哈尔滨工业大学

〈〈数据库系统〉〉

实验报告二

(2023 年度秋季学期)

姓名:	张智雄
学号:	2021112845
学院:	计算学部
教师:	程思瑶

实验二 数据库系统开发

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、MySQL 关系数据库管理系统、Anaconda+Python3.8 编程环境。

三、实验过程及结果

本实验拟开发一个学校信息管理数据库系统，并基于 PyQt5 和 pymysql 库实现可视化界面，实验过程及结果如下：

3.1 绘制系统的 E-R 图（包括 8 个实体和 7 个联系）

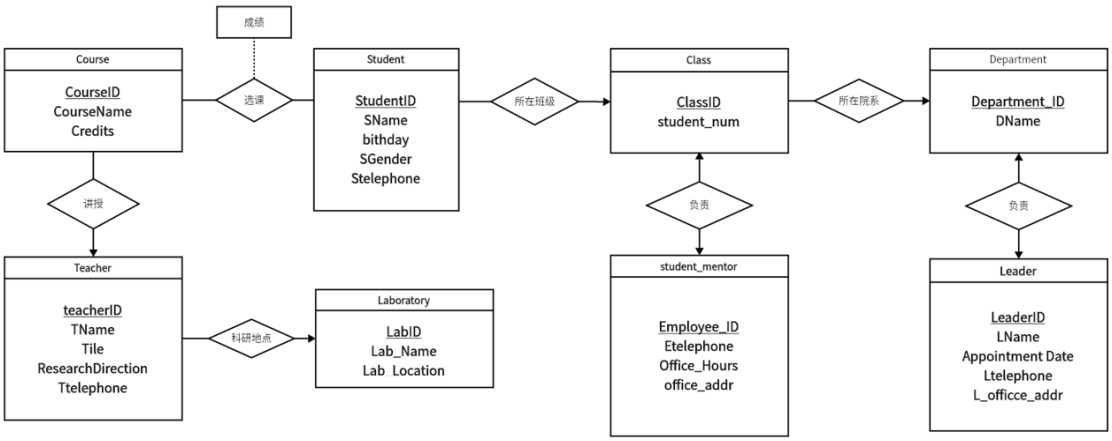


图 1 学校信息管理数据库系统 E-R 图

上图中的关系 8 个实体和 7 个联系，实体包括学生 Student、课程 Course、班级 Class、教师 Teacher、实验室 Laboratory、院系 Department、辅导员 Student mentor、院领导 Leader。

联系可分为一对一联系、一对多联系、多对多联系：

表格 1 联系的类别与含义

联系类别	联系名称	联系含义
一对一联系	**负责**	每个系只有一个系主任，一个系主任只能负责一个系
	负责	一个班只有一个辅导员，一个辅导员只能负责一个班
多对一联系	**所在院系**	一个班只能属于一个院系，一个院系可以包含多个班级
	所在班级	一名学生只能属于一个班级，一个班级可以包含多名学生
	讲授	一门课程只能由一位老师讲授，一位老师可以讲授多门课程
	科研地点	一位老师只能属于一个实验室，一个实验室可以拥有多个老师
多对多联系	**选课**	一名学生可以选修多门课，一门课可以同时被多名学生选修

3.2 根据实体与联系建立关系

为上述概念数据库模式中的每个普通实体集及实体间联系建立关系，建立过程中体现关系完整性约束：主键约束、外键约束，空值约束，具体如下：

3.2.1 简单实体建立关系

本实验采用的均为普通实体集，因此直接将实体集中所有简单属性作为对应关系的属性，而实体集的码即是关系的主码。

1) 学生信息关系 Student (Student_ID, Student_NAME, birthday, SGender, telephone)

使用 SQL 的 CREATE TABLE 语句建立关系：

```
CREATE TABLE student (
    Student_ID CHAR(6) NOT NULL,
    Student_NAME CHAR(20) NOT NULL,
    birthday DATE,
    SGender varchar(6),
    telephone CHAR(8) NOT NULL,
    PRIMARY KEY (Student_ID)
);
```

其中，设置 Student_ID 为主码，确保学号的唯一性和非空性；同时设置该关系中的 Student_ID，Student_NAME，telephone 属性均不能为空（空值约束）。

```
LOAD DATA LOCAL INFILE "path\\to\\student.txt" INTO TABLE student
SELECT * FROM student
```

而后从文本文件导入数据后，查询可视化如下图所示：

	Student_ID char(6)	* Student_NAME char(20)	bithday date	SGender varchar(6)	* telephone char(8)	total_credits int(11)
1	123345	Aiden Anderson	2003-12-03	Male	888-8888	0
2	123456	Alice Johnson	2002-05-15	Female	123-4567	3
3	234567	Bob Smith	2003-08-22	Male	987-6543	0
4	345678	Emily Davis	2005-11-30	Female	111-2222	0
5	456789	Michael Brown	2003-02-10	Male	555-5555	0
6	567890	Sophia Miller	2005-07-18	Female	876-5432	0
7	678901	James Wilson	2004-04-05	Male	333-3333	0
8	789012	Olivia Jones	2003-09-12	Female	444-4444	0

图 2 学生信息关系实例

2) 课程信息关系 Course (course_ID, course_name, credits)

使用 SQL 的 CREATE TABLE 语句建立关系：

```
CREATE TABLE course (
    course_ID CHAR(4) NOT NULL,
    course_name CHAR(15) NOT NULL,
    credits INT NOT NULL,
```

```
    PRIMARY KEY (course_ID)
);
```

其中，设置 course_ID 为主码，该属性具有唯一性和非空性；同时设置该关系中的所有属性不能为空（空值约束）。

而后从文本文件导入数据后，查询可视化如下图 3 左所示：

course_ID char(4)	* course_name char(15)	* credits int(11)
1001	DeepLearning	3
1002	OS	3
1003	CPU_design	2

laboratory_ID char(3)	* laboratory_NAME char(20)	laboratory_addr char(3)
102	lab2	B32
103	lab3	C11
201	lab1	A03

图 3　关系实例（左为课程信息关系，右为实验室信息关系）

3)　教师信息关系 Teacher(teacher_ID, teacher_name, Title, ResearchDirection, Ttelephone)

