



华中科技大学

数据库系统概论实验报告

姓 名： 罗南清
学 院： 网络空间安全学院
专 业： 信息安全专业
班 级： 1701
学 号： U201714868
指导教师： 路松峰

| | |
|------|--|
| 分数 | |
| 教师签名 | |

2019 年 12 月 30 日

目 录

| | |
|---------------------------------|-----------|
| 1 课程任务概述 | 1 |
| 2 实验任务一 数据库定义与基本操作 | 2 |
| 2.1 任务要求..... | 2 |
| 2.2 完成过程..... | 2 |
| 2.3 任务总结..... | 9 |
| 3 实验任务二 SQL 的复杂操作 | 10 |
| 3.1 任务要求..... | 10 |
| 3.2 完成过程..... | 10 |
| 3.3 任务总结..... | 17 |
| 4 实验任务三 SQL 的高级实验 | 18 |
| 4.1 任务要求..... | 18 |
| 4.2 完成过程..... | 18 |
| 4.3 任务总结..... | 25 |
| 5 实验任务四 数据库设计 | 26 |
| 5.1 任务要求..... | 26 |
| 5.2 完成过程..... | 26 |
| 5.3 任务总结..... | 32 |
| 6 课程总结..... | 33 |

1 课程任务概述

实验一要求熟练掌握 SQL 的数据定义语句 CREATE、ALTER、DROP、Select 等。

实验二要求熟练掌握 SQL 的连接查询语句；SQL 的嵌套查询语句；表名前缀、别名前缀的用法；不相关子查询和相关子查询的区别和用法；不同查询之间的等价替换方法（一题多解）及限制；掌握 SQL 的数据更新语句 INSERT、UPDATE、DELETE 等。

实验三要求掌握视图的定义与操作；对触发器的定义；对存储过程的定义；如何对用户进行授权和收回权限；用户定义完整性的方法等。

实验四要求熟练掌握使用 SQL 语句设计数据库的方法，实现前述实验的学生管理系统。

2 实验任务一 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 实验要求

- (1) 熟练掌握 SQL 的数据定义语句 CREATE、ALTER、DROP、Select
- (2) 写出实验报告

2.1.3 实验内容

- (1) 查询全体学生的学号、姓名和年龄；
 - (2) 查询所有计算机系学生的详细记录；
 - (3) 找出考试成绩为优秀（90 分及以上）或不及格的学生的学号、课程号及成绩；
 - (4) 查询年龄不在 19~20 岁之间的学生姓名、性别和年龄；
 - (5) 查询数学系（MA）、信息系（IS）的学生的姓名和所在系；
 - (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；
 - (7) 找出所有没有选修课成绩的学生学号和课程号；
 - (8) 查询学生 200215121 选修课的最高分、最低分以及平均成绩；
 - (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；
 - (10) 查询每个系名及其学生的平均年龄。
- （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？）

2.2 完成过程

（包括主要操作步骤描述及其执行效果，或者所用的 SQL 语句及语句执行、调试的主要过程、效果。）

2.2.1 创建表

首先建立一个数据库，建立的数据库如图 2.1 所示：

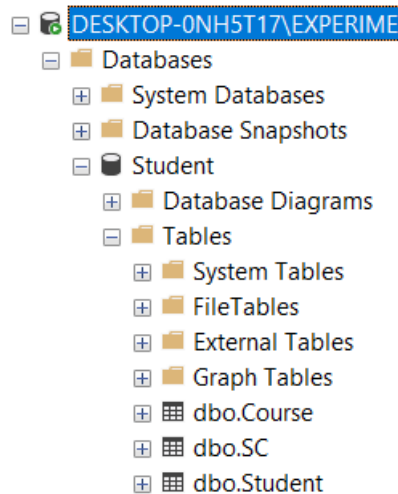


图 2.1 建立的测试数据库

在数据库 S_T_学号中创建学生表 Student、课程表 Course 和选修表 SC。

```
create table Student
```

```
(Sno CHAR(9) PRIMARY KEY,
```

```
Sname CHAR(20) UNIQUE,
```

```
Ssex CHAR(2),
```

```
Sage SMALLINT,
```

```
Sdept CHAR(20),
```

```
Scholarship char(2));
```

```
create table Course
```

```
(Cno CHAR(4) PRIMARY KEY,
```

```
Cname CHAR(40),
```

```
Cpno CHAR(4),
```

```
Ccredit SMALLINT,
```

```
FOREIGN KEY (Cpno) REFERENCES Course(Cno));
```

```
create table SC
```

```
(Sno CHAR(9),
```

```
Cno CHAR(4),
```

```
Grade SMALLINT,
```

```
primary key (Sno, Cno),
```

```
FOREIGN KEY (Sno) REFERENCES Student(Sno),
```

```
FOREIGN KEY (Cno) REFERENCES Course(Cno));
```

