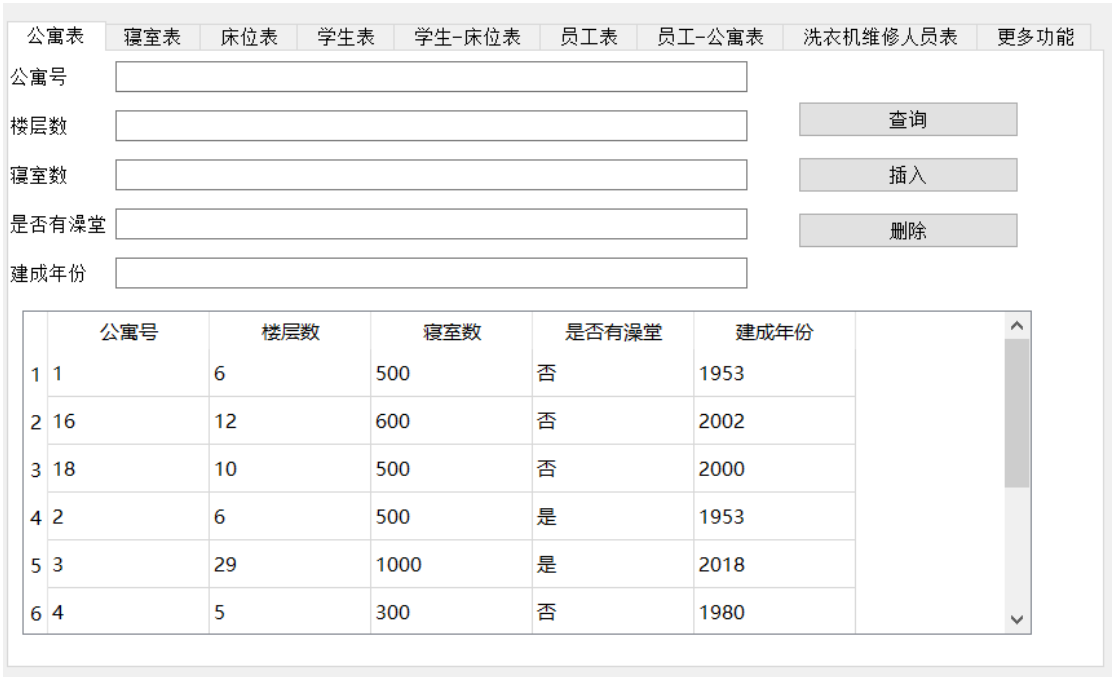
以公寓表为例，当用户只在“公寓号”栏输入 1，其他栏不输入时，系统会显示 dorm_id 为‘1’的元组；当用户在“楼层数”输入 6、“寝室数”输入 500

时，系统会显示满足 floors=6，rooms=500 的元组，如下图所示：

查询公寓号为‘1’的公寓

查询楼层数为 6 且寝室数为 500 的公寓

如果用户什么都不输入（或只输入空格），则系统会返回表中的所有元组：



查询操作代码

首先连接数据库，并获得游标。

```
def query(self):
    try:
        db = pymysql.connect(host='localhost', user='root',
password='1235', database='studentdormitory')
        cursor = db.cursor()
    except pymysql.Error:
        print('连接数据库失败')
```

然后根据 pushbutton 组件的 id 来判断本次查询是哪个表下的按钮发出的（这些按钮都绑定到同一个 query 方法），这里以公寓表的查询为例：初始化 condition=''，用来保存查询条件，然后从公寓表对应的每个 lineEdit 组件中提取用户输入，并动态构造 SQL 语句，这里对于第一个输入栏为空的情况，并不是直接将 condition 也设为空，而是设为'dorm_id like '%''（dorm_id 无空值），这样对于后面的每个查询条件，其都可以以'and'开头而不需判断前面是否存在条件。条件构造完成后，令 sql='select * from dorm where '+condition，并执行 sql 语句。

```
button = ui.sender().objectName()  # 判断是哪个表下的查询
condition = ''
if button == 'pushButton':  # 公寓表查询
    if len(self.lineEdit_2.text().split()) != '':
        condition+='dorm_id like \''+self.lineEdit_2.text()+'\' '
    else:
        condition+='dorm_id like \'%\' '
    if len(self.lineEdit.text().split()) != 0:
        condition+='and floors like \''+self.lineEdit.text()+'\' '
    if len(self.lineEdit_4.text().split()) != 0:
        condition+='and rooms like \''+self.lineEdit_4.text()+'\' '
    if len(self.lineEdit_3.text().split()) != 0:
        condition+='and has_bath like \'' + self.lineEdit_3.text()+'\' '
    if len(self.lineEdit_5.text().split()) != 0:
        condition+='and built_year like \''+self.lineEdit_5.text()+'\''
    sql = 'select * from dorm where ' + condition
    cursor.execute(sql)
```

