



# 华中科技大学

## 数据库系统概论实验报告

姓 名: 李欣宇  
学 院: 网络空间安全学院  
专 业: 信息安全  
班 级: 信安 1901 班  
学 号: U201911658  
指导教师: 王同洋

|      |  |
|------|--|
| 分数   |  |
| 教师签名 |  |

2021 年 11 月 29 日

# 目 录

|                          |           |
|--------------------------|-----------|
| <b>1 课程任务概述.....</b>     | <b>1</b>  |
| <b>2 数据库定义与基本操作.....</b> | <b>2</b>  |
| 2.1 任务要求.....            | 2         |
| 2.2 完成过程.....            | 2         |
| 2.3 任务总结.....            | 5         |
| <b>3 SQL 的复杂操作.....</b>  | <b>7</b>  |
| 3.1 任务要求.....            | 7         |
| 3.2 完成过程.....            | 7         |
| 3.3 任务总结.....            | 12        |
| <b>4 SQL 的高级实验.....</b>  | <b>13</b> |
| 4.1 任务要求.....            | 13        |
| 4.2 完成过程.....            | 13        |
| 4.3 任务总结.....            | 24        |
| <b>5 数据库设计.....</b>      | <b>25</b> |
| 5.1 任务要求.....            | 25        |
| 5.2 完成过程.....            | 25        |
| 5.3 任务总结.....            | 38        |
| <b>6 课程总结.....</b>       | <b>39</b> |
| <b>参考文献.....</b>         | <b>40</b> |

## 1 课程任务概述

本数据库实验课程包含四个实验任务，分别是数据库定义与基本操作(涉及数据库的创建、简单的查询语句)、SQL 的复杂操作(涉及复杂的 SQL 查询、数据的插入和更新)、SQL 的高级实验(涉及权限管理、函数、触发器、存储过程等)、数据库设计(使用高级语言实现一个数据库前端，支持数据库的查询和维护，支持图形界面)

本实验完成的环境：

MacOS 10.15.7

mysql Ver 8.0.23 for macos10.15 on x86\_64

Pycharm 2021.1.3

Navicat 15.0.30

Python 3.9.8

## 2 数据库定义与基本操作

### 2.1 任务要求

#### 2.1.1 实验目的

- (1) 掌握 DBMS 的数据定义功能
- (2) 掌握 SQL 语言的数据定义语句
- (3) 掌握 DBMS 的数据单表查询功能
- (4) 掌握 SQL 语言的数据单表查询语句

#### 2.1.2 实验内容

- (1) 创建数据库
- (2) 创建、删除表
- (3) 查看、修改表的定义
- (4) 理解索引的特点
- (5) 创建和删除索引
- (6) SELECT 语句的基本用法
- (7) 使用 WHERE 子句进行有条件的查询
- (8) 使用 IN, NOT IN, BETWEEN AND 等谓词查询
- (9) 利用 LIKE 子句实现模糊查询
- (10) 利用 ORDER BY 子句为结果排序
- (11) 用 SQL Server/MySQL 的聚集函数进行统计计算
- (12) 用 GROUP BY 子句实现分组查询的方法

#### 2.1.3 实验要求

- (1) 熟练掌握 SQL 的数据定义语句 CREATE、ALTER、DROP、Select

### 2.2 完成过程

完成 mysql 的安装和 navicat 的安装，创建数据库 S\_T\_U201911658,并按照指导书进行基本操作，下述只记录拓展练习内容

#### 2.2.1 查询全体学生的学号、姓名和年龄

SQL 语句: `SELECT Sno,Sname,Ssex,Sage,Sdept FROM Student;`

运行结果:

```
mysql> SELECT Sno,Sname,Sage FROM Student;
```

| Sno       | Sname | Sage |
|-----------|-------|------|
| 200215121 | 李勇    | 20   |
| 200215122 | 刘晨    | 19   |
| 200215123 | 王敏    | 18   |
| 200215125 | 张立    | 19   |

```
4 rows in set (0.00 sec)
```

图 2.1 实验 1 扩展练习 1

### 2.2.2 查询所有计算机系学生的详细记录

SQL 语句: SELECT \* FROM Student WHERE Sdept='CS';

运行结果:

```
mysql> SELECT * FROM Student WHERE Sdept='CS';
```

| Sno       | Sname | Ssex | Sage | Sdept | Scholarship |
|-----------|-------|------|------|-------|-------------|
| 200215121 | 李勇    | 男    | 20   | CS    | 否           |
| 200215122 | 刘晨    | 女    | 19   | CS    | 否           |

```
2 rows in set (0.00 sec)
```

图 2.2 实验 1 扩展练习 2

### 2.2.3 找出考试成绩为优秀或不及格的学生的学号、课程号及成绩

优秀为 90 分及以上

SQL 语句: SELECT Sno,Cno,Grade FROM SC WHERE Grade>=90 OR Grade<60;

运行结果:

```
mysql> SELECT Sno,Cno,Grade FROM SC WHERE Grade>=90 OR Grade<60;
```

| Sno       | Cno | Grade |
|-----------|-----|-------|
| 200215121 | 1   | 92    |
| 200215122 | 2   | 90    |

```
2 rows in set (0.00 sec)
```

图 2.3 实验 1 扩展练习 3

### 2.2.4 查询年龄不在 19~20 岁之间的学生姓名、性别和年龄

SQL 语句: SELECT Sname,Ssex,Sage FROM Student WHERE Sage NOT BETWEEN 19 AND 20;

运行结果:

```
mysql> SELECT Sname,Sage,Ssex FROM Student WHERE Sage NOT BETWEEN 19 AND 20;
```

| Sname | Sage | Ssex |
|-------|------|------|
| 王敏    | 18   | 女    |

```
1 row in set (0.00 sec)
```

图 2.4 实验 1 扩展练习 4

### 2.2.5 查询数学系 (MA)、信息系 (IS)的学生的姓名和所在系

SQL 语句: SELECT Sname,Sdept FROM Student WHERE Sdept in ('MA', 'IS');

运行结果:

```
[mysql> SELECT Sname,Sdept FROM Student WHERE Sdept in('MA','IS');
```

| Sname | Sdept |
|-------|-------|
| 王敏    | MA    |
| 张立    | IS    |

```
2 rows in set (0.00 sec)
```

图 2.5 实验1 扩展练习 5

### 2.2.6 查询名称中包含“数据”的所有课程的课程号、课程名及其学分

SQL 语句: SELECT Cno,Cname,Ccredit FROM Course WHERE Cname LIKE '%数据%';

运行结果:

```
[mysql> SELECT Cno,Cname,Ccredit FROM Course WHERE Cname LIKE '%数据%';
```

| Cno | Cname | Ccredit |
|-----|-------|---------|
| 1   | 数据库   | 4       |
| 5   | 数据结构  | 4       |
| 6   | 数据处理  | 2       |

```
3 rows in set (0.00 sec)
```

图 2.6 实验1 扩展练习 6

### 2.2.7 找出所有没有选修课成绩的学生学号和课程号

SQL 语句: SELECT Sno,Cno FROM SC WHERE Grade is NULL;

运行结果:

```
[mysql> SELECT Sno,Cno FROM SC WHERE Grade is NULL;
```

Empty set (0.00 sec)

图 2.7 实验1 扩展练习 7

### 2.2.8 查询学生 200215121 选修课的最高分、最低分以及平均成绩

SQL 语句: SELECT MAX(Grade),MIN(Grade),AVG(Grade) FROM SC Sno='200215121';

运行结果:

```
[mysql> SELECT MAX(Grade),MIN(Grade),AVG(Grade) FROM SC WHERE Sno='200215121';
```

| MAX(Grade) | MIN(Grade) | AVG(Grade) |
|------------|------------|------------|
| 92         | 85         | 88.3333    |

```
1 row in set (0.01 sec)
```

图 2.8 实验1 扩展练习 8

### 2.2.9 查询选修了 2 号课程的学生的学号及其成绩，查询结果按成绩升序排列

SQL 语句: SELECT Sno,Grade FROM SC WHERE Cno='2' ORDER BY Grade ASC;

运行结果:

```
[mysql> SELECT Sno,Grade FROM SC WHERE Cno='2' ORDER BY Grade ASC;
+-----+-----+
| Sno   | Grade |
+-----+-----+
| 200215121 | 85    |
| 200215122 | 90    |
+-----+-----+
2 rows in set (0.00 sec)
```

图 2.9 实验 1 扩展练习 9

### 2.2.10 查询每个系名及其学生的平均年龄

SQL 语句: SELECT Sdept,AVG(Sage) FROM Student GROUP BY Sdept;

运行结果:

```
[mysql> SELECT Sdept,AVG(Sage) FROM Student GROUP BY Sdept;
+-----+-----+
| Sdept | AVG(Sage) |
+-----+-----+
| CS    | 19.5000   |
| MA    | 18.0000   |
| IS    | 19.0000   |
+-----+-----+
3 rows in set (0.00 sec)
```

图 2.10 实验 1 扩展练习 10

思考: 如何查询学生平均年龄在 19 岁以下 (含 19 岁) 的系别及其学生的平均年龄?