2.2.2 插入数据

(1) 为表 Student 插入如图 2.2 的数据

```
insert into student values('200215121','李勇','男',20,'CS', '否');
```

```
insert into student values('200215122','刘晨','女',19,'CS', '否');
```

insert into student values('200215123','王敏','女',18,'MA', '否');

insert into student values('200215125','张立','男',19,'IS', '否');

表 Student

| 学号 Sno | 姓名 Sname | 性别 Ssex | 年龄 Sage | 所在系 Sdept | 奖学金 Scholarship |
|-----------|-------------|------------|------------|--------------|--------------------|
| 200215121 | 李勇 | 男 | 20 | CS | 否 |
| 200215122 | 刘晨 | 女 | 19 | CS | 否 |
| 200215123 | 王敏 | 女 | 18 | MA | 否 |
| 200215125 | 张立 | 男 | 19 | IS | 否 |

图 2.2 Student 表

(2) 为表 Course 插入如图 2.3 的数据

insert into course values('1', '数据库', NULL,4);

insert into course values('2', '数学', NULL,2);

insert into course values('3', '信息系统', NULL,4);

insert into course values('4', '操作系统', NULL,3);

insert into course values('5', '数据结构', NULL,4);

insert into course values('6', '数据处理', NULL, 2);

insert into course values('7', 'java', NULL,4);

通过 update 进行修改数据内容

update Course set Cpno = '5' where Cno = '1';

update Course set Cpno = '1' where Cno = '3';

update Course set Cpno = '6' where Cno = '4';

update Course set Cpno = '7' where Cno = '5';

update Course set Cpno = '6' where Cno = '7';

表 Course

| 课程号 Cno | 课程名 Cname | 现行课 Cpno | 学分 Ccredit |
|------------|--------------|-------------|---------------|
| 1 | 数据库 | 5 | 4 |
| 2 | 数学 | | 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 | | 2 |
| 7 | PASCAL 语言 | 6 | 4 |

图 2.3 Course 表

(3) 为表 SC 插入如图 2.4 的数据

insert into SC values('200215121', '1',92);

insert into SC values('200215121', '2',85);

insert into SC values('200215121', '3',88);

insert into SC values('200215122', '2',90);

insert into SC values('200215122', '3',80);

表 SC

| 学号 Sno | 课程号 Cno | 成绩 Grade |
|-----------|------------|-------------|
| 200215121 | 1 | 92 |
| 200215121 | 2 | 85 |
| 200215121 | 3 | 88 |
| 200215122 | 2 | 90 |
| 200215122 | 3 | 80 |

图 2.4 SC 表

2.2.3 基本练习

(1) SELECT 语句的基本用法: 查询全体学生的详细记录。

```
SELECT Sno,Sname,Ssex,Sage,Sdept
FROM Student
```

查询结果如图 2.5 所示:

| | Sno | Sname | Ssex | Sage | Sdept |
|---|-----------|-------|------|------|-------|
| 1 | 200215121 | 李勇 | 男 | 20 | CS |
| 2 | 200215122 | 刘晨 | 女 | 19 | CS |
| 3 | 200215123 | 王敏 | 女 | 18 | MA |
| 4 | 200215125 | 张立 | 男 | 19 | IS |

图 2.5 基本练习 1

(2)使用 WHERE 子句进行有条件的查询: 查询选修 2 号课程且成绩在 90 分及以上的所有学生的学号、姓名

```
SELECT Student.Sno, student.Sname
FROM Student, SC
WHERE Student.Sno = SC.Sno AND SC.Cno= ' 2 ' AND SC.Grade > =90
```

查询结果如图 2.6 所示:

| | Sno | Sname |
|---|-----------|-------|
| 1 | 200215122 | 刘晨 |

图 2.6 基本练习 2

(3) 使用 IN, NOT IN, BETWEEN 等谓词查询: 查询信息系 (IS)、数学系 (MA) 和计算机科学系 (CS) 学生的姓名和性别。

```
SELECT Sname,Ssex
FROM Student
WHERE Sdept IN ( 'IS','MA','CS' )
```