使用 SQL 的 CREATE TABLE 语句建立关系：

```
CREATE TABLE teacher (
    teacher_ID CHAR(4) NOT NULL,
    teacher_name CHAR(15) NOT NULL,
    Title CHAR(15),
    ResearchDirection CHAR(15),
    Ttelephone CHAR(8),
    PRIMARY KEY (teacher_ID)
);
```

其中，设置 teacher_ID 为主码，确保职工号具有唯一性和非空性；同时设置该关系中的 teacher_ID，teacher_name 属性不能为空（空值约束）。

而后从文本文件导入数据后，查询可视化如图 4 所示：

	teacher_ID char(4)	* teacher_name char(15)	Title char(15)	ResearchDirection char(15)	Ttelephone char(8)
1	1806	Clark	Assistant Profe	CV	818-5234
2	1903	Alice	Professor	CPU	111-2221
3	2105	Alan	Lecturer	math	123-4562
4	2301	Bill	Associate Profe	CV	987-6543

图 4　教师信息关系实例

4)　实验室信息关系 Laboratory (laboratory_ID, laboratory_NAME, laboratory_addr)

使用 SQL 的 CREATE TABLE 语句建立关系：

```
CREATE TABLE laboratory (
    laboratory_ID CHAR(3) NOT NULL,
    laboratory_NAME CHAR(20) NOT NULL,
    laboratory_addr CHAR(3),
    PRIMARY KEY (laboratory_ID)
);
```

其中，设置 laboratory_ID 为主码，确保实验室的唯一性和非空性；同时设置该关系中的 laboratory_ID，laboratory_NAME 属性不能为空（空值约束）。

而后从文本文件导入数据后查询可视化如上图 3 右所示。

5) 院领导信息关系 Leader (leaderID, LName, Appointment_Date, Ltelephone, L_officce_addr)

使用 SQL 的 CREATE TABLE 语句建立关系：

```
1. CREATE TABLE leader (
2.     leaderID CHAR(4) NOT NULL,
3.     LName VARCHAR(15) NOT NULL,
4.     Appointment_Date CHAR(15),
5.     Ltelephone CHAR(8),
6.     L_officce_addr CHAR(3),
7.     PRIMARY KEY (leaderID)
8. );
```

其中，设置 leaderID 为主码，确保数据的唯一性和非空性；同时设置该关系中的 leaderID，LName 属性不能为空（空值约束）。

而后从文本文件导入数据后，查询可视化如下图 5 所示。

	leaderID char(4)	* LName varchar(15)	Appointment_Dat char(15)	Ltelephone char(8)	L_officce_addr char(3)
1	8001	aaa	9:00-17:00	253-6995	A11
2	8002	bbb	9:00-17:00	965-5596	B22
3	8003	ccc	9:00-17:00	478-5962	C33

图 5 院领导信息关系实例

6) 院系信息关系 department (Department_ID, Dname, leaderID)

使用 SQL 的 CREATE TABLE 语句建立关系：

```
1. CREATE TABLE department (
2.     Department_ID CHAR(3) NOT NULL,
3.     Dname CHAR(30) NOT NULL,
4.     leaderID CHAR(4),
5.     PRIMARY KEY (Department_ID),
6.     FOREIGN KEY (leaderID) REFERENCES leader(leaderID)
7. );
```

其中，设置 Department_ID 为主码，确保院系的唯一性和非空性；同时设置该关系中的 Department_ID，Dname 属性不能为空（空值约束）。

此外，设置 leaderID 是对应于 leader 表的外键，确保数据一致性。

而后从文本文件导入数据后，查询可视化如下图 6 左所示。

Department_I char(3)	* Dname char(30)	leaderID char(4)
101	Math	8001
202	AI	8002
303	CS	8003

class_ID char(8)	student_num int(11)	Department_ID char(3)
20210101	3	101
20210201	4	303
20220302	3	202

图 6 信息关系实例（左图为院系信息关系，右图为班级信息关系）

7) 班级信息关系 Class(class_ID, student_num, Department_ID)

使用 SQL 的 CREATE TABLE 语句建立关系：

```
1. CREATE TABLE Class (
2.     class_ID CHAR(8) NOT NULL,
3.     student_num INT,
4.     Department_ID CHAR(3),
5.     PRIMARY KEY (class_ID),
6.     FOREIGN KEY (Department_ID) REFERENCES department(Department_
ID)
7. );
```

其中，设置 class_ID 为主码，确保班级的唯一性和非空性；同时不额外设置空值约束，因此仅 class_ID 属性不能为空（空值约束）。

此外，设置 Department_ID 是对应于 department 表的外键，确保数据一致性。

而后从文本文件导入数据后，查询可视化如上图 6 右所示。

8) 辅导员信息关系 Student_mentor (Employee_ID, Employee_name, E_Office Hours, E_Office_addr, class_ID)

使用 SQL 的 CREATE TABLE 语句建立关系：

```
1. CREATE TABLE student_mentor (
2.     Employee_ID CHAR(4) NOT NULL,
3.     Employee_name VARCHAR(15) NOT NULL,
4.     Etelephone CHAR(8),
5.     E_Office_Hours CHAR(15),
6.     E_Office_addr CHAR(3),
7.     class_ID CHAR(8),
8.     PRIMARY KEY (Employee_ID),
9.     FOREIGN KEY (class_ID) REFERENCES class(class_ID)
10.);
```

其中，设置职工号 Employee_ID 为主码，确保数据的唯一性和非空性；同时设置该关系中的 Employee_ID,，Employee_name 属性不能为空（空值约束）。

此外，设置 class_ID 是对应于 class 表的外键，确保数据一致性。

而后从文本文件导入数据后，查询可视化如下图 7 所示。

Employee_ID char(4)	* Employee_name varchar(15)	Etelephone char(8)	E_Office_Hours char(15)	E_Office_addr char(3)	class_ID char(8)
9001 char(4)	LeeKnown	253-6998	8:00-19:00	D01	20210101
9002	Faker	965-5594	8:00-19:00	D02	20220302
9003	Kkoma	478-5963	8:00-19:00	D03	20210201

图 7 辅导员信息关系实例

3.2.2 联系建立关系

上述院领导、院系、班级、辅导员实体间的联系都是简单一对一或一对多的联系，因此在建立关系时就已将联系包含之内，不需要额外建立关系。而学生、课程、教师、实验室实体间的联系则需要额外建立关系（**事实上，仅学生与课程之间的多对多联系需要额外建立关系，其余联系可以蕴含在实体对应的关系中**）