SQL 语句: SELECT Sdept,AVG(Sage) FROM Student GROUP BY Sdept  
HAVING AVG(Sage)<=19;

运行结果:

```
[mysql> SELECT Sdept,AVG(Sage) FROM Student GROUP BY Sdept HAVING AVG(Sage)<=19
+-----+-----+
| Sdept | AVG(Sage) |
+-----+-----+
| MA    | 18.0000   |
| IS    | 19.0000   |
+-----+-----+
2 rows in set (0.01 sec)
```

图 2.11 实验 1 扩展练习 10 思考

## 2.3 任务总结

实验中学习并实践了基本的一些 SQL 语句, 包括建立表、删除表、修改表, 以及以及利用 LIKE 子句实现模糊查询、利用 ORDER BY 子句为结果排序、用 GROUP BY 子句实现分组查询的方法等等。

需要注意的就是书写 SQL 语句是的大小写问题:在实验中发现, MySQL 创

建的数据库名和数据表名默认是字母是小写的，尽管输入的时候可能使用的是大写字母，可见 MySQL 对数据库名和数据表明是不区分字母大小写的，此外对应 SQL 语句中的关键字同样也不区分字母大小写。而对于数据表中的属性，即列名，则是区分大小写的。

此外，聚集函数只能用于“select”子句和“having”子句，不能用于“where”子句，聚集函数的值作为条件时需要放在“having”子句中。

本次实验动手实现了以前只在书上看到和纸上写的 SQL 语句，加深了我的理解。



## 3 SQL 的复杂操作

### 3.1 任务要求

#### 3.1.1. 实验目的

掌握 SQL 语言的数据多表查询语句和更新操作

#### 3.1.2. 实验内容

#### 3.1.2 实验内容

- (1) 等值连接查询（含自然连接查询）与非等值连接查询
- (2) 自身连接查询
- (3) 外连接查询
- (4) 复合条件连接查询
- (5) 嵌套查询（带有 IN 谓词的子查询）
- (6) 嵌套查询（带有比较运算符的子查询）
- (7) 嵌套查询（带有 ANY 或 ALL 谓词的子查询）
- (8) 嵌套查询（带有 EXISTS 谓词的子查询）
- (9) 集合查询
- (10) update 语句用于对表进行更新
- (11) delete 语句用于对表进行删除
- (12) insert 语句用于对表进行插入

#### 3.1.3 实验要求

- (1) 熟练掌握 SQL 的连接查询语句
- (2) 熟练掌握 SQL 的嵌套查询语句
- (3) 掌握表名前缀、别名前缀的用法
- (4) 掌握不相关子查询和相关子查询的区别和用法
- (5) 掌握不同查询之间的等价替换方法（一题多解）及限制
- (6) 熟练掌握 SQL 的数据更新语句 INSERT、UPDATE、DELETE
- (7) 记录实验结果，认真完成实验报告

### 3.2 完成过程

#### 3.2.1 查询每门课程及其被选情况

输出所有课程中每门课的课程号、课程名称、选修该课程的学生学号及成绩  
--如果没有学生选择该课，则相应的学生学号及成绩为空值

SQL 语句：SELECT Course.Cno,Cname,Sno,Grade FROM Course LEFT  
OUTER JOIN SC ON ( Course.Cno=SC.Cno);

运行结果:

```
mysql> SELECT Course.Cno,Cname,Sno,Grade
[ -> FROM Course LEFT OUTER JOIN SC ON (Course.Cno=SC.Cno);
```

| Cno | Cname    | Sno       | Grade |
|-----|----------|-----------|-------|
| 1   | 数据库      | 200215121 | 92    |
| 2   | 数学       | 200215122 | 90    |
| 2   | 数学       | 200215121 | 85    |
| 3   | 信息系统     | 200215122 | 80    |
| 3   | 信息系统     | 200215121 | 88    |
| 4   | 操作系统     | NULL      | NULL  |
| 5   | 数据结构     | NULL      | NULL  |
| 6   | 数据处理     | NULL      | NULL  |
| 7   | PASCAL语言 | NULL      | NULL  |

```
9 rows in set (0.00 sec)
```

图 3.1 实验2 扩展练习 1

### 3.2.2 查询与“张立”同岁的学生的学号、姓名和年龄。

使用至少三种方法:

#### ① 嵌套查询

SQL 语句 1: SELECT Sno,Sname,Sage FROM Student WHERE Sage=  
( SELECT Sage FROM Student WHERE Sname='张立');

运行结果:

```
mysql> SELECT Sno,Sname,Sage
[ -> FROM Student
[ -> WHERE Sage=(SELECT Sage FROM Student WHERE Sname='张立');
```

| Sno       | Sname | Sage |
|-----------|-------|------|
| 200215122 | 刘晨    | 19   |
| 200215125 | 张立    | 19   |

```
2 rows in set (0.00 sec)
```

图 3.2 实验2 扩展练习 2.1

#### ② 自身连接

SQL 语句 2:SELECT s1.Sno,s1.Sname,s1.Sage FROM Student s1, Student s2  
WHERE s1.Sage=s2.Sage AND s2.Sname='张立';

运行结果:

```
mysql> SELECT s1.Sno,s1.Sname,s1.Sage
[ -> FROM Student s1,Student s2
[ -> WHERE s1.Sage=s2.Sage AND s2.Sname='张立';
```

| Sno       | Sname | Sage |
|-----------|-------|------|
| 200215122 | 刘晨    | 19   |
| 200215125 | 张立    | 19   |

```
2 rows in set (0.00 sec)
```

图 3.3 实验2 扩展练习 2.2

#### ③ EXIST 谓词

SQL 语句 3:SELECT Sno,Sname,Sage FROM Student s1 WHERE EXISTS  
(SELECT \* FROM Student s2 WHERE s1.Sage=s2.Sage AND s2.Sname='张立');

运行结果:

```
mysql> SELECT Sno,Sname,Sage
[ -> FROM Student s1
[ -> WHERE EXISTS(SELECT * FROM Student s2 WHERE s1.Sage=s2.Sage AND s2.Sname='张立');
+-----+-----+-----+
| Sno | Sname | Sage |
+-----+-----+-----+
| 200215122 | 刘晨 | 19 |
| 200215125 | 张立 | 19 |
+-----+-----+-----+
2 rows in set (0.01 sec)
```

图 3.4 实验 2 扩展练习 2.3

### 3.2.3 查询选修了 3 号课程且成绩为良好的所有学生的学号和姓名。

成绩良好为 80~89 分

SQL 语句：SELECT SC.Sno,Sname FROM SC,Student WHERE SC.Sno=Student.Sno AND Cno='3' AND Grade BETWEEN 80 AND 89;

运行结果：

```
mysql> SELECT SC.Sno,Student.Sname
[ -> FROM Student,SC
[ -> WHERE Student.Sno=SC.Sno AND Cno='3' AND Grade BETWEEN 80 AND 89;
+-----+-----+
| Sno | Sname |
+-----+-----+
| 200215121 | 李勇 |
| 200215122 | 刘晨 |
+-----+-----+
2 rows in set (0.00 sec)
```

图 3.5 实验 2 扩展练习 3

### 3.2.4 查询学生 200215122 选修的课程号、课程名

SQL 语句：SELECT SC.Cno,Course.Cname FROM SC,Course WHERE Sno='200215122' AND SC.Cno=Course.Cno;

运行结果：

```
mysql> SELECT SC.Cno,Course.Cname
[ -> FROM SC,Course
[ -> WHERE Sno='200215122' AND SC.Cno=Course.Cno;
+-----+-----+
| Cno | Cname |
+-----+-----+
| 2 | 数学 |
| 3 | 信息系统 |
+-----+-----+
2 rows in set (0.00 sec)
```

图 3.6 实验 2 扩展练习 4

思考：如何查询学生 200215122 选修的课程号、课程名及成绩？

SQL 语句：SELECT SC.Cno,Cname,Grade FROM SC,Course WHERE Sno='200215122' AND SC.Cno=Course.Cno;

说明：只需要在“SELECT”子句中加上属性“Grade”。

运行结果：

```
mysql> SELECT SC.Cno,Course.Cname,SC.Grade
[ -> FROM SC,Course
[ -> WHERE Sno='200215122' AND SC.Cno=Course.Cno;
+-----+-----+-----+
| Cno | Cname | Grade |
+-----+-----+-----+
| 2 | 数学 | 90 |
| 3 | 信息系统 | 80 |
+-----+-----+-----+
2 rows in set (0.00 sec)
```

图 3.7 实验 2 扩展练习 4 思考

### 3.2.5 找出每个学生低于他所选修课程平均成绩 5 分以上的课程号

输出学号和课程号

SQL 语句: SELECT Sno,Cno from SC sc1 WHERE Grade+5<(SELECT AVG(Grade) FROM SC sc2 WHERE sc1.Sno=sc2.Sno GROUP BY Sno);

说明: 使用相关子查询, 子查询中计算该学生的平均成绩, 外层查询要求成绩比平均成绩低 5 分。

运行结果:

```
[mysql> SELECT Sno,Cno
[   -> FROM SC sc1
[   -> WHERE Grade+5<(SELECT AVG(Grade) FROM SC sc2 WHERE sc1.Sno=sc2.Sno GROUP BY Sno);
Empty set (0.00 sec)
```

图 3.8 实验 2 扩展练习 5

### 3.2.6 查询比所有男生年龄都小的女生的学号、姓名和年龄

SQL 语句: SELECT Sno,Sname,Sage FROM Student WHERE Ssex='女' AND Sage<ALL (SELECT Sage FROM Student WHERE Ssex='男');

运行结果:

```
[mysql> SELECT Sno,Sname,Sage
[   -> FROM Student
[   -> WHERE Ssex='女' AND Sage < ALL(SELECT Sage FROM Student WHERE Ssex='男 ');
+-----+-----+-----+
| Sno   | Sname | Sage |
+-----+-----+-----+
| 200215123 | 王敏 | 18 |
+-----+-----+-----+
1 row in set (0.00 sec)
```

图 3.9 实验 2 扩展练习 6

### 3.2.7 查询所有选修了 2 号课程的学生姓名及所在系

SQL 语句: SELECT Sname,Sdept FROM Student,SC WHERE Student.Sno=SC.Sno AND Cno='2';

运行结果:

```
[mysql> SELECT Sname,Sdept
[   -> FROM Student,SC
[   -> WHERE Student.Sno=SC.Sno AND Cno='2';
+-----+-----+
| Sname | Sdept |
+-----+-----+
| 李勇  | CS    |
| 刘晨  | CS    |
+-----+-----+
2 rows in set (0.00 sec)
```

图 3.10 实验 2 扩展练习 7

### 3.2.8 使用 update 语句把成绩为良的学生的年龄增加 2 岁, 并查询出来

#### ① UPDATE (更新)

SQL 语句: UPDATE Student SET Sage=Sage+2 where Sno IN (SELECT DISTINCT Sno FROM SC WHERE Grade BETWEEN 80 AND 89);

运行结果:

```

[mysql> UPDATE Student
[   -> SET Sage=Sage+2
[   -> WHERE Sno IN (SELECT DISTINCT Sno FROM SC WHERE Grade BETWEEN 80 AND 89);
Query OK, 2 rows affected (0.01 sec)
Rows matched: 2  Changed: 2  Warnings: 0

```

图 3.11 实验 2 扩展练习 8.1

## ② SELECT (查询)

SQL 语句: SELECT Student.Sno,Sage,Grade FROM Student,SC WHERE Grade BETWEEN 80 AND 89 AND Student.Sno=SC.Sno;