查询结果如图 2.7 所示:

| | Sname | Ssex |
|---|-------|------|
| 1 | 李勇 | 男 |
| 2 | 刘晨 | 女 |
| 3 | 王敏 | 女 |
| 4 | 张立 | 男 |

图 2.7 基本练习 3

(4) 利用 **LIKE** 子句实现模糊查询：查询所有姓刘学生的姓名、学号和性别。

```
SELECT Sname,Sno,Ssex
FROM Student
WHERE Sname LIKE '刘%'
```

查询结果如图 2.8 所示：

| | Sname | Sno | Ssex |
|---|-------|-----------|------|
| 1 | 刘晨 | 200215122 | 女 |

图 2.8 基本练习 4

(5) 利用 **ORDER** 子句为结果排序：查询选修了 3 号课程的学生的学号及其成绩，查询结果按分数降序排列。

```
SELECT Sno,Grade
FROM SC
WHERE Cno= '3'
ORDER BY Grade DESC
```

查询结果如图 2.9 所示：

| | Sno | Grade |
|---|-----------|-------|
| 1 | 200215121 | 88 |
| 2 | 200215122 | 80 |

图 2.9 基本练习 5

(6) 用 **SQL Server** 的统计函数进行统计计算：计算 1 号课程的学生平均成绩。

```
SELECT AVG(Grade)
FROM SC
WHERE Cno= '1'
```

查询结果如图 2.10 所示：

| | (无列名) |
|---|-------|
| 1 | 92 |

图 2.10 基本练习 6

(7) 用 GROUP BY 子句实现分组查询的方法：查询选修了 3 门以上课程的学生学号。

```
SELECT Sno
FROM SC
GROUP BY Sno
HAVING COUNT(*) >3
```

查询结果为空。

2.2.4 扩展练习

(1) 查询全体学生的学号、姓名和年龄；

```
SELECT Sno,Sname,Ssex,Sage,Sdept
FROM Student
```

实验结果如图 2.11 所示：

| | Sno | Sname | Sage |
|---|-----------|-------|------|
| 1 | 200215121 | 李勇 | 26 |
| 2 | 200215122 | 刘晨 | 25 |
| 3 | 200215123 | 王敏 | 20 |

图 2.11 拓展训练 1

(2) 查询所有计算机系学生的详细记录

```
SELECT Sno,Sname,Ssex,Sage,Sdept
FROM Student
WHERE Sdept='CS'
```

实验结果如图 2.12 所示：

| | Sno | Sname | Ssex | Sage | Sdept |
|---|-----------|-------|------|------|-------|
| 1 | 200215121 | 李勇 | 男 | 26 | CS |
| 2 | 200215122 | 刘晨 | 女 | 25 | CS |

图 2.12 拓展训练 2

(3) 找出考试成绩为优秀（90 分及以上）或不及格的学生的学号、课程号及成绩；

```
SELECT Student.Sno, SC.Cno,SC.Grade
FROM Student, SC
WHERE Student.Sno = SC.Sno AND (SC.Grade > 90 OR SC.Grade < 60)
```

实验结果如图 2.13 所示：

| | Sno | Cno | Grade |
|---|-----------|-----|-------|
| 1 | 200215121 | 1 | 92 |
| 2 | 200215122 | 1 | 95 |

图 2.13 拓展训练 3

(4) 查询年龄不在 19~20 岁之间的学生姓名、性别和年龄;

```
SELECT Sname,Sdept,Sage
FROM Student
WHERE Sage NOT BETWEEN 19 AND 20
```

实验结果如图 2.14 所示:

| | Sname | Sdept | Sage |
|---|-------|-------|------|
| 1 | 李勇 | CS | 26 |
| 2 | 刘晨 | CS | 25 |

图 2.14 拓展训练 4

(5) 查询数学系 (MA)、信息系 (IS) 的学生的姓名和所在系;

```
SELECT Sname, Sdept
FROM Student
WHERE Sdept='MA' OR Sdept='IS'
```

实验结果如图 2.15 所示:

| | Sname | Sdept |
|---|-------|-------|
| 1 | 王敏 | MA |

图 2.15 拓展训练 5

(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;

```
SELECT Cno,Cname,Ccredit
FROM Course
WHERE Cname LIKE '数据%'
```

实验结果如图 2.16 所示:

| | Cno | Cname | Ccredit |
|---|-----|-------|---------|
| 1 | 1 | 数据库 | 4 |
| 2 | 5 | 数据结构 | 4 |
| 3 | 6 | 数据处理 | 2 |