1) 学生选课信息关系 S_course (Student_ID, course_ID, grade)

使用 SQL 的 CREATE TABLE 语句建立关系：

```
CREATE TABLE S_course (
    Student_ID CHAR(6) NOT NULL,
    course_ID CHAR(4) NOT NULL,
    grade INT,
    PRIMARY KEY (Student_ID, course_ID),
    FOREIGN KEY (Student_ID) REFERENCES student(Student_ID),
    FOREIGN KEY (course_ID) REFERENCES course(course_ID)
);
```

其中，设置(Student_ID, course_ID)为主码，确保组合唯一性，因此 Student_ID 和 course_ID 属性值均不能为空（空值约束）。

此外，设置 Student_ID 是对应于 student 表的外键，course_ID 是对应于 course 表的外键，确保数据一致性。

而后从文本文件导入数据后，查询可视化如下图 8 左所示。

	Student_ID char(6)	course_I char(4)	grade int(11)
1	123345	1003	85
2	123456	1003	83
3	123456	1004	80
4	123456	1005	80

	Student_ID char(6)	Class_ID char(8)
1	123456	20210101
2	234567	20210101
3	345678	20210101
4	123345	20210201

图 8 信息关系实例（左图为学生选课信息关系，右图为学生所属班级信息关系）

2) 学生所属班级信息关系 S_Class (Student_ID, Class_ID)

使用 SQL 的 CREATE TABLE 语句建立关系：

```
CREATE TABLE S_Class (
    Student_ID CHAR(6) NOT NULL,
    Class_ID CHAR(8) NOT NULL,
    PRIMARY KEY (Student_ID, Class_ID),
    FOREIGN KEY (Student_ID) REFERENCES student(Student_ID),
    FOREIGN KEY (Class_ID) REFERENCES class(class_ID)
);
```

其中，设置(Student_ID, Class_ID)为主码，确保组合唯一性，因此 Student_ID 和 Class_ID 属性值均不能为空（空值约束）。

此外，设置 Student_ID 是对应于 student 表的外键，Class_ID 是对应于 class 表的外键，确保数据一致性。

而后从文本文件导入数据后，查询可视化如上图 8 右所示。

3) 教师授课信息关系 teach (course_ID, teacher_ID)

使用 SQL 的 CREATE TABLE 语句建立关系：

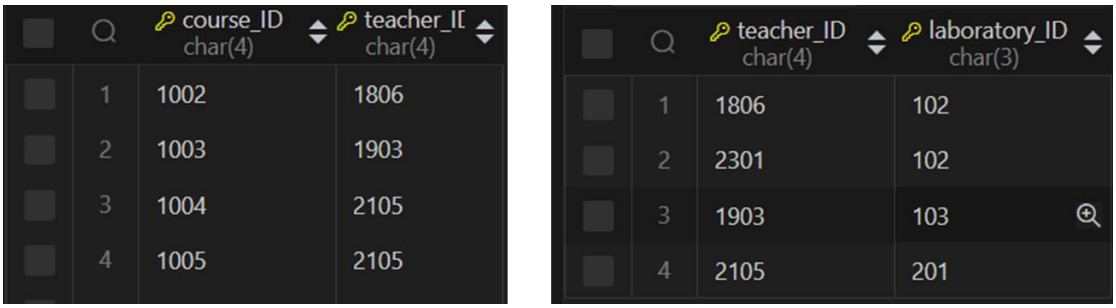
```
CREATE TABLE teach (
    course_ID CHAR(4) NOT NULL,
    teacher_ID CHAR(4) NOT NULL,
```

```
    PRIMARY KEY (course_ID, teacher_ID),
    FOREIGN KEY (course_ID) REFERENCES course(course_ID),
    FOREIGN KEY (teacher_ID) REFERENCES teacher(teacher_ID)
);
```

其中，设置(course_ID, teacher_ID)为主码，确保组合唯一性，因此 course_ID 和 teacher_ID 属性值均不能为空（空值约束）。

此外，设置 course_ID 是对应于 course 表的外键，teacher_ID 是对应于 teacher 表的外键，确保数据一致性。

而后从文本文件导入数据后，查询可视化如下图 9 左所示。



	course_ID char(4)	teacher_ID char(4)
1	1002	1806
2	1003	1903
3	1004	2105
4	1005	2105

	teacher_ID char(4)	laboratory_ID char(3)
1	1806	102
2	2301	102
3	1903	103
4	2105	201

图 9　信息关系实例（左图为教师授课信息关系，右图为教师就职信息关系）

4)　教师就职信息关系 work (teacher_ID, laboratory_ID)

使用 SQL 的 CREATE TABLE 语句建立关系：

```
CREATE TABLE work (
    teacher_ID CHAR(4) NOT NULL,
    laboratory_ID CHAR(3) NOT NULL,
    PRIMARY KEY (teacher_ID, laboratory_ID),
    FOREIGN KEY (teacher_ID) REFERENCES teacher(teacher_ID),
    FOREIGN KEY (laboratory_ID) REFERENCES laboratory(laboratory_
ID)
);
```

其中，设置(teacher_ID, laboratory_ID)为主码，确保组合唯一性，因此 teacher_ID 和 laboratory_ID 属性值均不能为空（空值约束）。

此外，设置 teacher_ID 是对应于 teacher 表的外键，laboratory_ID 是对应于 laboratory 表的外键，确保数据一致性。

而后从文本文件导入数据后，查询可视化如上图 9 右所示。

3.3 常用视图创建

本实验中，针对实际需求对四个常用的查询创建视图如下。

1)　学生成绩视图 s_course_view (Student_ID, Student_NAME, course_name, grade)：包含学号，学生姓名，课程名称，成绩信息

使用 SQL 的 CREATE VIEW 语句建立视图，从关系"S_course"、"student"和"course"的自然连接中选择数据，包括学号 Student_ID、学生姓名 Student_NAME、课程名称 course_name 以及成绩 grade；