执行 sql 语句后，获得运行结果，然后把元组信息一行一行输入到 tableWidget 组件中，显示给用户。

```
result = cursor.fetchall()
self.tableWidget.setRowCount(0)
for row_num, row_data in enumerate(result):
    self.tableWidget.insertRow(row_num)
    for column_number, data in enumerate(row_data):
```

```
        self.tableWidget.setItem(row_num, column_number,
QtWidgets.QTableWidgetItem(str(data)))
```

对其他表的查询同理，只需要修改组件名、构造查询中的属性名和表名即可。

插入操作

插入功能允许用户向任一表中插入一行数据，插入的数据**需满足数据库的完整性约束**，如主键不能重复、不能插入空值等（本系统将用户输入空字符串或空格视为插入空值），若插入的数据不满足完整性约束将弹出提示信息。

以寝室表为例：

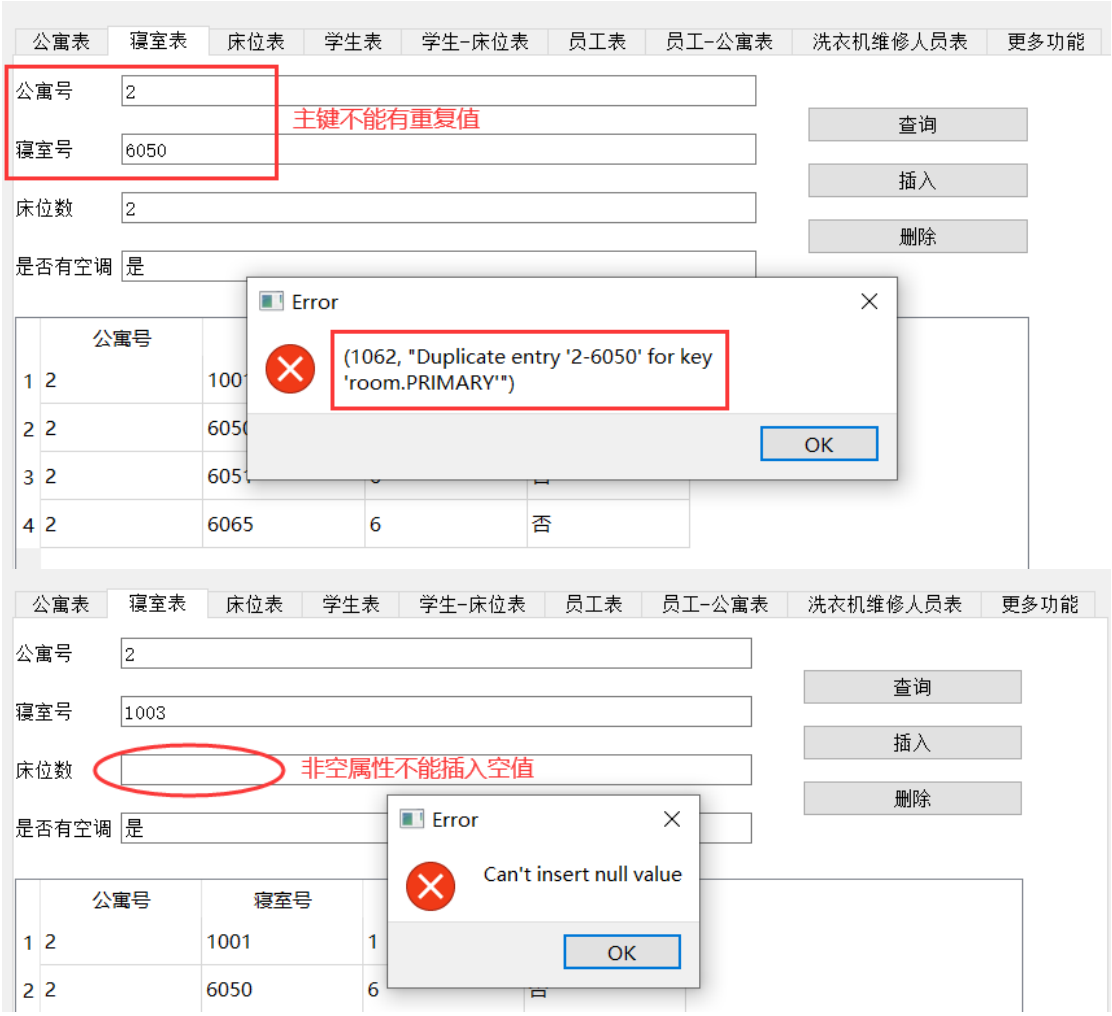
首先查询之前已经插入的 2 公寓的寝室信息:

再向寝室表插入一组数据(2, 6051, 6 否)，提示插入成功:

再次查询 2 公寓的寝室信息:

可以看到，元组成功插入到了数据库中。

若用户插入的数据不满足完整性约束，则会提示错误信息：



插入操作代码

与查询功能一样，首先需要连接数据库，然后根据 pushbutton 组件的 id 判断是哪个表下的插入操作，不同的表需要构建不同的 sql 插入语句。

以对寝室表的插入为例，因为该表所有属性都要求非空，因此判断用户是否插入空值比较简单，只需判断是否存在属性的输入长度为 0 即可。若有为 0 的字段，则调用 QMessageBox.critical()方法显示报错信息。

若用户没有输入空值，则根据用户的输入构建 sql 插入语句。

```
button = ui.sender().objectName()
if button == 'pushButton_2':
    # 判断是否插入空值
    if len(self.lineEdit.text()) * len(self.lineEdit_2.text()) *
len(self.lineEdit_3.text()) * len(self.lineEdit_4.text()) *
len(self.lineEdit_5.text()) == 0:
        QMessageBox.critical(self, 'Error', 'Can\'t insert null value')
        return
    sql = 'insert into dorm(dorm_id, floors, rooms, has_bath,
built_year) values ' + \
          '(\'' + self.lineEdit_2.text() + '\', ' + self.lineEdit.text()
+ ',' + self.lineEdit_4.text() + ',\'' \
          + self.lineEdit_3.text() + '\',\'' + self.lineEdit_5.text() +
'\');'
```

接下来用 try-except 将构建的 sql 语句交给 DBMS 执行，若成功插入则提示插入成功，若未成功执行（例如插入的数据不满足定义的完整性约束）则报错，并显示 DBMS 返回的错误信息。最后关闭与数据库的连接。

```
try:
    cursor.execute(sql)
    db.commit()
    QMessageBox.information(self, '提示', '插入成功')
except pymysql.Error as err:
    QMessageBox.critical(self, 'Error', str(err))
db.close()
```