运行结果:

```

[mysql> SELECT Student.Sno,Sage,Grade
[   -> FROM Student,SC
[   -> WHERE Grade BETWEEN 80 AND 89 AND Student.Sno=SC.Sno;
+-----+-----+-----+
| Sno      | Sage | Grade |
+-----+-----+-----+
| 200215121 | 22   | 85    |
| 200215121 | 22   | 88    |
| 200215122 | 21   | 80    |
+-----+-----+-----+
3 rows in set (0.01 sec)

```

图 3.11 实验 2 扩展练习 8.2

### 3.2.9 使用 insert 语句增加两门课程: C 语言和人工智能, 并查询出来

## ① INSERT (插入)

SQL 语句 1: INSERT INTO Course(Cno,Cname) VALUES('8','C 语言'),('9','人工智能');

## ② SELECT (查询)

SQL 语句 2: SELECT Cname FROM Course;

运行结果:

```

[mysql> INSERT INTO Course(Cno,Cname) VALUES('8','C语言'),('9','人工智能');
Query OK, 2 rows affected (0.00 sec)
Records: 2  Duplicates: 0  Warnings: 0

[mysql> SELECT Cname FROM Course;
+-----+
| Cname      |
+-----+
| 数据库     |
| 数学       |
| 信息系统   |
| 操作系统   |
| 数据结构   |
| 数据处理   |
| PASCAL语言 |
| C语言      |
| 人工智能   |
+-----+
9 rows in set (0.00 sec)

```

图 3.12 实验 2 扩展练习 9

### 3.2.10 使用 delete 语句把人工智能课程删除, 并查询出来

## ① DELETE (删除)

SQL 语句 1: DELETE FROM Course WHERE Cname='人工智能';

## ② SELECT (查询)

SQL 语句: SELECT Cname FROM Course;

运行结果:

```
[mysql> DELETE FROM Course WHERE Cname='人工智能';  
Query OK, 1 row affected (0.00 sec)  
  
[mysql> SELECT Cname FROM Course;  
+-----+  
| Cname |  
+-----+  
| 数据库 |  
| 数学 |  
| 信息系统 |  
| 操作系统 |  
| 数据结构 |  
| 数据处理 |  
| PASCAL语言 |  
| C语言 |  
+-----+  
8 rows in set (0.00 sec)
```

图 3.13 实验2 扩展练习 10

## 3.3 任务总结

该部分实验主要实现了 SQL 查询语句中较为复杂的多表查询和嵌套查询, 以及对表的增删改的操作。

本次的实验相对于上次来说更为复杂一些, 查询语句更加的困难了, 尤其是 EXISTS 字句和嵌套查询的使用。对于左连接, 右连接等通过查阅资料也终于区分清楚, 含有“exist”谓词的嵌套查询一般需要将要求转换为“存在”或“不存在”组成的句子, 必要时还需要借助逻辑中的量词、蕴含等对语句进行转换。

同时, 在实验过程中根据指导书的指导和实践发现 Mysql 不支持集合交谓词 “intersect”和 集合差谓词“except”, 但是二者均可以转换为 “where” 条件的运算。

本次实验我实现了很多复杂的查询语句, 同时也进一步掌握了插入和更新语句的用法。

## 4 SQL 的高级实验

### 4.1 任务要求

#### 4.1.1 实验目的

- (1) 掌握 SQL 语言的视图、触发器、存储过程、安全等功能

#### 4.1.2 实验内容

- (1) 创建表的视图
- (2) 利用视图完成表的查询
- (3) 删除表的视图
- (4) 创建触发器
- (5) 创建存储过程
- (6) 对用户进行授权和查询
- (7) 用户定义完整性

#### 4.1.3 实验要求

- (1) 掌握视图的定义与操作
- (2) 掌握对触发器的定义
- (3) 掌握对存储过程的定义
- (4) 掌握如何对用户进行授权和收回权限
- (5) 掌握用户定义完整性的方法
- (6) 写出实验报告

### 4.2 完成过程

#### 4.2.1 创建 CS 系的视图 CS\_View

SQL 语句:CREATE VIEW CS\_View AS SELECT \* FROM Student WHERE Sdept='CS';

说明:使用“CREATE VIEW <视图名> AS <子查询>”语句创建视图

运行结果:可以在数据库左侧视图栏看到对应视图“cs\_view”,如图 4.1



图 4.1 创建视图 CS\_View



#### 4.2.2 在视图 CS\_View 上查询 CS 系选修了 1 号课程的学生

SQL 语句:SELECT CS\_View.Sno,Sname,Ssex,Sage,Sdept,Scholarship,Cno,Grade FROM CS\_View,SC WHERE CS\_View.Sno=SC.Sno and Cno='1';

运行结果:

```
mysql> SELECT CS_View.Sno,Sname,Ssex,Sage,Sdept,Scholarship,Cno,Grade
-> FROM CS_View,SC
-> WHERE CS_View.Sno=SC.Sno AND Cno='1';
```

| Sno       | Sname | Ssex | Sage | Sdept | Scholarship | Cno | Grade |
|-----------|-------|------|------|-------|-------------|-----|-------|
| 200215121 | 李勇    | 男    | 22   | CS    | 否           | 1   | 92    |

1 row in set (0.00 sec)

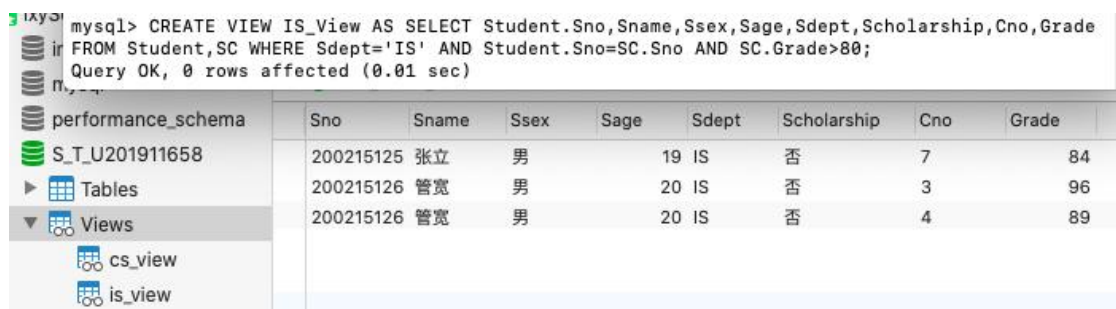
图 4.2 运行结果

#### 4.2.3 创建 IS 系成绩大于 80 的学生的视图 IS\_View

SQL 语句:CREATE VIEW IS\_View AS SELECT Student.Sno, Sname, Ssex, Sage, Sdept, Scholarship, Cno, Grade FROM Student,SC WHERE Sdept='IS' AND Student.Sno=SC.Sno AND SC.Grade>80;

运行结果:

```
mysql> CREATE VIEW IS_View AS SELECT Student.Sno,Sname,Ssex,Sage,Sdept,Scholarship,Cno,Grade
FROM Student,SC WHERE Sdept='IS' AND Student.Sno=SC.Sno AND SC.Grade>80;
Query OK, 0 rows affected (0.01 sec)
```



| Sno       | Sname | Ssex | Sage | Sdept | Scholarship | Cno | Grade |
|-----------|-------|------|------|-------|-------------|-----|-------|
| 200215125 | 张立    | 男    | 19   | IS    | 否           | 7   | 84    |
| 200215126 | 管宽    | 男    | 20   | IS    | 否           | 3   | 96    |
| 200215126 | 管宽    | 男    | 20   | IS    | 否           | 4   | 89    |

图 4.3 创建视图 IS\_View

#### 4.2.4 创建 IS 系成绩大于 80 的学生的视图 IS\_View

SQL 语句:SELECT DISTINCT Sname FROM IS\_View;

运行结果:

```
mysql> SELECT DISTINCT Sname FROM IS_View;
```

| Sname |
|-------|
| 张立    |
| 管宽    |

2 rows in set (0.00 sec)

图 4.4 运行结果

#### 4.2.5 删除视图

SQL 语句:DROP VIEW IS\_View;

运行结果:



```
[mysql> DROP VIEW IS_View;  
Query OK, 0 rows affected (0.01 sec)
```

图 4.5 运行结果

#### 4.2.6 授权的相关操作

(1) 利用可视化窗口创建 2 个不同的用户 U1 和 U2,利用系统管理员给 U1 授予 Student 表的查询和更新的权限, 给 U2 对 SC 表授予插入的权限。

SQL 语句:

授权 U1:grant select,update on table student to U1@localhost;

授权 U2:grant insert on table SC to U2@localhost;

运行结果:

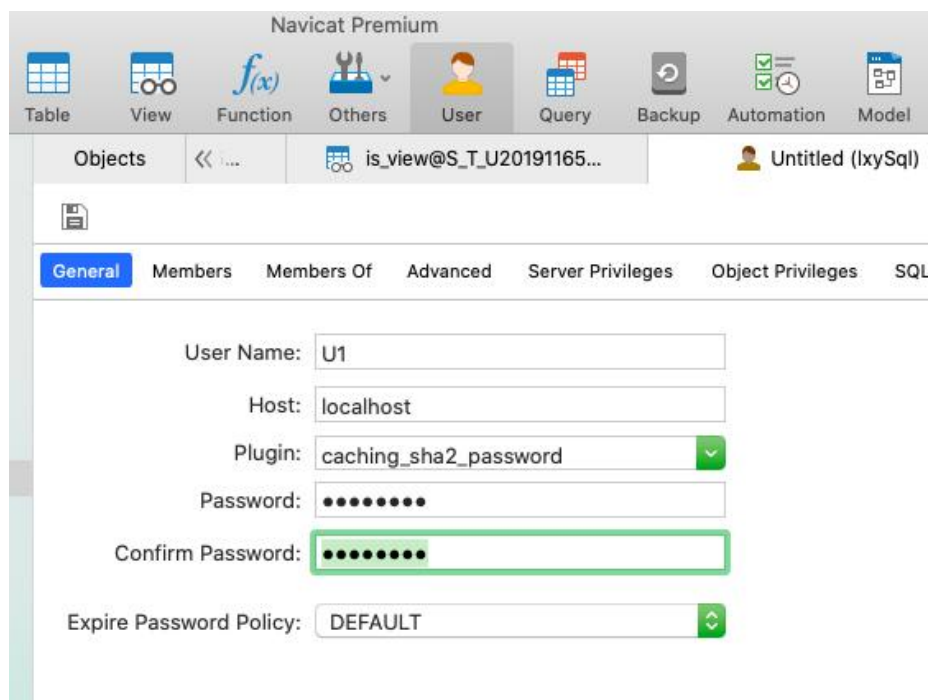


图 4.6 创建用户 U1

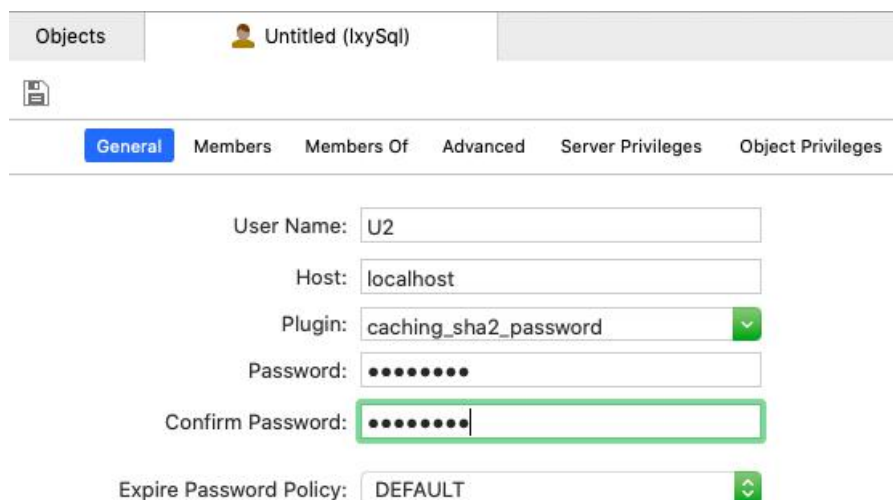


图 4.7 创建用户 U2

```

[mysql> grant select,update on table Student to U1@localhost;
Query OK, 0 rows affected (0.01 sec)

[mysql> grant insert on table SC to U2@localhost;
Query OK, 0 rows affected (0.01 sec)

```

图 4.8 给 U1 和 U2 授权

(2) 用 U1 登录, 查询学生表的信息;

SQL 语句:SELECT \* FROM Student;

运行结果:

```

[mysql> select user();
+-----+
| user() |
+-----+
| U1@localhost |
+-----+
1 row in set (0.00 sec)

[mysql> use S_T_U201911658
Database changed
[mysql> SELECT * FROM Student;
+-----+-----+-----+-----+-----+-----+
| Sno      | Sname  | Ssex  | Sage  | Sdept  | Scholarship |
+-----+-----+-----+-----+-----+-----+
| 200215121 | 李勇   | 男    | 22    | CS     | 否          |
| 200215122 | 刘晨   | 女    | 21    | CS     | 否          |
| 200215123 | 王敏   | 女    | 18    | MA     | 否          |
| 200215125 | 张立   | 男    | 19    | IS     | 否          |
| 200215126 | 管宽   | 男    | 20    | IS     | 否          |
| 200215127 | 秦冉   | 女    | 19    | MA     | 否          |
| 200215128 | 李欣宇 | 女    | 20    | CSE    | 否          |
+-----+-----+-----+-----+-----+-----+
7 rows in set (0.00 sec)

```

图 4.9 运行结果

(3) 用 U1 登录, 把所有学生的年龄增加 1 岁, 然后查询;

SQL 语句:

UPDATE Student SET Sage=Sage+1;

SELECT \* FROM Student;

运行结果:

```

[mysql> UPDATE Student SET Sage=Sage+1;
Query OK, 7 rows affected (0.00 sec)
Rows matched: 7  Changed: 7  Warnings: 0

[mysql> SELECT * FROM Student;
+-----+-----+-----+-----+-----+-----+
| Sno      | Sname  | Ssex  | Sage  | Sdept  | Scholarship |
+-----+-----+-----+-----+-----+-----+
| 200215121 | 李勇   | 男    | 23    | CS     | 否          |
| 200215122 | 刘晨   | 女    | 22    | CS     | 否          |
| 200215123 | 王敏   | 女    | 19    | MA     | 否          |
| 200215125 | 张立   | 男    | 20    | IS     | 否          |
| 200215126 | 管宽   | 男    | 21    | IS     | 否          |
| 200215127 | 秦冉   | 女    | 20    | MA     | 否          |
| 200215128 | 李欣宇 | 女    | 21    | CSE    | 否          |
+-----+-----+-----+-----+-----+-----+
7 rows in set (0.00 sec)

```

图 4.10 运行结果

(4) 用 U1 登录, 删除 IS 系的学生;

SQL 语句:DELETE FROM Student WHERE Sdept='IS';

说明: 由于用户 U1 没有删除 Student 表的权限, 因此无法执行。

运行结果:

```
mysql> DELETE FROM Student WHERE Sdept='IS';
ERROR 1142 (42000): DELETE command denied to user 'U1'@'localhost' for table 'student'
```

图 4.11 运行结果

(5) 查询 CS 系的选课信息。

SQL 语句:SELECT \* FROM SC,student WHERE Student.Sdept='CS' AND Student.Sno=SC.Sno;

说明: 由于用户 U1 没有对表 sc 的权限, 因此查询失败。

运行结果:

```
mysql> SELECT * FROM SC,Student WHERE Student.Sdept='CS' AND Student.Sno=SC.Sno;
ERROR 1142 (42000): SELECT command denied to user 'U1'@'localhost' for table 'sc'
```

图 4.12 运行结果

(6) 用 U2 登录, 在 SC 表中插入 1 条记录 ('200215122', '1', 75)

SQL 语句:INSERT INTO SC VALUES ('200215122','1',75);

运行结果:

```
mysql> select user();
+-----+
| user() |
+-----+
| U2@localhost |
+-----+
1 row in set (0.00 sec)

mysql> use S_T_U201911658
Database changed
mysql> INSERT INTO SC VALUES('200215122','1',75);
Query OK, 1 row affected (0.01 sec)
```

| a | Sno       | Cno | Grade |
|---|-----------|-----|-------|
|   | 200215121 | 1   | 92    |
|   | 200215121 | 2   | 85    |
|   | 200215121 | 3   | 88    |
|   | 200215122 | 1   | 75    |
|   | 200215122 | 2   | 90    |
|   | 200215122 | 3   | 80    |
|   | 200215123 | 4   | 90    |
|   | 200215125 | 4   | 56    |
|   | 200215125 | 5   | 75    |
|   | 200215125 | 7   | 84    |
|   | 200215126 | 3   | 96    |
|   | 200215126 | 4   | 89    |

图 4.13 运行结果

(7) 用 U2 登录, 查询 SC 表的信息

SQL 语句:SELECT \* FROM SC;

说明: 用户 U2 仅有对表 SC 的插入权限, 没有查询权限, 因此执行失败。

运行结果:

```
[mysql> SELECT * FROM SC;
ERROR 1142 (42000): SELECT command denied to user 'U2'@'localhost' for table 'sc'
```

图 4.14 运行结果

(8) 用 U2 登录, 查询视图 CS\_View 的信息。

SQL 语句:SELECT \* FROM CS\_View;

说明: 用户 U2 没有对视图 CS\_View 的权限, 因此无法查询。

运行结果:

```
[mysql> SELECT * FROM CS_View;
ERROR 1142 (42000): SELECT command denied to user 'U2'@'localhost' for table 'cs_view'
mysql> |
```

图 4.15 运行结果

#### 4.2.7 用系统管理员登录, 收回 U1 的所有权限

SQL 语句:revoke select,update on Student from U1@localhost;

运行结果:

```
[mysql> select user();
+-----+
| user() |
+-----+
| root@localhost |
+-----+
1 row in set (0.00 sec)

[mysql> use S_T_U201911658;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
[mysql> revoke select,update on Student from U1@localhost;
Query OK, 0 rows affected (0.00 sec)
```

图 4.16 运行结果

#### 4.2.8 用 U1 登录, 查询学生表的信息

SQL 语句:use S\_T\_U201911658;

说明: 由于用户 U1 仅有对表 Student 的权限, 当权限撤回后 U1 对该数据库中的表均没有了权限, 无法访问该数据库, 因此选择使用 S\_T\_U201911658 数据库时就已经被拒绝了。

运行结果:

```
[mysql> select user();
+-----+
| user() |
+-----+
| U1@localhost |
+-----+
1 row in set (0.00 sec)

[mysql> use S_T_U201911658;
ERROR 1044 (42000): Access denied for user 'U1'@'localhost' to database 's_t_u201911658'
```

图 4.17 运行结果

#### 4.2.9 对 SC 表建立更新触发器并进行相关操作

对 SC 表建立一个更新触发器，当更新了 SC 表的成绩时，如果更新后的成绩大于等于 95，则检查该成绩的学生是否有奖学金，如果奖学金是“否”，则修改为“是”。如果修改后的成绩小于 95，则检查该学生的其他成绩是不是有大于 95 的，如果都没有，且修改前的成绩是大于 95 时，则把其奖学金修改为“否”。然后进行成绩修改，并进行验证是否触发器正确执行。

SQL 语句:

```
CREATE TRIGGER sc_t
AFTER UPDATE ON sc
FOR EACH ROW
BEGIN
    IF(new.Grade>=95) THEN
        UPDATE Student SET Scholarship='是'
        WHERE Student.Sno=new.Sno AND Scholarship='否';
    ELSE
        UPDATE Student SET Scholarship='否'
        WHERE Student.Sno=new.Sno AND old.Grade>=95 AND
        NOT EXISTS(SELECT * FROM SC WHERE SC.Sno=new.Sno
AND Grade>=95);
    END IF;
END;
```

运行结果:

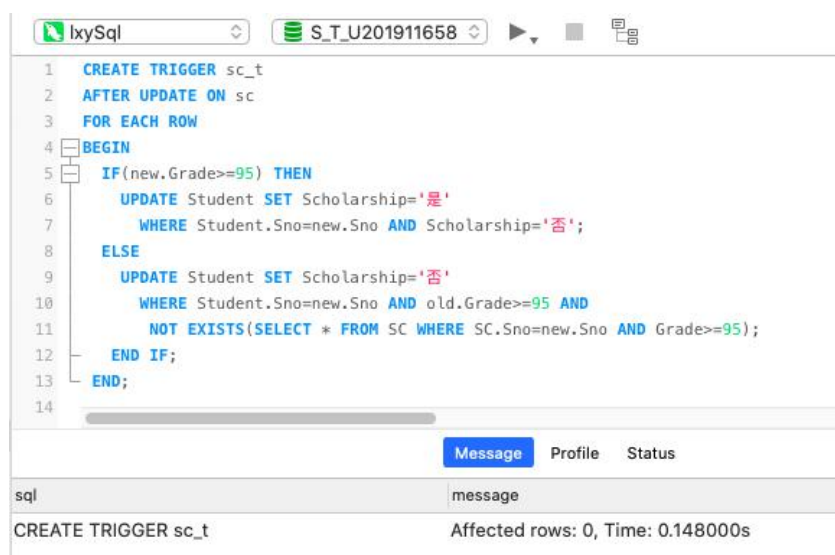


图 4.18 运行结果

(2) 首先把某个学生成绩修改为 98，查询其奖学金。

下述图 4.19 为未修改前的 SC 表和 Student 表，以供对照，要修改的学生为学号 200215121，课程号为 1 的成绩（即 SC 表第一行）

| Sno       | Cno | Grade |
|-----------|-----|-------|
| 200215121 | 1   | 92    |
| 200215121 | 2   | 85    |
| 200215121 | 3   | 88    |
| 200215122 | 1   | 75    |
| 200215122 | 2   | 90    |
| 200215122 | 3   | 80    |
| 200215123 | 4   | 90    |
| 200215125 | 4   | 56    |
| 200215125 | 5   | 75    |
| 200215125 | 7   | 84    |
| 200215126 | 3   | 93    |
| 200215126 | 4   | 89    |

| Sno       | Sname | Ssex | Sage | Sdept | Scholarship |
|-----------|-------|------|------|-------|-------------|
| 200215121 | 李勇    | 男    | 23   | CS    | 否           |
| 200215122 | 刘晨    | 女    | 22   | CS    | 否           |
| 200215123 | 王敏    | 女    | 19   | MA    | 否           |
| 200215125 | 张立    | 男    | 20   | IS    | 否           |
| 200215126 | 管宽    | 男    | 21   | IS    | 否           |
| 200215127 | 秦冉    | 女    | 20   | MA    | 否           |
| 200215128 | 李欣宇   | 女    | 21   | CSE   | 否           |

图 4.19 未修改的 SC 表和 Student 表

SQL 语句：UPDATE SC SET Grade=98 WHERE Sno='200215121' AND Cno='1';

说明：将 sno=200215122, cno=1 的成绩改为 98, 原为 92, 可以看到在 Student 表中该学生的 scholarship 属性已经自动改为 ‘是’

运行结果：

```
mysql> UPDATE SC SET Grade=98 WHERE Sno='200215121' AND Cno='1';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1 Changed: 1 Warnings: 0

mysql>
```

| Sno       | Sname | Ssex | Sage | Sdept | Scholarship |
|-----------|-------|------|------|-------|-------------|
| 200215121 | 李勇    | 男    | 23   | CS    | 是           |

| Sno       | Cno | Grade |
|-----------|-----|-------|
| 200215121 | 1   | 98    |

图 4.19 改为 98

(3) 再把刚才的成绩修改为 80, 再查询其奖学金。

SQL 语句：UPDATE SC SET Grade=80 WHERE Sno='200215121' AND Cno='1';

说明：运行后使用 navicat 查看发现 Student 表中该学生的 scholarship 属性变为了 ‘否’

运行结果：



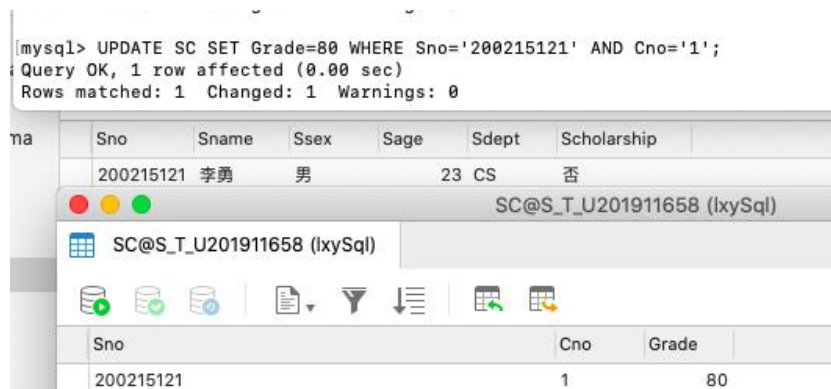


图 4.20 改为 80

#### 4.2.10 删除刚定义的触发器

SQL 语句: DROP TRIGGER sc\_t;

运行结果:

```
[mysql> DROP TRIGGER sc_t;
Query OK, 0 rows affected (0.01 sec)
```

图 4.21 运行结果

4.2.11 定义一个带学号为参数的查看某个学号的所有课程的成绩，查询结果要包含学生姓名。进行验证。

SQL 语句:

CREATE PROCEDURE getCSAvgMax ()

BEGIN

SELECT SC. Sno, AVG(Grade) ,MAX(Grade) FROM SC, Student  
WHERE Student. Sno=SC.Sno AND Sdept='CS' GROUP BY Sno;

END;

CALL getCSAvgMax();

运行结果:

|   |            |            |
|---|------------|------------|
| 1 CREATE PROCEDURE getCSAvgMax()                        |            |            |
| 2 BEGIN   |            |            |
| 3 SELECT SC.Sno,AVG(Grade),MAX(Grade) FROM SC,Student   |            |            |
| 4 WHERE Student.Sno=SC.Sno AND Sdept='CS' GROUP BY Sno; |            |            |
| 5 END;  |            |            |
| 6 CALL getCSAvgMax();                                   |            |            |
| Message Result 1 Profile Status                         |            |            |
| Sno   | AVG(Grade) | MAX(Grade) |
| 200215121   | 84.3333    | 88         |
| 200215122   | 81.6667    | 90         |

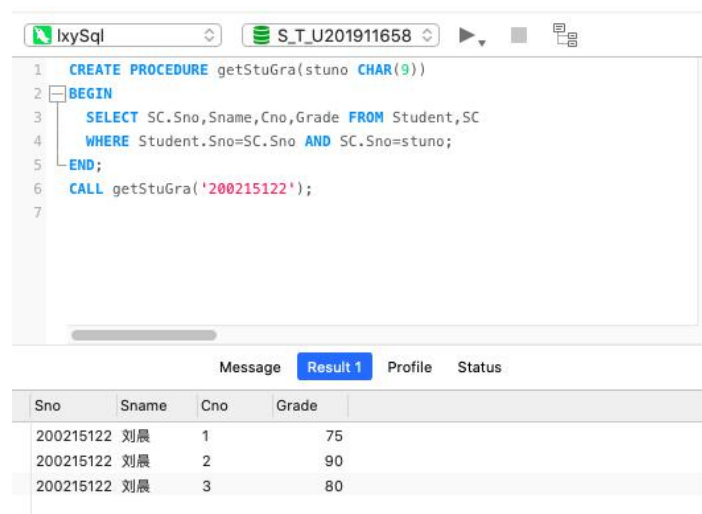
图 4.22 运行结果

4.2.12 定义一个带学号为参数的查看某个学号的所有课程的成绩，查询结果要包含学生姓名。进行验证。

SQL 语句:

```
CREATE PROCEDURE getStuGra(stuno CHAR(9))
BEGIN
    SELECT SC.Sno,Sname,Cno,Grade FROM Student,SC
    WHERE Student.Sno=SC.Sno AND SC.Sno=stuno;
END;
CALL getStuGra('200215122');
```

运行结果:



The screenshot shows a SQL IDE window with a query editor and a results pane. The query editor contains the following SQL code:

```
1 CREATE PROCEDURE getStuGra(stuno CHAR(9))
2 BEGIN
3     SELECT SC.Sno,Sname,Cno,Grade FROM Student,SC
4     WHERE Student.Sno=SC.Sno AND SC.Sno=stuno;
5 END;
6 CALL getStuGra('200215122');
```

The results pane shows the output of the query, which is a table with the following data:

| Sno       | Sname | Cno | Grade |
|-----------|-------|-----|-------|
| 200215122 | 刘晨    | 1   | 75    |
| 200215122 | 刘晨    | 2   | 90    |
| 200215122 | 刘晨    | 3   | 80    |

图 4.23 运行结果

4.2.13 把上一题改成函数。再进行验证。

SQL 语句:

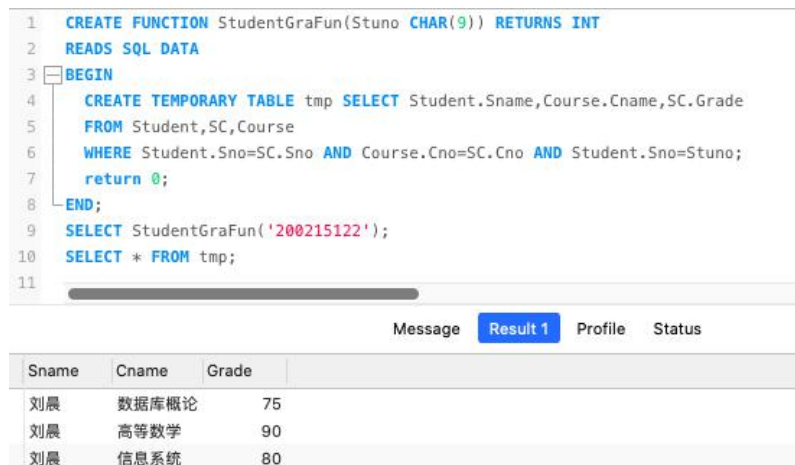
```
CREATE FUNCTION StudentGraFun(Stuno CHAR(9)) RETURNS INT
READS SQL DATA
BEGIN
    CREATE TEMPORARY TABLE tmp
    SELECT Student.Sno,Student.Sname,Course.Cno,Course.Cname,SC.Grade
    FROM Student,SC,Course
    WHERE Student.Sno=SC.Sno AND Course.Cno=SC.Cno
    AND Student.Sno=Stuno;
    return 0;
END;
SELECT StudentGraFun('200215122');
```