图 2.16 拓展训练 6

(7) 找出所有没有选修课成绩的学生学号和课程号;

```
SELECT Student.Sno,Course.Cno
FROM Student,Course,SC
WHERE Student.Sno=SC.Sno AND SC.Cno=Course.Cno AND SC.Grade is
NULL
```

由于表中所有的学生都已经选选修课了, 所以查询的结果为空。

(8) 查询学生 200215121 选修课的最高分、最低分以及平均成绩;

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

实验结果如图 2.17 所示，

| | | | |
|---|----|----|----|
| 1 | 92 | 78 | 85 |
|---|----|----|----|

图 2.17 拓展训练 8

(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；

```
SELECT Sno,Grade
FROM SC
WHERE Cno='2' ORDER BY Grade ASC
```

实验结果如图 2.18 所示，

| | Sno | Grade |
|---|-----------|-------|
| 1 | 200215121 | 85 |
| 2 | 200215122 | 90 |

图 2.18 拓展训练 9

(10) 查询每个系名及其学生的平均年龄；

```
SELECT Sdept,AVG(Sage)
FROM Student
GROUP BY Sdept
```

实验结果如图 2.19 所示，

| | Sdept | (No column name) |
|---|-------|------------------|
| 1 | CS | 25 |
| 2 | MA | 20 |

图 2.19 拓展训练 10

2.3 任务总结

第一次做实验的时候，配置环境时也花了不少时间。因为不知道选择哪款软件，发现有微软的 SQL Server 和 MySQL，不知道哪款和我们的最接近。但是从周边同学反应来看，才知道我们学的数据库语言仅仅是一个样例，其实对应不同的数据库软件会有一些不同的语法规则，但是大体思想是和书本内容差不多。

其次就是微软的 SQL Server 除了官方文档就不要去寻找其它东西读了，网上有很多老旧版本的写法，新版本软件中已经不支持了。

总之，第一次实验体验到了操作数据库的快乐。但同时加深了我对 SQL 语句的了解，比如创建表 Create，插入信息 Insert，各种各样的查询语句等。同时，通过这次实验，我第一次接触 Microsoft SQL Server Management Studio 这个软件，通过自己的不断摸索，我感受到使用这个软件来学习数据库和做数据库实验非常的方便。

3 实验任务二 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

- (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

3.2 完成过程

3.2.1 基本练习

（1）等值连接查询与自然连接查询：查询每个学生及其选修课的情况。

```
SELECT Student.*, SC.*  
FROM Student, SC  
WHERE Student.Sno = SC.Sno;
```

查询结果如图 3.1 所示：

| | Sno | Sname | Ssex | Sage | Sdept | Scholarship | Sno | Cno | Grade |
|---|-----------|-------|------|------|-------|-------------|-----------|-----|-------|
| 1 | 200215121 | 李勇 | 男 | 20 | CS | 否 | 200215121 | 1 | 92 |
| 2 | 200215121 | 李勇 | 男 | 20 | CS | 否 | 200215121 | 2 | 85 |
| 3 | 200215121 | 李勇 | 男 | 20 | CS | 否 | 200215121 | 3 | 88 |
| 4 | 200215122 | 刘晨 | 女 | 19 | CS | 否 | 200215122 | 2 | 90 |
| 5 | 200215122 | 刘晨 | 女 | 19 | CS | 否 | 200215122 | 3 | 80 |

图 3.1 基本练习（1）

（2）自身连接查询：查询每一门课的间接先修课。

```
SELECT FIRST.Cno, SECOND.Cpno
FROM Course FIRST, Course SECOND
WHERE FIRST.Cpno = SECOND.Cno;
```

查询结果如图 3.2 所示：

| | Cno | Cpno |
|---|-----|------|
| 1 | 1 | 7 |
| 2 | 3 | 5 |
| 3 | 4 | NULL |
| 4 | 5 | 6 |
| 5 | 7 | NULL |

图 3.2 基本练习（2）

（3）外连接查询：查询每个学生及其选修课的情况（要求输出所有学生--含未选修课程的学生的情况）

```
SELECT Student.Sno, Sname, Ssex, Sage, Sdept, Cno, Grade
FROM Student
LEFT OUTER JOIN SC ON(Student.Sno = SC.Sno);
```