删除操作

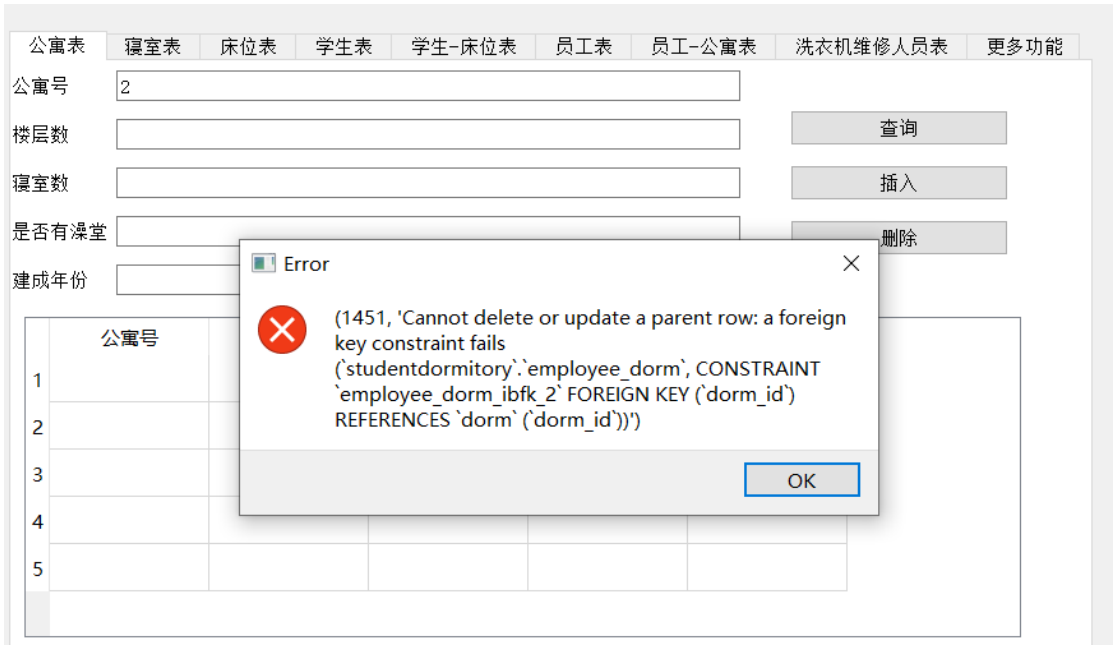
用户可以输入全部属性的值来删除某一特定元组，也可以只输入部分属性的值来删除所有满足该条件的元组（在不破坏完整性约束的条件下）。若用户的删除操作破坏了完整性约束，则系统会报错。

以洗衣机维修人员表为例，首先查询该表：

删除所有性别为女的记录，提示删除成功：

再次查询，可以看到成功删除了这些元组：

若用户的删除操作破坏了数据库的完整性约束（例如破坏了外键约束），则删除失败并弹出报错信息。例如，在公寓表中删除公寓号为 2 的元组，由于在其他表中公寓号为外键且存在公寓号为 2 的记录，因此这个删除操作会破坏外键约束，系统报错：



删除操作代码

删除操作与查询操作有些类似，都需要先在表中找到对应的元组，只不过查询是输出这些元组，而删除操作是将这些元组从数据库中删除，故在此略去连接数据库和构建 sql 语句的代码。

得到动态构建的 sql 语句后，用 try-except 将 sql 语句交给 DBMS 执行，若 DBMS 成功执行了 sql 语句则显示“删除成功”，否则报错并显示 DBMS 返回的错误代码。

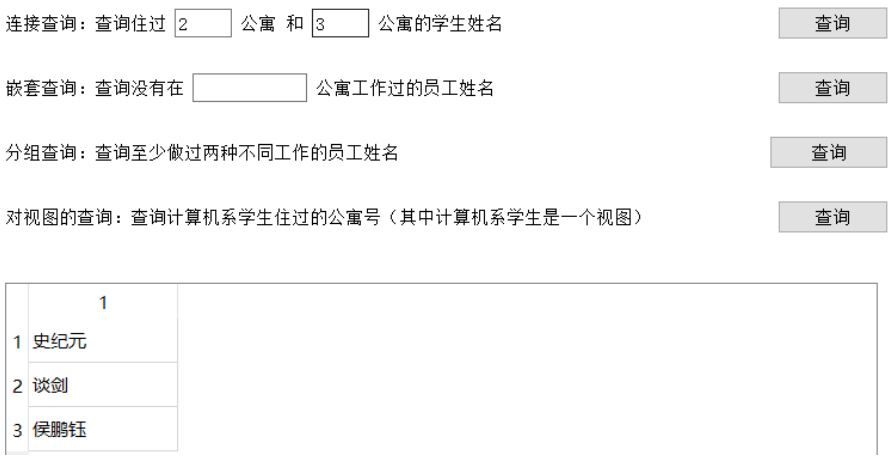
```
try:
    cursor.execute(sql)
    db.commit()
    QMessageBox.information(self, '提示', '删除成功')
except pymysql.Error as err:
    QMessageBox.critical(self, 'Error', str(err))

db.close()
```