说明：由于 Mysql 中函数的返回值只能是一个基本类型，即一行一列的表，所以不能使用返回值输出，只能创建一个虚表，把查询到的结果存放到虚表中，



查询虚表输出结果

运行结果:



```
1 CREATE FUNCTION StudentGraFun(Stuno CHAR(9)) RETURNS INT
2 READS SQL DATA
3 BEGIN
4     CREATE TEMPORARY TABLE tmp SELECT Student.Sname, Course.Cname, SC.Grade
5     FROM Student, SC, Course
6     WHERE Student.Sno=SC.Sno AND Course.Cno=SC.Cno AND Student.Sno=Stuno;
7     return 0;
8 END;
9 SELECT StudentGraFun('200215122');
10 SELECT * FROM tmp;
```

| Sname | Cname | Grade |
|-------|-------|-------|
| 刘晨    | 数据库概论 | 75    |
| 刘晨    | 高等数学  | 90    |
| 刘晨    | 信息系统  | 80    |

图 4.24 运行结果

#### 4.2.14 完整性约束

在 SC 表上定义一个完整性约束，要求成绩在 0-100 之间。定义约束前，先把某个学生的成绩修改成 120，进行查询，再修改回来。定义约束后，再把该学生成绩修改为 120，然后进行查询。

(1) 定义约束前:

说明：将学号为 ‘200215121’ 的学生，课程号为 ‘1’ 的成绩改为 120，然后再改成原本的 89

SQL 语句及运行结果:

```
[mysql> UPDATE SC SET Grade=120
  -> WHERE Sno='200215121' AND Cno='1';
Query OK, 0 rows affected (0.00 sec)
Rows matched: 1  Changed: 0  Warnings: 0

[mysql> SELECT * FROM SC WHERE Sno='200215121' AND Cno='1';
+-----+-----+-----+
| Sno   | Cno  | Grade |
+-----+-----+-----+
| 200215121 | 1    | 120   |
+-----+-----+-----+
1 row in set (0.00 sec)

[mysql> UPDATE SC SET Grade=89
  -> WHERE Sno='200215121' AND Cno='1';
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

[mysql> SELECT * FROM SC WHERE Sno='200215121' AND Cno='1';
+-----+-----+-----+
| Sno   | Cno  | Grade |
+-----+-----+-----+
| 200215121 | 1    | 89    |
+-----+-----+-----+
1 row in set (0.00 sec)
```

图 4.25 运行结果

(1) 定义约束后:

定义约束的 SQL 语句: ALTER TABLE SC ADD CONSTRAINT graRange

check(Grade BETWEEN 0 AND 100);

说明：定义约束后，将学号为‘200215121’的学生，课程号为‘1’的成绩更新为 120，发现拒绝更新。

SQL 语句及运行结果：

```
[mysql> ALTER TABLE SC ADD CONSTRAINT graRange check(Grade BETWEEN 0 AND 100);
Query OK, 12 rows affected (0.08 sec)
Records: 12 Duplicates: 0 Warnings: 0

[mysql> UPDATE SC SET Grade=120
-> WHERE Sno='200215121' AND Cno='1';
ERROR 3819 (HY000): Check constraint 'graRange' is violated.
[mysql> SELECT * FROM SC WHERE Sno='200215121' AND Cno='1';
+-----+-----+-----+
| Sno      | Cno | Grade |
+-----+-----+-----+
| 200215121 | 1   | 89    |
+-----+-----+-----+
1 row in set (0.00 sec)
```

图 4.26 运行结果

### 4.3 任务总结

实现了一些高级操作，比如创建视图，授权，触发器，存储过程，函数，完整性约束等等。

其中对于触发器的印象比较深刻，之前在命令行中实现这两个功能的时候，发现有些语法错误，查资料发现原来是不同 SQL 数据库的语法略有不同，比如触发器语句中没有“referencing”部分，而是直接对“new”和“old”使用；已经删除触发器时不需要在语句后用“on”表明所属数据表等，通过查阅资料修改语法，最后完成了实现。

值得一提的还有函数部分，因为函数只能返回基本类型，起初我是使用了 group\_concat 函数对学生信息进行了拼接，但是还是只能输出一行一列的信息，后来通过老师指点发现可以创建虚表，将结果存入虚表，最终通过查询虚表将结果输出出来。

## 5 数据库设计

### 5.1 任务要求

#### 5.1.1 实验目的

掌握数据库设计和开发技巧

#### 5.1.2 实验内容

通过一个数据库具体设计实例，掌握数据库设计的方法。

#### 5.1.3 实验要求

熟练掌握使用 SQL 语句设计数据库的方法，实现前述实验的学生管理系统，完成实验报告。

#### 5.1.4 系统功能要求

- (1) 新生入学信息增加，学生信息修改。
- (2) 课程信息维护（增加新课程，修改课程信息，删除没有选课的课程信息）。
- (3) 录入学生成绩，修改学生成绩。
- (4) 按系统统计学生的平均成绩、最好成绩、最差成绩、优秀率、不及格人数。
- (5) 按系对学生成绩进行排名，同时显示出学生、课程和成绩信息。
- (6) 输入学号，显示该学生的基本信息和选课信息。

### 5.2 完成过程

#### 5.2.1 实验环境准备

使用 python 3.9.8 在 pycharm 中进行编程，连接 mysql 使用 mysql-connector-python

使用\$ pip3 install mysql-connector-python

导入库为 import mysql.connector

图形化界面调用 python 的 tkinter 库

#### 5.2.2 相关函数说明

- (1) 连接数据库

```
cnx=mysql.connector.connect(user=username,  
                             password=password,  
                             host='localhost',  
                             database='S_T_U201911658')
```

调用该函数可连接数据库，并返回一个数据库对象 cnx（自定义），user 为用户名，password 为密码，database 为要连接的数据库，host 为主机名，如果连接失败会返回 mysql.connector.Error

(2)获取数据库游标

```
cursor = cnx.cursor()
```

调用该函数可以反悔数据库对象 cnx 的一个游标 cursor，利用该游标可以对数据库进行相关操作

(3) 执行 sql 语句

```
cursor.execute(query)
```

query 为要执行的 sql 语句，利用该函数可以执行 sql 语句，执行后的返回值均存于 cursor 中，可以遍历 cursor 中取出查询值

(4) 提交操作

```
cnx.commit()
```

执行完增删改等 sql 语句后使用该函数进行事务提交

### 5.2.3 ER 图和关系模型

S\_T\_U201911658 共有三张表分别是 SC, Student, Course, ER 图如下图 5.1 所示

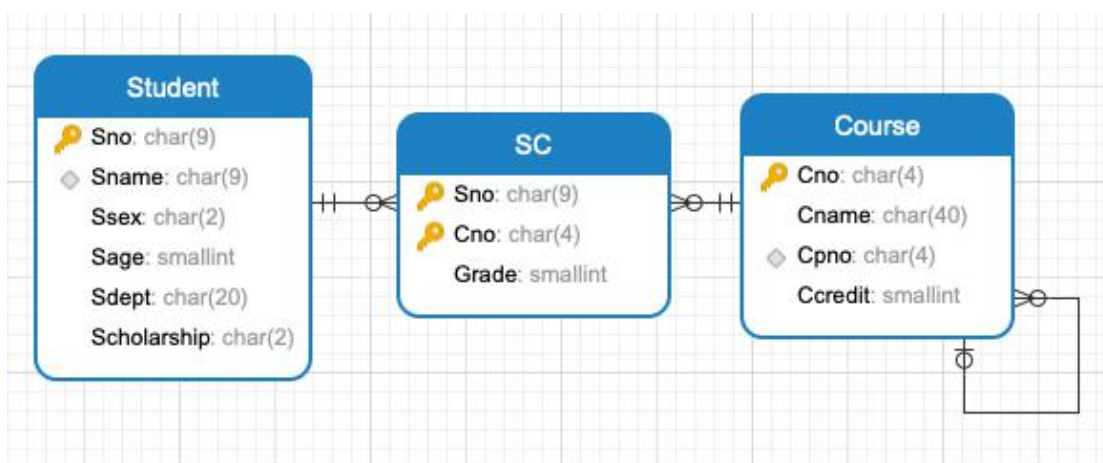


图 5.1 ER 图

关系模型如下：

Student(Sno,Sname,Ssex,Sage,Sdept,Scholarship),其中， Sno 为主码， Sname 取唯一值

SC(Sno,Cno,Grade)， 其中 (Sno， Cno) 为主码， Sno 和 Cno 均为外码

Course(Cno,Cname,Cpno,Ccredit)， 其中， Cno 为主码， Cpno 为外码， 被参照表为 Course， 被参照列是 Cno

### 5.2.4 思路概述

使用 tkinter 进行图形化界面设计，主界面为登录界面，登录成功进入主菜单界面，共分为 9 个分支选项，涵盖了实验指导书要求实现的全部功能，同时增加了删除学生成绩和查询全部课程信息，查询全部选课信息的功能。具体实现框架