对结果进行分组，按照课程编号 course_ID、学号 Student_ID 以及成绩 grade 进行分组；

对分组的结果进行排序，按照课程编号 course_ID 升序排列，成绩 grade 降序排列。

```
1. CREATE VIEW S_course_view
2. AS
3. SELECT S_course.Student_ID,student.Student_NAME,course.course_name,S_course.grade
4. FROM S_course, student, course
5. WHERE S_course.Student_ID=student.Student_ID and S_course.course_ID=course.course_ID
6. GROUP BY S_course.course_ID, S_course.Student_ID, grade
7. ORDER BY S_course.course_ID ASC, grade DESC;
```

查询可视化如下图所示。

	* Student_ID char(6)	* Student_NAME char(20)	* course_name char(15)	grade int(11)
1	789012	Olivia Jones	DeepLearning	75
2	567890	Sophia Miller	OS	100
3	901234	Emma Moore	OS	98
4	678901	James Wilson	OS	95
5	890123	Liam Taylor	CPU_design	85

图 10　学生成绩视图查询

2)　学生的基本信息视图 S_basic_info_view (Student_ID, Student_NAME, age, Class_ID, Dname, telephone, mentor_name)：包含学号，学生，年龄，班级，院系，辅导员，学生联系方式信息。

使用 SQL 的 CREATE VIEW 语句建立视图，使用左外连接将学生表与"s_class"表、"class"表、"student_mentor"表和"department"表连接起来；

对结果进行分组，按照学号 Student_ID、学生姓名 Student_NAME、年龄 age、班号 Class_ID、院系名称 Dname、联系方式 telephone、辅导员姓名 mentor_name 进行分组（每个分组内的这些列的值都是唯一的）。

其中，使用了 TIMESTAMPDIFF 函数计算年份差异，并使用 DATE_FORMAT 函数来比较月份和日期，以确保准确计算年龄；

对分组的结果进行排序，按照班号 Class_ID 升序排列，学号 Student_ID 升序排列。

```
1. CREATE VIEW S_basic_info_view AS
2. SELECT
3.     Student_ID,
4.     Student_NAME,
5.     TIMESTAMPDIFF(YEAR, student.bithday, CURDATE()) - (DATE_FORMAT(CURDATE(), '%m%d') < DATE_FORMAT(student.bithday, '%m%d')) AS age,
```

```
    Class_ID,
    Dname,
    telephone,
    Employee_name AS mentor_name
FROM
    student
    NATURAL LEFT OUTER JOIN s_class
    NATURAL LEFT OUTER JOIN class
    NATURAL LEFT OUTER JOIN student_mentor
    NATURAL LEFT OUTER JOIN department
GROUP BY
    Student_ID,
    Student_NAME,
    age,
    Class_ID,
    Dname,
    telephone,
    mentor_name
ORDER BY
    Class_ID ASC,
    Student_ID ASC;
```

查询可视化如下图所示。

* Student_ID char(6)	* Student_NAME char(20)	age bigint(22)	Class_ID char(8)	Dname char(30)	* telephon char(8)	mentor_name varchar(15)
123456	Alice Johnson	21	20210101	Math	123-4567	LeeKnown
234567	Bob Smith	20	20210101	Math	987-6543	LeeKnown
345678	Emily Davis	16	20210101	Math	111-2222	LeeKnown
123345	Aiden Anderson	18	20210201	CS	888-8888	Kkoma
789012	Olivia Jones	20	20210201	CS	444-4444	Kkoma

图 11　学生基本信息视图查询

3)　教师的基本信息视图 t_basic_info_view (teacher_ID, teacher_name, course_name, Title, ResearchDirection, laboratory_NAME, Ttelephone)：包含职工号，姓名，讲授课程，职称，研究方向，所属实验室，联系方式信息

使用 SQL 的 CREATE VIEW 语句建立视图，使用左外连接将教师表与"teach"表、"course"表、"work"表和"laboratory"表连接起来；

对结果进行分组，按照职工号 teacher_ID、教师姓名 teacher_name、课程名称 course_name、职称 Title、研究方向 ResearchDirection、实验室名称 laboratory_NAME、电话号码 Ttelephone 进行分组（每个分组内的这些列的值都是唯一的）；

对分组的结果进行排序，按照实验室名称 laboratory_NAME 升序排列，研究方向 ResearchDirection 升序排列。

```
CREATE VIEW t_basic_info_view AS
SELECT
    teacher_ID,
    teacher_name,
    course_name,
```

```
    Title,
    ResearchDirection,
    laboratory_NAME,
    Ttelephone
FROM
    teacher
    NATURAL LEFT OUTER JOIN teach
    NATURAL LEFT OUTER JOIN course
    NATURAL LEFT OUTER JOIN work
    NATURAL LEFT OUTER JOIN laboratory
GROUP BY
    teacher_ID,
    teacher_name,
    course_name,
    Title,
    ResearchDirection,
    laboratory_NAME,
    Ttelephone
ORDER BY
    laboratory_NAME ASC,
    ResearchDirection ASC;
```

查询可视化如下图所示。

* teacher_ID char(4)	* teacher_name char(15)	course_name char(15)	Title char(15)	ResearchDirection char(15)	laboratory_NAME char(20)	Ttelephone char(8)
2105	Alan	math2	Lecturer	math	lab1	123-4562
2105	Alan	math1	Lecturer	math	lab1	123-4562
1806	Clark	OS	Assistant Profe	CV	lab2	818-5234
2301	Bill	DeepLearning	Associate Profe	CV	lab2	987-6543
1903	Alice	CPU_design	Professor	CPU	lab3	111-2221

图 12　教师基本信息视图查询

4)　院系的基本信息视图 D_basic_info_view (Department_ID, Dname, LName, Ltelephone)：包含院系编号，名称，系主任，系主任联系方式信息

使用 SQL 的 CREATE VIEW 语句建立视图，使用左外连接将部门表与"leader"表连接起来，以获取部门领导的信息；

对结果进行分组，按照院系编号 Department_ID、名称 Dname、系主任姓名 LName、系主任联系方式 Ltelephone 进行分组（分组内值都是唯一的）。

对分组的结果进行排序，按照院系编号 Department_ID 升序排列。

```
CREATE VIEW D_basic_info_view AS
SELECT
    Department_ID,
    Dname,
    LName,
    Ltelephone
FROM
    department
    NATURAL LEFT OUTER JOIN leader
GROUP BY
```

```
    Department_ID,
    Dname,
    LName,
    Ltelephone
ORDER BY
    Department_ID ASC;
```