查询结果如图 3.3 所示：

| | Sno | Sname | Ssex | Sage | Sdept | Cno | Grade |
|---|-----------|-------|------|------|-------|------|-------|
| 1 | 200215121 | 李勇 | 男 | 20 | CS | 1 | 92 |
| 2 | 200215121 | 李勇 | 男 | 20 | CS | 2 | 85 |
| 3 | 200215121 | 李勇 | 男 | 20 | CS | 3 | 88 |
| 4 | 200215122 | 刘晨 | 女 | 19 | CS | 2 | 90 |
| 5 | 200215122 | 刘晨 | 女 | 19 | CS | 3 | 80 |
| 6 | 200215123 | 王敏 | 女 | 18 | MA | NULL | NULL |
| 7 | 200215125 | 张立 | 男 | 19 | IS | NULL | NULL |

图 3.3 基本练习（3）

（4）复合条件连接查询：查询选修了 2 号课程而且成绩在 90 以上的所有学生的学号和姓名。

```
SELECT Student.Sno, Sname
FROM Student, SC
WHERE Student.Sno = SC.Sno AND SC.Cno = '2' AND SC.Grade >= 90;
```

查询结果如图 3.4 所示：

| | Sno | Sname |
|---|-----------|-------|
| 1 | 200215122 | 刘晨 |

图 3.4 基本练习（4）

（5）嵌套查询（带有 IN 谓词的子查询）：查询与“刘晨”在同一个系学习的学生的学号、姓名和所在系。

```
SELECT Sno, Sname, Sdept
FROM Student
WHERE Sdept
IN (SELECT Sdept FROM Student WHERE Sname = '刘晨');
```

查询结果如图 3.5 所示：

| | Sno | Sname | Sdept |
|---|-----------|-------|-------|
| 1 | 200215121 | 李勇 | CS |
| 2 | 200215122 | 刘晨 | CS |

图 3.5 基本练习（5）

（6）嵌套查询（带有比较运算符的子查询）：找出每个学生超过他所选修课程平均成绩的课程号。

```
SELECT Sno, Cno
FROM SC x
WHERE Grade >= ( SELECT AVG(Grade) FROM SC y WHERE y.Sno =
x.Sno);
```

查询结果如图 3.6 所示

| | Sno | Cno |
|---|-----------|-----|
| 1 | 200215121 | 1 |
| 2 | 200215121 | 3 |
| 3 | 200215122 | 2 |

图 3.6 基本练习（6）

（7）嵌套查询（带有 ANY 或 ALL 谓词的子查询） 例如：查询其他系中比计算机系某个学生年龄小的学生的姓名和年龄。

```
SELECT Sname, Sage
FROM Student
WHERE Sage < (SELECT MAX(Sage) FROM Student WHERE Sdept =
'CS')AND Sdept <> 'CS';
```

查询结果如图 3.7 所示：

| | Sname | Sage |
|---|-------|------|
| 1 | 王敏 | 18 |
| 2 | 张立 | 19 |

图 3.7 基本练习（7）

（8）嵌套查询（带有 EXISTS 谓词的子查询）：查询所有选修了 1 号课程的学生姓名。

```
SELECT Sname
FROM Student
WHERE EXISTS
(SELECT * FROM SC WHERE Sno=Student.Sno AND Cno='1');
```

查询结果如图 3.8 所示

| | Sname |
|---|-------|
| 1 | 李勇 |

图 3.8 基本练习（8）

（9）集合查询：查询计算机系的学生以及年龄不大于 19 岁的学生。

```
SELECT *
FROM Student
WHERE Sdept='CS' UNION
SELECT * FROM Student WHERE Sage<=19;
```

查询结果如图 3.9 所示：

| | Sno | Sname | Ssex | Sage | Sdept | Scholarship |
|---|-----------|-------|------|------|-------|-------------|
| 1 | 200215121 | 李勇 | 男 | 20 | CS | 否 |
| 2 | 200215122 | 刘晨 | 女 | 19 | CS | 否 |
| 3 | 200215123 | 王敏 | 女 | 18 | MA | 否 |
| 4 | 200215125 | 张立 | 男 | 19 | IS | 否 |

图 3.9 基本练习（9）

(10) update 语句用于对表进行更新：将信息系所有学生的年龄增加 1 岁。

```
UPDATE Student
SET Sage= Sage+1
WHERE Sdept='IS'
```

(11) delete 语句用于对表进行删除：删除学号为 95019 的学生记录。

```
DELETE
FROM Student
WHERE Sno='95019'
```