如下图 5.2 所示

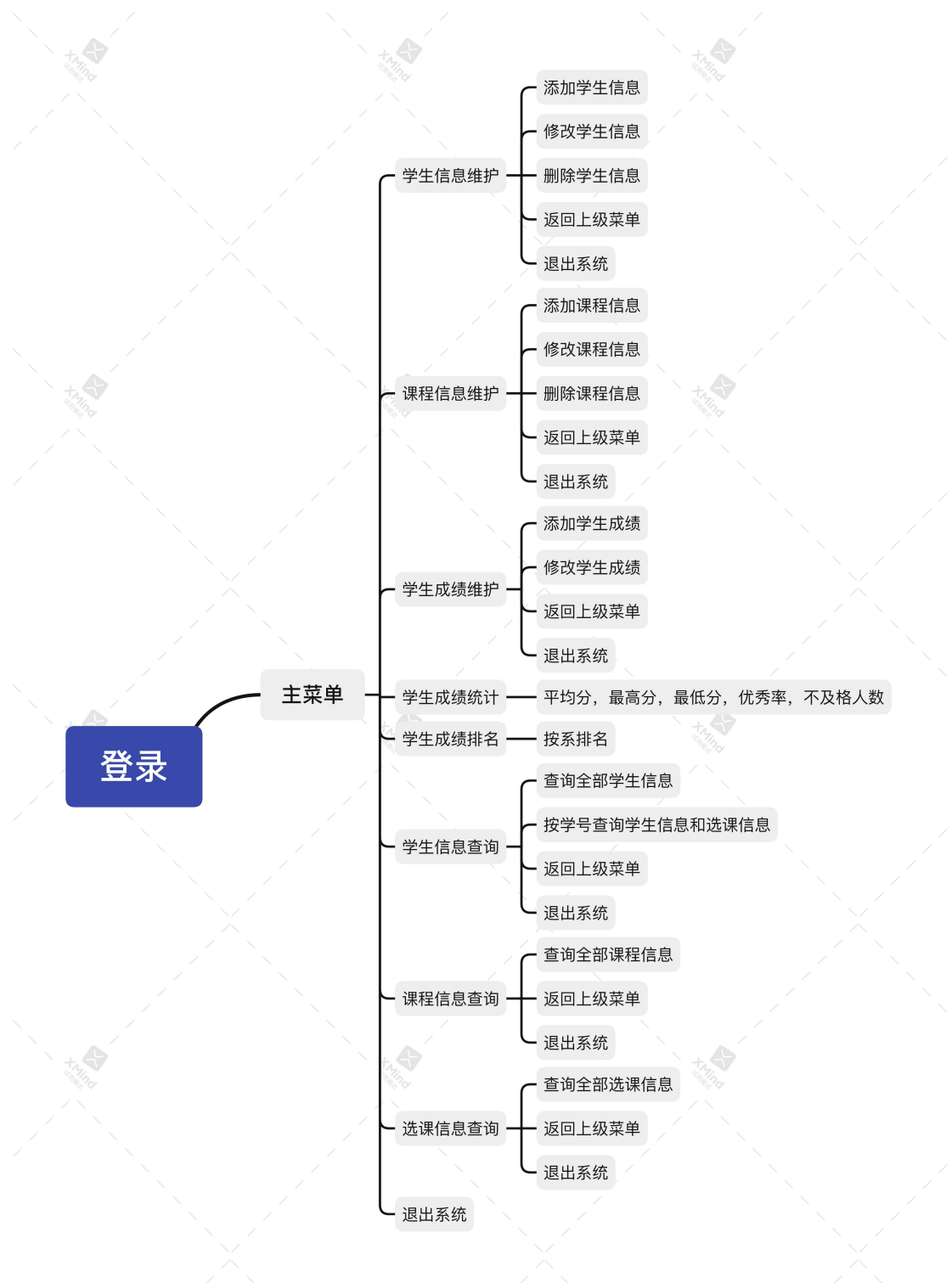


图 5.2 程序框架

对于插入类操作，如添加学生信息，添加课程信息，添加学生成绩等，根据用户输入的信息，检查是否符合条件（如成绩是否不合法，年龄是否不合法等）后进行 INSERT 操作，根据数据库返回信息，即是否操作成功，给用户相应提示信息。

对于修改类操作，如修改学生信息，修改课程信息，修改学生成绩等，根据

用户输入信息，同样检查是否符合条件后进行 UPDATE 操作，若执行成功返回修改后的相关信息给予用户反馈，如果修改失败给用户提示信息。

对于删除类操作，如删除未选课的课程，删除学生信息等，先进行查询，给予用户反馈信息，询问用户是否删除这些信息，若用户选‘是’则进行 DELETE 操作，选择‘否’则返回上级菜单。

对于查询类，如查询学生信息，学生选课等则提示用户输入索引项，进行 SELECT 操作，并给予反馈，若索引项不存在或查询失败则给予用户失败的反馈。对于无需用户输入的查询直接给予反馈信息。

### 5.2.5 功能实现效果

#### (1) 登录

##### 1.登录界面



图 5.3 登录界面

##### 2.登录成功提示



图 5.4 登录成功

##### 3.登录失败提示



图 5.5 登录失败

(2) 主菜单



图 5.6 主菜单

(3) 学生信息维护

菜单界面如下图 5.7

说明：该部分完成系统功能要求 “（1）： 新生入学信息增加，学生信息修改。”，并增加学生信息删除功能



图 5.7 学生信息维护菜单

1.添加学生信息



图 5.8 插入学生信息



| Objects Student@S_T_U2019116... |        |      |      |       |             |
|---------------------------------|--------|------|------|-------|-------------|
| Sno                             | Sname  | Ssex | Sage | Sdept | Scholarship |
| 20010507                        | Sylvia | 女    | 20   | CSE   | 是           |
| 200215121                       | 李勇     | 男    | 23   | CS    | 否           |
| 200215122                       | 刘晨     | 女    | 22   | CS    | 否           |
| 200215123                       | 王敏     | 女    | 19   | MA    | 否           |
| 200215125                       | 张立     | 男    | 20   | IS    | 否           |
| 200215126                       | 管宽     | 男    | 21   | IS    | 否           |
| 200215127                       | 秦冉     | 女    | 20   | MA    | 否           |
| 200215128                       | 李欣宇    | 女    | 21   | CSE   | 否           |

图 5.9 插入后 navicat 中 Student 表

## 2.修改学生信息

学生信息管理系统

学号: 20010507

姓名: sylviaLee

性别: ☐ 男 ☐ 女 ☒ 不修改

年龄:

系别:

奖学金: ☐ 是 ☐ 否 ☒ 不修改

修改成功

该学生信息修改后如下:  
20010507 sylviaLee 女 20 CSE 是

图 5.10 修改学生信息

| Sno       | Sname     | Ssex | Sage | Sdept | Scholarship |
|-----------|-----------|------|------|-------|-------------|
| 20010507  | sylviaLee | 女    | 20   | CSE   | 是           |
| 200215121 | 李勇        | 男    | 23   | CS    | 否           |
| 200215122 | 刘晨        | 女    | 22   | CS    | 否           |
| 200215123 | 王敏        | 女    | 19   | MA    | 否           |
| 200215125 | 张立        | 男    | 20   | IS    | 否           |
| 200215126 | 管宽        | 男    | 21   | IS    | 否           |
| 200215127 | 秦冉        | 女    | 20   | MA    | 否           |
| 200215128 | 李欣宇       | 女    | 21   | CSE   | 否           |

图 5.11 修改后 navicat 中 Student 表

## 3.删除学生信息

学生信息管理系统

学号: 20010507

提示

该学生信息如下:  
20010507 sylviaLee 女 20 CSE 是

您确定要删除吗

图 5.12 询问是否确认删除





图 5.13 成功删除提示

| Sno       | Sname | Ssex | Sage | Sdept | Scholarship |
|-----------|-------|------|------|-------|-------------|
| 200215121 | 李勇    | 男    | 23   | CS    | 否           |
| 200215122 | 刘晨    | 女    | 22   | CS    | 否           |
| 200215123 | 王敏    | 女    | 19   | MA    | 否           |
| 200215125 | 张立    | 男    | 20   | IS    | 否           |
| 200215126 | 管宽    | 男    | 21   | IS    | 否           |
| 200215127 | 秦冉    | 女    | 20   | MA    | 否           |
| 200215128 | 李欣宇   | 女    | 21   | CSE   | 否           |

图 5.14 删除后 navicat 中 Student 表

#### (4) 课程信息维护

菜单界面如下图 5.15

说明：该部分完成系统功能要求 “（2）：课程信息维护(增加新课程，修改课程信息，删除没有选课的课程信息)。”



图 5.15 课程信息维护菜单

#### 1.添加课程信息



图 5.16 添加课程信息

| Cno | Cname     | Cpno   | Ccredit |
|-----|-----------|--------|---------|
| 1   | 数据库概论     | 5      | 4       |
| 10  | 计算机网络     | (NULL) | 2       |
| 11  | 大学物理      | 2      | 4       |
| 2   | 高等数学      | (NULL) | 2       |
| 3   | 信息系统      | 5      | 4       |
| 4   | 操作系统      | 6      | 3       |
| 5   | 数据结构      | 7      | 4       |
| 6   | 数据处理      | (NULL) | 2       |
| 7   | pascal 语言 | 6      | 2       |
| 8   | C 语言设计    | (NULL) | 2       |
| 9   | 编译原理      | (NULL) | 3       |

图 5.17 添加课程信息后的 navicat 中 course 表

## 2.修改课程信息

学生信息管理系统

课程号: 11

课程名:

先修课程号:

学分: 3

确定

修改成功

该课程信息修改后如下:  
11 大学物理 2 3

OK

图 5.18 修改课程信息

| Cno | Cname | Cpno   | Ccredit |
|-----|-------|--------|---------|
| 1   | 数据库概论 | 5      | 4       |
| 10  | 计算机网络 | (NULL) | 2       |
| 11  | 大学物理  | 2      | 3       |

图 5.19 修改课程信息后的 navicat 中 course 表

## 3.删除未选课的课程信息

学生信息管理系统

下面是全部未被选课的课程信息

| Cno | Cname | Cpno | Ccredit |
|-----|-------|------|---------|
| 10  | 计算机网络 | None | 2       |
| 11  | 大学物理  | 2    | 3       |

删除

提示

您确定要删除吗

Cancel OK

图 5.20 删除未选课的课程信息



图 5.21 删除课程信息成功提示

|   | Cno | Cname     | Cpno   | Ccredit |
|---|-----|-----------|--------|---------|
| 1 |     | 数据库概论     | 5      | 4       |
| 2 |     | 高等数学      | (NULL) | 2       |
| 3 |     | 信息系统      | 5      | 4       |
| 4 |     | 操作系统      | 6      | 3       |
| 5 |     | 数据结构      | 7      | 4       |
| 6 |     | 数据处理      | (NULL) | 2       |
| 7 |     | pascal 语言 | 6      | 2       |
| 8 |     | C 语言设计    | (NULL) | 2       |
| 9 |     | 编译原理      | (NULL) | 3       |

图 5.22 删除课程信息后的 navicat 中 course 表

### (5) 学生成绩维护

菜单界面如下图 5.23

说明：该部分完成系统功能要求 “（3）录入学生成绩，修改学生成绩。”