查询可视化如下图所示。

	* Department_ID char(3)	* Dname char(30)	LName varchar(15)	Ltelephone char(8)
1	101	Math	aaa	253-6995
2	202	AI	bbb	965-5596
3	303	CS	ccc	478-5962

图 13 院系基本信息视图查询

3.4 索引创建

索引是一种用于提高数据库查询性能的数据结构，它允许数据库系统更快地定位和检索数据。以下是对三种常用的属性（非主键）建立索引。

- 在"department"表的 Dname 列上创建一个升序排序的索引，用于优化按照部门名称的查询。
- 在"course"表的 credits 列上创建一个升序排序的索引，用于提高按照课程学分的检索效率。
- 在"student"表的 birthday 列上创建一个升序排序的索引，用于优化按照学生生日的查询。

```
CREATE INDEX Dname_index ON department(Dname ASC);
CREATE INDEX credits_index ON course(credits ASC);
CREATE INDEX birthday_index ON student(bithday ASC);
```

而后通过 SHOW INDEX 语句对索引进行查询，可以看到除主键外，新建立了 3 个索引。

```
MySQL 5.6 Command Line Cli
mysql> use school_management_system
Database changed
mysql> SHOW INDEX FROM department;
+------------+------------+---------------+--------------+---------------+-----------+-------------+----------+--------+------+------------+---------+---------------+
| Table      | Non_unique | Key_name      | Seq_in_index | Column_name   | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment |
+------------+------------+---------------+--------------+---------------+-----------+-------------+----------+--------+------+------------+---------+---------------+
| department |          0 | PRIMARY       |            1 | Department_ID | A         |           3 |     NULL | NULL   |      | BTREE      |         |               |
| department |          1 | leaderID      |            1 | leaderID      | A         |           3 |     NULL | NULL   | YES  | BTREE      |         |               |
| department |          1 | Dname_index   |            1 | Dname         | A         |           3 |     NULL | NULL   |      | BTREE      |         |               |
+------------+------------+---------------+--------------+---------------+-----------+-------------+----------+--------+------+------------+---------+---------------+
3 rows in set (0.00 sec)

mysql> SHOW INDEX FROM course;
+--------+------------+---------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+
| Table  | Non_unique | Key_name      | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment |
+--------+------------+---------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+
| course |          0 | PRIMARY       |            1 | course_ID   | A         |           5 |     NULL | NULL   |      | BTREE      |         |               |
| course |          1 | credits_index |            1 | credits     | A         |           5 |     NULL | NULL   |      | BTREE      |         |               |
+--------+------------+---------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+
2 rows in set (0.00 sec)

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 |
+---------+------------+----------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+
| student |          0 | PRIMARY        |            1 | Student_ID  | A         |          10 |     NULL | NULL   |      | BTREE      |         |               |
| student |          1 | birthday_index |            1 | bithday     | A         |          10 |     NULL | NULL   | YES  | BTREE      |         |               |
+---------+------------+----------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+
2 rows in set (0.00 sec)
```

图 14 索引查看

3.5 插入删除

对建立的数据库使用 SQL 语言进行插入、删除操作，在插入空值、重复值时需给予提示或警告，以检验表的完整性约束。

3.5.1 插入（包含插入空值和重复值）

使用 SQL INSERT 语句，向数据库中的"student"表插入一条新的学生记录。

```
1. INSERT INTO student (student_id, student_name, bithday, SGender,
telephone)
2. VALUES ('123344', 'John Doe', '2023/01/15', 'Male', '987-5432');
```

在 UI 界面下显示如下，插入成功。

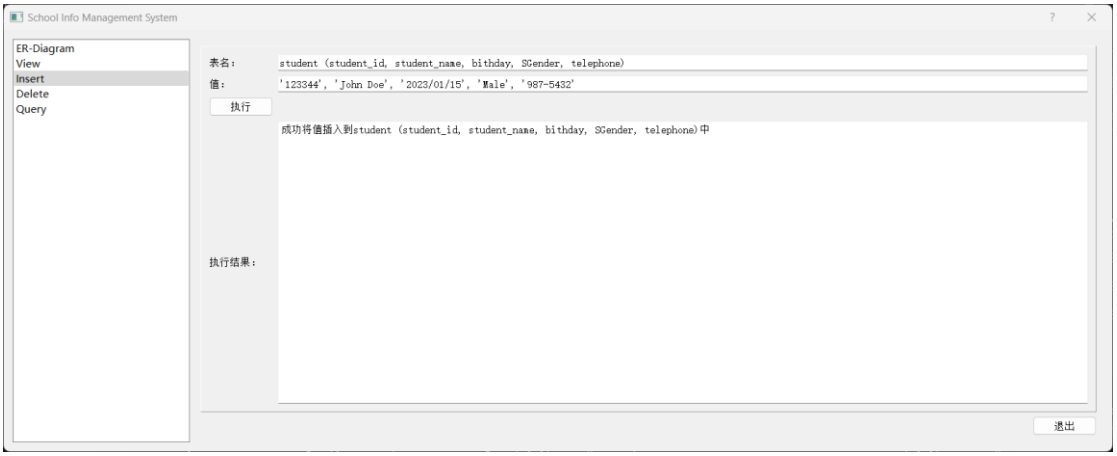


图 15　插入成功运行界面

使用 SQL INSERT 语句，向数据库中的"student"表插入一条新的学生记录，此时设置 student_name 为空。

```
1. INSERT INTO student (student_id, bithday, SGender, telephone)
2. VALUES ('123343','1992-05-20', 'Female', '87654321');
```