其他功能

除了对于单个表的插入、删除、查询操作外，本系统还提供几种常用的连接查询、嵌套查询、分组查询等功能，在“更多功能”选项中：

连接查询：查询住过 A 公寓和 B 公寓的学生姓名，其中 A、B 为用户输入的公寓号，例如查询住过 1 公寓和 2 公寓的学生姓名：

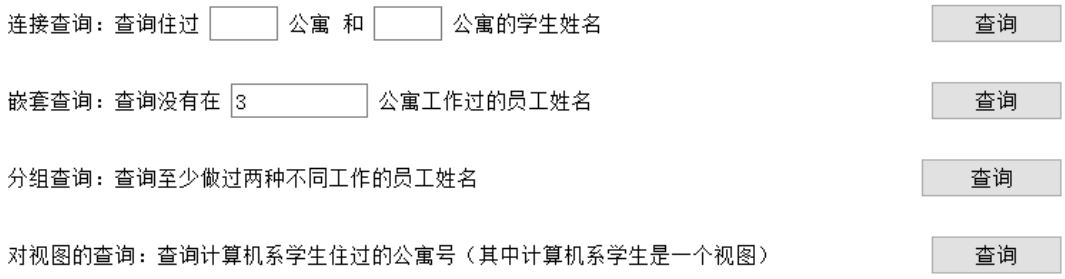


对应的 sql 语句为：

```
sql = 'select distinct Sname from student, student_bed s1, student_bed
s2 \
       where student.Sid=s1.Sid and student.Sid=s2.Sid and\
       s1.dorm_id=\'' + self.lineEdit_10.text() + '\' and
s2.dorm_id=\'' + self.lineEdit_11.text() + '\';'
```

嵌套查询：查询没有在 A 公寓工作过的员工姓名，A 为用户输入

例如查询没有在 3 公寓工作过的员工姓名：

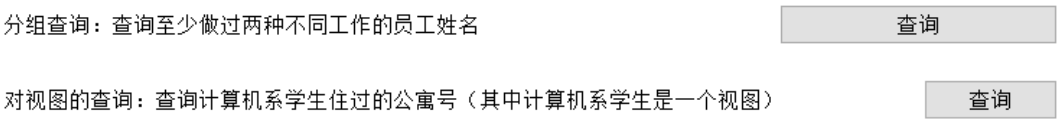


	1
1	李四
2	冯雪娟
3	陈义
4	赵四

对应的 sql 语句为：

```
sql = 'select Ename from employee where Eid not in (select Eid from
employee_dorm where dorm_id=\'' + \
     self.lineEdit_14.text() + '\');'
```

分组查询：查询至少做过两种不同工作的员工姓名

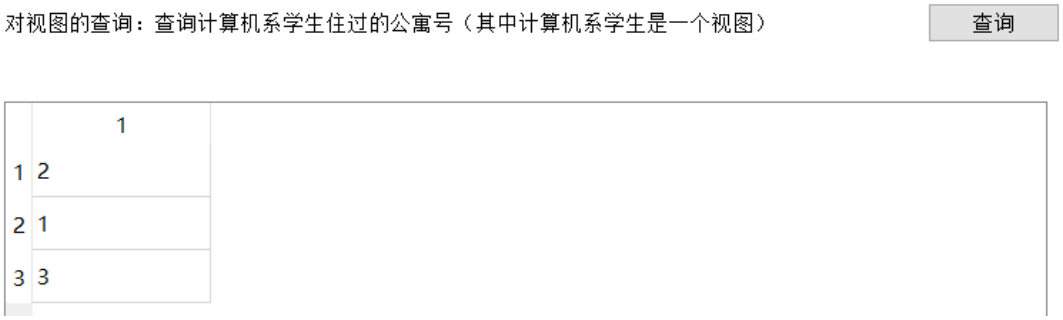


	1
1	陈义

对应的 sql 语句为：

```
sql = 'select Ename from employee where Eid in(select Eid from
employee_dorm group by Eid having count(distinct job)>=2);'
```

以及对视图的查询：查询计算机系学生住过的公寓号：



对应的 sql 语句为：

```
sql = 'select distinct dorm_id from student_bed where Sid in (select
Sid from compstud);'
```

四、实验收获

经过本次实验，我对数据库的建立、表的建立、插入、删除、查询有了更直观的认识，体会到了数据库完整性约束的作用和重要性，初步掌握了嵌入 SQL 语言的使用和动态构建 SQL 语句的方法，另外 GUI 的制作也让我熟悉了 pyqt 和 QT designer 的使用。

本次实验让我认识到了一个真正能运用到实际中的数据库的建立是需要很认真的设计、并且工作量很大的，我应该努力学习数据库这门课，争取将知识掌握牢固。