图 5.23 学生成绩维护菜单

### 1. 录入学生成绩



图 5.24 录入学生成绩

|           |   |     |
|-----------|---|-----|
| 200215122 | 2 | 90  |
| 200215122 | 3 | 80  |
| 200215122 | 9 | 100 |
| 200215123 | 4 | 90  |
| 200215125 | 5 | 56  |

图 5.25 录入学生成绩后 navicat 中 SC 表

## 2.修改学生成绩

图 5.26 修改学生成绩

|           |   |    |
|-----------|---|----|
| 200215122 | 1 | 75 |
| 200215122 | 2 | 90 |
| 200215122 | 3 | 80 |
| 200215122 | 9 | 99 |
| 200215123 | 4 | 90 |
| 200215125 | 5 | 56 |
| 200215125 | 6 | 75 |

图 5.27 修改学生成绩后 navicat 中 SC 表

### (6) 学生成绩统计

说明：该部分完成系统功能要求“（4）按系统统计学生的平均成绩、最好成绩、最差成绩、优秀率、不及格人数。”

| Sdept | AvgGrade | MaxGrade | MinGrade |
|-------|----------|----------|----------|
| CS    | 86.5714  | 99       | 75       |
| MA    | 86.7500  | 90       | 78       |
| IS    | 79.7500  | 99       | 56       |
| CSE   | 84.0000  | 90       | 78       |

| Sdept | Rate  |
|-------|-------|
| CS    | 28.6% |
| MA    | 50.0% |
| IS    | 25.0% |
| CSE   | 50.0% |

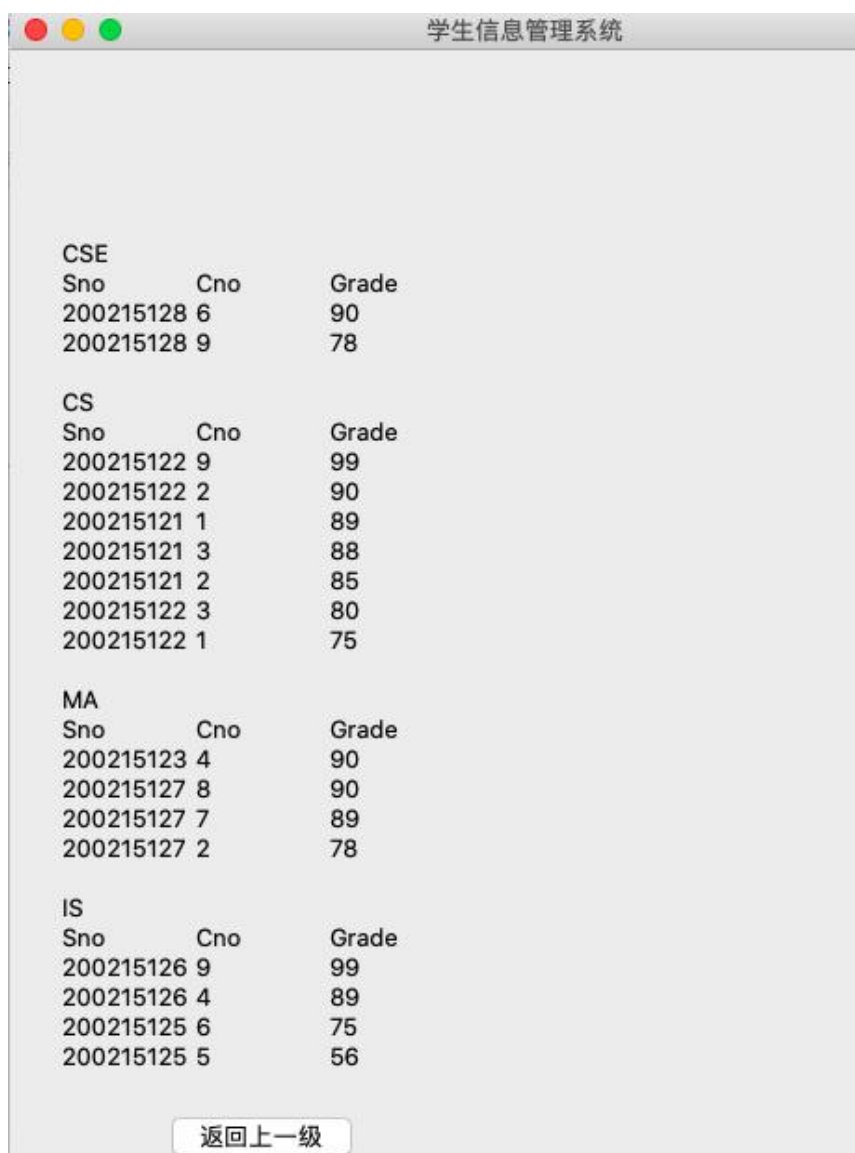
  

| Sdept | Number |
|-------|--------|
| IS    | 1      |

图 5.28 学生成绩统计

(7) 学生成绩排名

说明：该部分完成系统功能要求“（5）按系对学生成绩进行排名，同时显示出学生、课程和成绩信息。”



| Sno       | Cno | Grade |
|-----------|-----|-------|
| 200215128 | 6   | 90    |
| 200215128 | 9   | 78    |

| Sno       | Cno | Grade |
|-----------|-----|-------|
| 200215122 | 9   | 99    |
| 200215122 | 2   | 90    |
| 200215121 | 1   | 89    |
| 200215121 | 3   | 88    |
| 200215121 | 2   | 85    |
| 200215122 | 3   | 80    |
| 200215122 | 1   | 75    |

| Sno       | Cno | Grade |
|-----------|-----|-------|
| 200215123 | 4   | 90    |
| 200215127 | 8   | 90    |
| 200215127 | 7   | 89    |
| 200215127 | 2   | 78    |

| Sno       | Cno | Grade |
|-----------|-----|-------|
| 200215126 | 9   | 99    |
| 200215126 | 4   | 89    |
| 200215125 | 6   | 75    |
| 200215125 | 5   | 56    |

返回上一级

图 5.29 学生成绩排名

(8) 学生信息查询

菜单界面如下图 5.30 所示

说明：该部分完成系统功能要求“（6）输入学号，显示该学生的基本信息和选课信息。”，同时增加了查询全部学生信息的功能



图 5.30 学生信息查询菜单

### 1. 查询所有学生信息



图 5.31 查询所有学生信息

### 2. 按学号查询学生信息和选课信息



图 5.32 输入学号查询学生信息



图 5.33 查询学生信息界面

### (9) 课程信息查询

菜单界面如下图 5.34 所示

说明: 该部分为指导书要求功能之外的设计



图 5.34 查询课程信息菜单

### 1. 查询所有课程信息

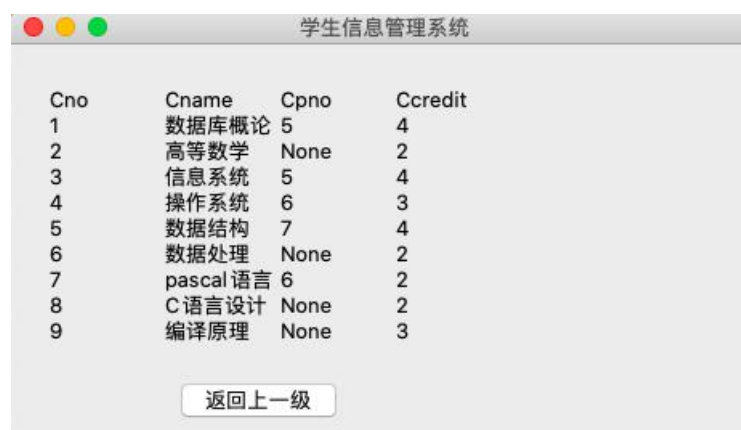


图 5.35 查询所有课程信息显示

### (10) 选课信息查询

菜单界面如下图 5.36 所示

说明：该部分为指导书要求功能之外的设计



图 5.36 查询选课信息菜单

### 1. 查询所有选课信息



图 5.37 所有选课信息显示

#### (11) 退出系统



图 5.38 退出系统

## 5.3 任务总结

该部分实验主要是学习如何设计编程应用程序来控制数据库，前期学会使用 `mysql.connector` 的几个接口函数，并对要实现的功能进行架构，尤其是在使用框架图画出整个应用程序的结构后，整个程序会好写很多，在使用 `python` 写完控制台程序后因为学有余力又想尝试写 GUI，这也是我第一次使用 `tkinter` 写 GUI，过程中遇到不少困难，但是都通过查阅资料基本解决，虽然写出的 UI 略显简陋，但是成就感满满。

这个程序实际上还有一些 BUG，未来得及发现并修复，以待后期进行修改。



## 6 课程总结

1.学习掌握 Mysql 数据库和可视化客户端管理工具 navicat 的使用。

2.学习和巩固使用 sql 语言对数据库的基本操作，包括创建数据库，创建表，增删改查数据等。

3.进一步了解了 sql 对数据库查询中的复杂语句，包括等值连接、自身连接、签到查询、子查询、谓词查询等，带有 all/any 等谓词的子查询等，感受到 sql 语句的查询具有技巧性，尤其是关于 exist 的使用，虽然在实验中有一定了解，但是在实际使用时并不算熟练，还需要进一步巩固加强学习。

4.学习到了数据库中 sql 的高级操作，包括创建视图，创建触发器，存储过程和函数等的使用，对用户权限的管理和用户完整性约束等，对熟练和巧妙使用 mysql 打下基础。

5.学习了数据库的设计和开发技巧，使用 python 写出了一个相对完整带有 UI 的能够对接 mysql 数据库的学生信息管理系统，但是由于时间有限在设计上还有一些不足，虽已经带有了一定的输入合法性判断，但是判断规则还不够强，错误处理方面也有瑕疵，错误类型还有待进一步细分，以待后期进行完善。

## 参考文献

- 1.sql 使用小技巧之约束 (Constraints、NOT NULL、UNIQUE、PRIMARY KEY);  
<https://blog.csdn.net/luyaran/article/details/82460172>
- 2.sql 的四种连接——左外连接、右外连接、内连接、全连接;  
<https://www.cnblogs.com/zccfrancis/p/14669178.html>
- 3.Python 图形化界面设计; <https://www.jianshu.com/p/91844c5bca78>
- 4.python tkinter 实现界面切换的示例代码; <https://www.jb51.net/article/163046.htm>