(12) insert 语句用于对表进行插入：插入一条选课记录('95020', '1')。

```
INSERT INTO SC(Sno, Cno)
```

VALUES ('95020', '1')

3.2.2 扩展练习

(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）。

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

查询结果如图 3.10 所示：

| | Cname | Cno | Sno | Grade |
|----|-------|------|-----------|-------|
| 1 | 数据库 | 1 | 200215121 | 92 |
| 2 | 数据库 | 1 | 200215122 | 95 |
| 3 | 数学 | 2 | 200215121 | 85 |
| 4 | 数学 | 2 | 200215122 | 90 |
| 5 | 信息系统 | 3 | 200215121 | 88 |
| 6 | 信息系统 | 3 | 200215122 | 80 |
| 7 | 操作系统 | NULL | NULL | NULL |
| 8 | 数据结构 | NULL | NULL | NULL |
| 9 | 数据处理 | NULL | NULL | NULL |
| 10 | java | 7 | 200215121 | 78 |
| 11 | C 语言 | NULL | NULL | NULL |

图 3.10 拓展训练 1

(2) 查询与“张立”同岁的学生的学号、姓名和年龄。（要求使用至少 3 种方法求解）

方法一：

```
SELECT Sno,Sname,Sage
FROM Student
WHERE Sage=
      (SELECT Sage
       FROM Student
       WHERE Sname = '张立');
```

方法二：

```
SELECT Sno,Sname,Sage
FROM Student
WHERE Sage IN
      (SELECT Sage
       FROM Student
       WHERE Sname = '张立');
```

方法三：

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


查询结果为空。

(3) 查询选修了 3 号课程而且成绩为良好 (80~89 分) 的所有学生的学号和姓名。

```
SELECT Student.Sno,Sname
FROM SC,Student
WHERE Cno='3' AND Grade BETWEEN 80 AND 90 AND
Sc.Sno=Student.Sno;
```

输出结果如图 3.11 所示,

| | Sno | Sname |
|---|-----------|-------|
| 1 | 200215121 | 李勇 |
| 2 | 200215122 | 刘晨 |

图 3.11 拓展训练 3

(4) 查询学生 200215122 选修的课程号、课程名

```
SELECT Course.Cno,Course.Cname
FROM Course,SC
WHERE Course.Cno = SC.Cno AND SC.Sno='200215122';
```

查询结果如图 3.12 所示,

| | Cno | Cname |
|---|-----|-------|
| 1 | 1 | 数据库 |
| 2 | 2 | 数学 |
| 3 | 3 | 信息系统 |

图 3.12 拓展训练 4

(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)

```
SELECT Sno, Cno
FROM SC x
WHERE Grade+5 < (SELECT AVG(Grade)
                  FROM SC y
                  WHERE x.Sno=y.Sno);
```

查询结果如图 3.13 所示,

| | Sno | Cno |
|---|-----------|-----|
| 1 | 200215121 | 7 |
| 2 | 200215122 | 3 |

图 3.13 拓展训练 5

(6) 查询比所有男生年龄都小的女生的学号、姓名和年龄。

```
SELECT Sno,Sname,Sage
FROM Student
```

```
WHERE Ssex='女' AND Sage < ALL(SELECT Sage
                                FROM Student
                                WHERE Ssex='男');
```

实验结果如图 3.14 所示，

| | Sno | Sname | Sage |
|---|-----------|-------|------|
| 1 | 200215122 | 刘晨 | 25 |
| 2 | 200215123 | 王敏 | 20 |

图 3.14 拓展训练 6

(7) 查询所有选修了 2 号课程的学生姓名及所在系

```
SELECT Sname, Sdept
FROM Student
WHERE EXISTS
    (SELECT *
     FROM SC
     WHERE Sno=Student.Sno AND Cno='2');
```

实验结果如图 3.15 所示，

| | Sname | Sdept |
|---|-------|-------|
| 1 | 李勇 | CS |
| 2 | 刘晨 | CS |

图 3.15 拓展训练 7

(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。

```
UPDATE Student
SET Sage=Sage+2
WHERE Sno IN (
    SELECT Sno
    FROM SC
    WHERE Grade BETWEEN 80 AND 89);
```

```
SELECT Sno,Sage
FROM Student
```

查询结果如图 3.16 所示，

| | Sno | Sage |
|---|-----------|------|
| 1 | 200215121 | 28 |
| 2 | 200215122 | 27 |
| 3 | 200215123 | 20 |