在 UI 界面下显示如下，插入失败，原因是 Student_NAME 域并未有默认值。

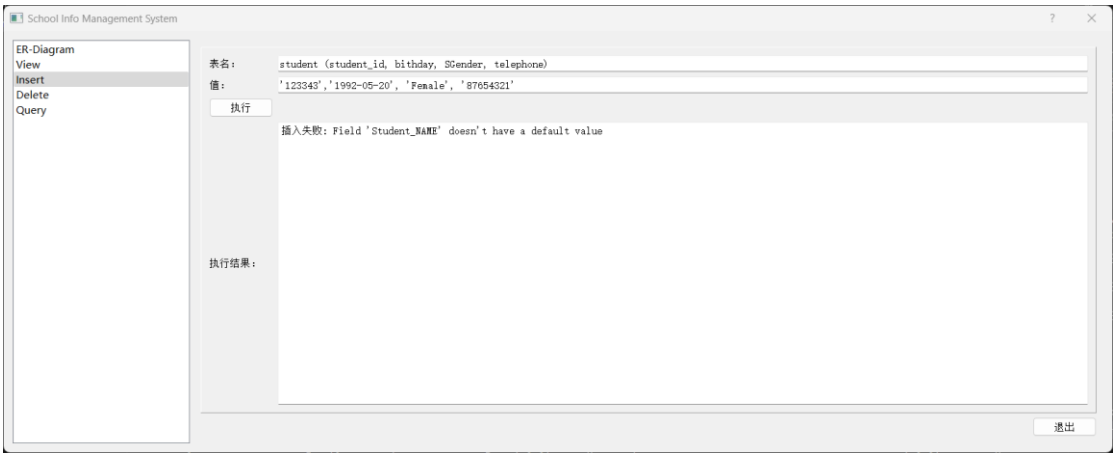


图 16　插入空值运行界面

之后再向数据库中的"student"表插入一条新的学生记录，此时设置 Student_id 与之前插入的数据重复。

```
INSERT INTO student (student_id, student_name, bithday, SGender, telephone)
VALUES ('123344','Jane Smith','1995-03-10','Female', '76543210');
```

在 UI 界面下显示如下，插入失败，原因是 Student_id 为主键不能重复。

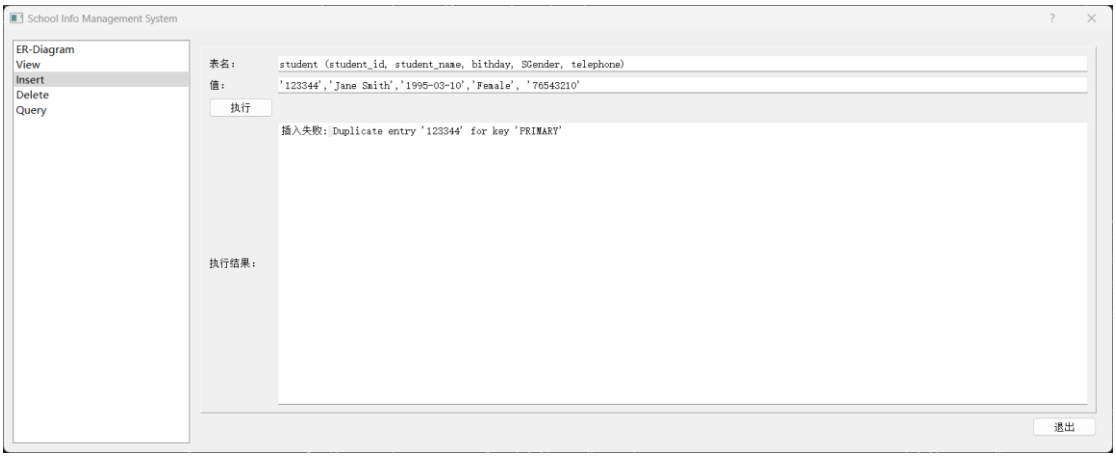


图 17　插入重复值运行界面

运行结束后对比前后"student"表信息，发现仅成功插入一条信息，与本次操作预期结果吻合。

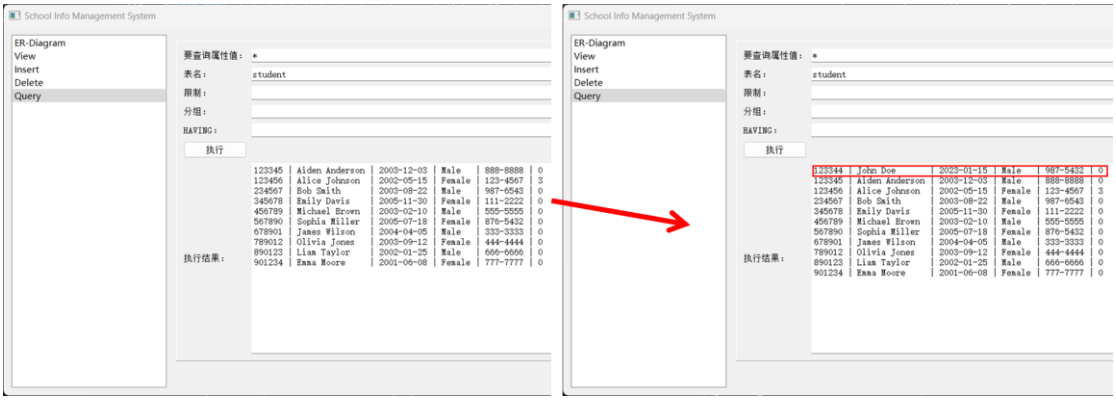


图 18　插入前后数据库信息对比

3.5.2 删除（包含删除空值和不符合外键约束的元组）

使用 SQL DELETE 语句，从"student"表中删除学号为 123344 的学生记录。

```
DELETE FROM student WHERE student_id = '123344';
```

在 UI 界面下显示如下图 19，删除成功。

然后，从"laboratory"表中删除地址为 E33 的实验室记录（实际表中没有满足此条件的元组）。

```
DELETE FROM laboratory WHERE laboratory_addr = 'E33';
```

在 UI 界面下显示如下图 20，未找到匹配的数据，没有对原表进行改动。

最后，从"teacher"表中删除教师职工号为 1806 的教师记录。

```
DELETE FROM teacher WHERE teacher_ID = '1806';
```

在 UI 界面下显示如下图 21，表明在试图删除或更新参考表（teacher）中的行时存在外键约束违规。

具体而言，teach 表中的 teacher_ID 列引用了 teacher 表中的 teacher_ID 列，因此删除不能简单删除"teacher"表中的元组记录。