图 3.16 拓展训练 8

(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

```
INSERT
INTO Course(Cno,Cname)
```

```
VALUES('8','C 语言'),('9','人工智能');
```

```
SELECT *
```

```
FROM Course
```

查询结果如图 3.17 所示，

| | Cno | Cname | Cpno | Ccredit |
|---|-----|-------|------|---------|
| 3 | 3 | 信息系统 | 1 | 4 |
| 4 | 4 | 操作系统 | 6 | 3 |
| 5 | 5 | 数据结构 | 7 | 4 |
| 6 | 6 | 数据处理 | NULL | 2 |
| 7 | 7 | java | 6 | 4 |
| 8 | 8 | C 语言 | NULL | NULL |
| 9 | 9 | 人工智能 | NULL | NULL |

图 3.17 拓展训练 9

(10) 使用 delete 语句把人工智能课程删除，并查询出来

```
DELETE
```

```
FROM Course
```

```
WHERE Cname='人工智能';
```

```
SELECT *
```

```
FROM Course
```

查询结果如图 3.18 所示，

| | Cno | Cname | Cpno | Ccredit |
|---|-----|-------|------|---------|
| 2 | 2 | 数学 | NULL | 2 |
| 3 | 3 | 信息系统 | 1 | 4 |
| 4 | 4 | 操作系统 | 6 | 3 |
| 5 | 5 | 数据结构 | 7 | 4 |
| 6 | 6 | 数据处理 | NULL | 2 |
| 7 | 7 | java | 6 | 4 |
| 8 | 8 | C 语言 | NULL | NULL |

图 3.18 拓展训练 10

3.3 任务总结

通过这次实验，加深了我对等值连接、自身连接和外连接的理解，通过对结果的观察知道了他们的区别。学会了使用复杂语句来嵌套查询，比如 IN 子句、EXISTS 子句。实现了对 update、delete、insert 语句的使用。

其次，在这次实验的过程中，开始的时候我对等值连接、自身连接和外连接的定义不是很理解，后来通过查阅资料和分析结果，逐渐掌握了他们的区别和各自的使用方法。

4 实验任务三 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

4.2 完成过程

(1) 创建 CS 系的视图 CS_View

```
CREATE VIEW CS_VIEW
```

```
AS
```

```
SELECT *
```

```
FROM Student
```

```
WHERE Sdept='CS';
```

(2) 在视图 CS_View 上查询 CS 系选修了 1 号课程的学生

```
SELECT CS_VIEW.*
```

```
FROM CS_VIEW,SC
```

```
WHERE CS_VIEW.Sno=SC.Sno AND Cno='1';
```

查询结果如图 4.1 所示，

| | Sno | Sname | Ssex | Sage | Sdept | Scholarship |
|---|-----------|-------|------|------|-------|-------------|
| 1 | 200215121 | 李勇 | 男 | 28 | CS | 否 |
| 2 | 200215122 | 刘晨 | 女 | 27 | CS | 是 |

图 4.1 实验步骤 (2)

(3) 创建 IS 系成绩大于 80 的学生的视图 IS_View

```
CREATE VIEW IS_VIEW  
AS  
SELECT Student.Sno,Sname,Grade  
FROM Student, SC  
WHERE SC.Grade > 80 AND Student.Sno=SC.Cno AND Sdept='IS';
```

(4) 在视图 IS_View 查询 IS 系成绩大于 80 的学生

```
SELECT IS_VIEW.*  
FROM IS_VIEW  
WHERE IS_VIEW.Grade > 80;
```

查询结果如图 4.2 所示，

| | Sno |
|---|-----------|
| 1 | 200215126 |

图 4.2 实验步骤 (4)

(5) 删除视图 IS_View

```
DROP VIEW IS_VIEW;
```

(6) 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。

然后用 U1 登录，分别

- 1) 查询学生表 的信息；
- 2) 把所有学生的年龄增加 1 岁，然后查询；
- 3) 删除 IS 系的学生；
- 4) 查询 CS 系的选课信息。

用 U2 登录，分别

- 1) 在 SC 表中插入 1 条记录 ('200215122' , '1' , 75) ；
- 2) 查询 SC 表的信息，
- 3) 查询视图 CS_View 的信息

```
GRANT SELECT,UPDATE  
ON Student  
TO U1;
```

```
GRANT INSERT  
ON SC  
TO U2;
```

```
SELECT *  
FROM Student;
```

```
UPDATE Student
SET Sage=Sage+1;
```

```
SELECT *
FROM Student;
```

```
DELETE
FROM Student
WHERE Sdept='IS';
```

```
INSERT
INTO SC
VALUES('200215122','1',75);
```

```
SELECT *
FROM SC
```

```
SELECT *
FROM CS_VIEW;
```