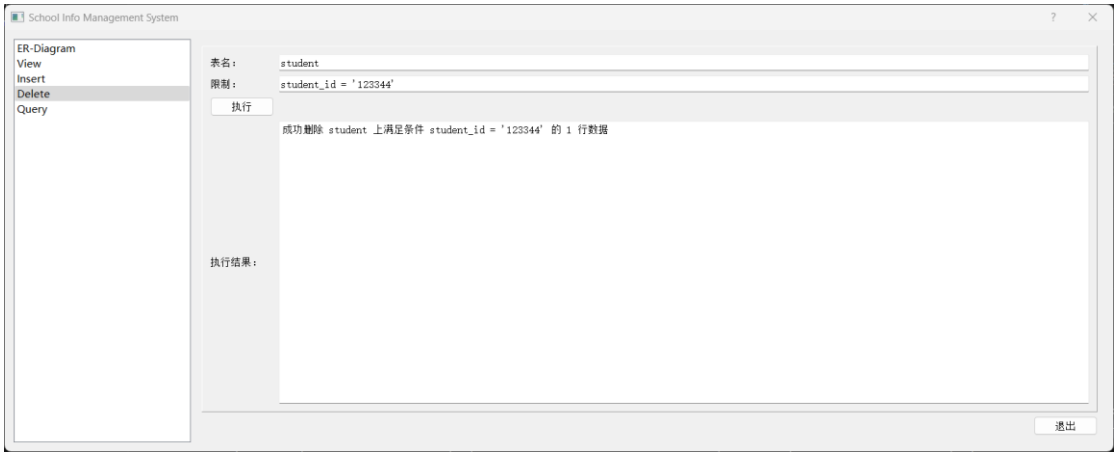


图 19　正常删除运行界面

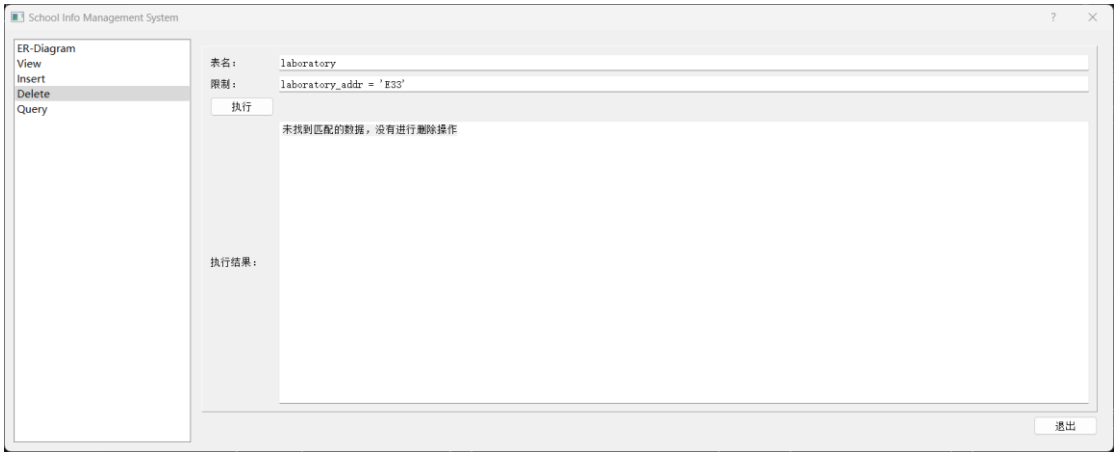


图 20　删除空值运行界面

图 21　删除不满足外键约束运行界面

3.6 查询

对建立的数据库使用 SQL 语句进行连接查询、嵌套查询、分组查询，体现分组、Having 语句进行查询。

1) 连接查询（查看某个班的学生的选课门数）

使用 SQL SELECT 语句，将学生表、学生班级表和选修课程表通过自然左外连接联结在一起，然后根据条件筛选出班级为 20210101 的记录。最后，通过对学生 ID 和姓名进行分组，并使用 COUNT 函数统计每个学生选修的课程数量。

```
1. SELECT s.Student_ID, s.Student_NAME, COUNT(s_course.course_ID) AS CourseCount
2. FROM student s
3. natural left outer join s_class
4. natural left outer join s_course
5. WHERE s_class.Class_ID = '20210101'
6. GROUP BY s.Student_ID, s.Student_NAME;
```

在 UI 界面下显示如下：

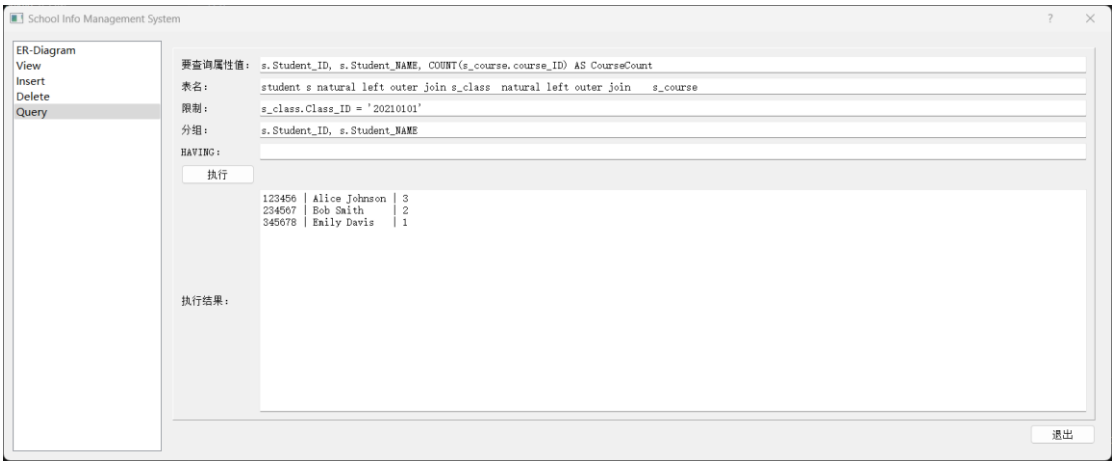


图 22 连接查询运行界面

2) 嵌套查询（查询辅导员“XXX”负责的学生）

使用 SQL SELECT 语句，首先在子查询中找出辅导员名为'Faker'的学生所在的班号，然后在主查询中使用这些班号来获取相应的学号和学生姓名。

```
1. SELECT s.Student_ID, s.Student_NAME
2. FROM student s natural left outer join s_class
3. WHERE class_ID IN (SELECT class_ID FROM student_mentor WHERE Employee_name = 'Faker')
```

在 UI 界面下显示如下：

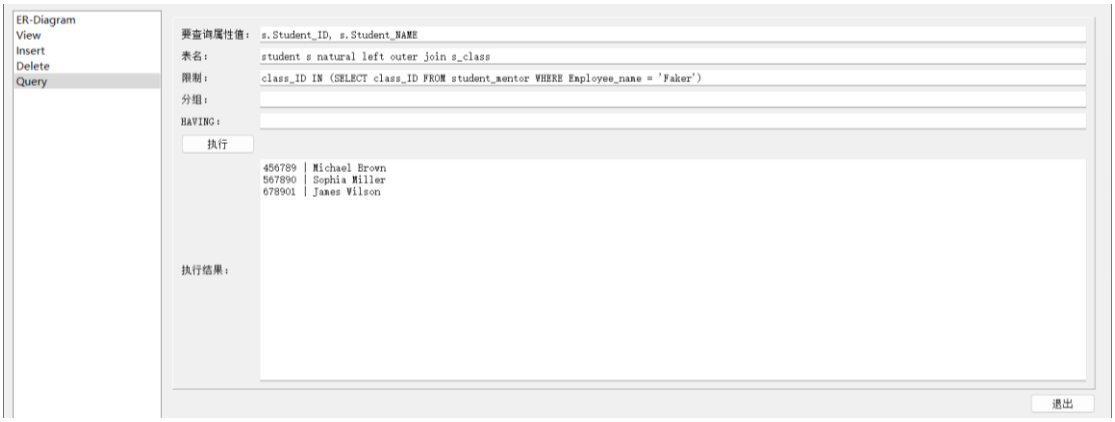


图 23 嵌套查询运行界面

3) 分组查询（查看选修了至少两门课程的学生）

首先通过自然连接将学生表和选修课程表联接在一起，然后按照学号进行分组。最后，使用 HAVING 子句筛选出选修课程数量大于等于 2 的学生记录。

```
SELECT s.Student_ID, COUNT(sc.course_ID) AS CourseCount
FROM student s
natural join s_course sc
GROUP BY s.Student_ID
HAVING CourseCount >= 2;
```

在 UI 界面下显示如下：

图 24 分组查询执行界面

3.7 设置触发器

本实验拟实现的触发器功能为：学生选了某课并且成绩>60 的情况下，总学分=总学分+某课的学分。

因此需要首先在"student"表中添加一个名为 total_credits 的新列，数据类型为整数（INT），并设置默认值为 0，含义为学生修读的总学分。

创建一个触发器，当在 s_course 表中插入新数据时触发。如果新插入的课程成绩大于 60 分，就从 course 表中获取对应课程的学分，然后更新 student 表中相应学生的 total_credits 列，将其增加新选修课程的学分。

```
ALTER TABLE student
ADD COLUMN total_credits INT DEFAULT 0;

CREATE TRIGGER update_total_credits
AFTER INSERT ON s_course
FOR EACH ROW
BEGIN
    DECLARE selected_credits INT;
    SELECT credits INTO selected_credits
    FROM course
    WHERE course_id = NEW.course_id;
    IF NEW.grade > 60 THEN
        UPDATE student
        SET total_credits = total_credits + selected_credits
```

```
        WHERE student_id = NEW.student_id;
    END IF;
END;
```

而后，向"s_course"表内添加数据，触发器即会自动判断是否将学分加入学生当前信息中。例如，对学号为 123456 的同学添加两门课程的成绩，一门 80 分（2 学分），一门 59 分（3 学分）。

```
INSERT INTO s_course (student_id, course_id, grade)
VALUES ('123456', '1005', 80);

INSERT INTO s_course (student_id, course_id, grade)
VALUES ('123456', '1001', 59);
```

观察前后该生学分数对比如下图，观察得 80 分的课程正常加入，而 59 分的课程学分并未计入该生总学分，符合预期实验结果。

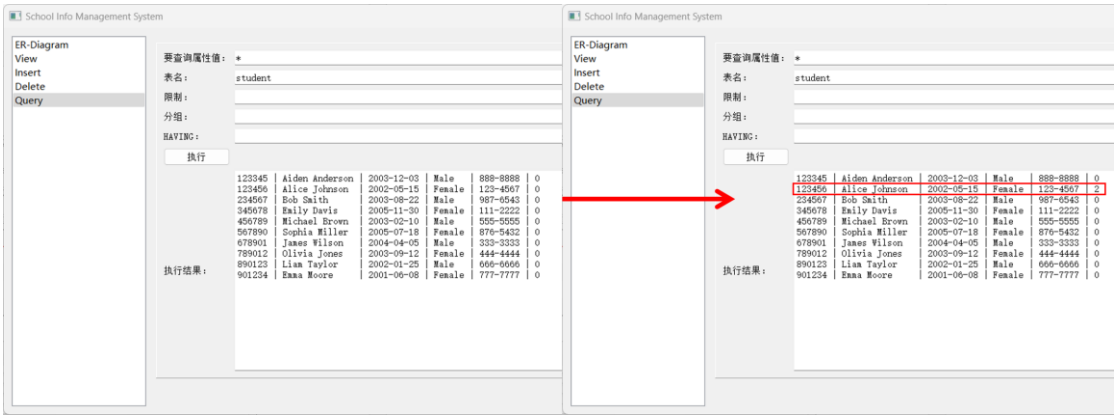


图 25 触发器效果测试

四、实验心得

4.1 问题解决

4.1.1 表格数据输出不对齐问题

实验中使用了“charset gbk”解决了数据输出不对齐的问题。对于数值部分未能完全输出的问题，是因为建立表格时最后一列的属性是 VARCHAR(n)，并且所有的这一列的属性值的长度均未达到 n。更改数据类型为 CHAR 或者重新定义合适的 n 即可解决。

4.2 实验收获

本实验完成了小型学校数据库系统的设计，并基于 PyQt5 和 pymysql 库实现用户可视化界面。首先明确用户需求绘制 E-R 图，然后将 E-R 图转换为等价的关系模式表示的数据库逻辑结构，使用 SQL 语言进行表的创建并设置主键约束、外键约束、空值约束。并基于此数据库完成了以下操作：

a) 对常用的查询创建了 4 个视图，并对 3 个非主键属性设置了索引；

b) 对不同的表格进行插入和删除的操作来验证数据库约束性的正确性。

c) 实现了连接查询、嵌套查询、分组查询来；

d) 实现了触发器功能，当插入新的选课信息后，对学生的总学分进行自动求和计算。

本次实验进一步巩固了概念数据库设计、SQL 语言的使用等知识。进一步加深了对数据库设计相关内容的理解。