查询结果如图 4.3 所示，

| | Sno | Sname | Ssex | Sage | Sdept | Scholarship |
|---|-----------|-------|------|------|-------|-------------|
| 1 | 200215121 | 李勇 | 男 | 28 | CS | 否 |
| 2 | 200215122 | 刘晨 | 女 | 27 | CS | 是 |
| 3 | 200215123 | 王敏 | 女 | 20 | MA | 否 |

| | Sno | Sname | Ssex | Sage | Sdept | Scholarship |
|---|-----------|-------|------|------|-------|-------------|
| 1 | 200215121 | 李勇 | 男 | 29 | CS | 否 |
| 2 | 200215122 | 刘晨 | 女 | 28 | CS | 是 |
| 3 | 200215123 | 王敏 | 女 | 21 | MA | 否 |

| | Sno | Cno | Grade |
|---|-----------|-----|-------|
| 1 | 200215121 | 1 | 92 |
| 2 | 200215121 | 2 | 85 |
| 3 | 200215121 | 3 | 88 |
| 4 | 200215121 | 7 | 78 |
| 5 | 200215122 | 1 | 95 |
| 6 | 200215122 | 2 | 90 |
| 7 | 200215122 | 3 | 80 |

| | Sno | Sname | Ssex | Sage | Sdept | Scholarship |
|---|-----------|-------|------|------|-------|-------------|
| 1 | 200215121 | 李勇 | 男 | 29 | CS | 否 |
| 2 | 200215122 | 刘晨 | 女 | 28 | CS | 是 |

图 4.3 实验步骤（6）

(7) 用系统管理员登录，收回 U1 的所有权限

```
REVOKE SELECT,UPDATE
```

```
ON Student
```

```
FROM U1
```

(8) 用 U1 登录，查询学生表的信息

```
SELECT *
```

```
FROM Student
```

通过管理员回收用户 U1 的所有权限之后，U1 没有了 SELECT 权限，故无法进行查询。

(9) 用系统管理员登录

(10) 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。1) 首先把某个学生成绩修改为 98，查询其奖学金。2) 再把刚才的成绩修改为 80，再查询其奖学金。

触发器：

```
CREATE TRIGGER SC_TR
```

```
ON SC
```

```
AFTER UPDATE
```

```
AS
```

```
BEGIN
```

```
declare @grade int
```

```
declare @grade_del int
```

```
declare @scholar char(2)
```

```
SELECT @grade = Grade FROM inserted
```

```
IF(@grade>=95)
```

```
    BEGIN
```

```
        SELECT @scholar = Scholarship FROM Student
```

```
        IF(@scholar='否')
```

```
            BEGIN
```

```
                UPDATE Student SET Scholarship = '是' FROM Student,inserted WHERE  
Student.Sno=inserted.Sno
```

```
            END
```

```
        END
```

```
    ELSE IF(@grade<95)
```

```
        BEGIN
```

```
            declare @count int
```

```

SELECT @count = count(*) FROM SC,inserted WHERE
SC.Sno=inserted.Sno AND SC.Grade>95
IF(@count=0)
BEGIN
SELECT @grade_del = Grade FROM deleted
if(@grade_del>95)
BEGIN
UPDATE Student SET Scholarship='否' FROM Student,inserted WHERE
Student.Sno=inserted.Sno
END
END
END
END

```

1) 首先把某个学生成绩修改为 98，查询其奖学金。

```

UPDATE SC
SET Grade=98
WHERE Sno='200215121' AND Cno='1'
SELECT *
FROM Student

```

查询结果如图 4.4 所示，通过将李勇的 1 号课程分数修改为 98，奖学金自动变化为“是”，则说明了定义的触发器执行的正确性。

| | Sno | Sname | Ssex | Sage | Sdept | Scholarship |
|---|-----------|-------|------|------|-------|-------------|
| 1 | 200215121 | 李勇 | 男 | 23 | CS | 是 |
| 2 | 200215122 | 刘晨 | 女 | 22 | CS | 否 |
| 3 | 200215123 | 王敏 | 女 | 19 | MA | 否 |
| 4 | 200215125 | 张立 | 男 | 20 | IS | 否 |

图 4.4 触发器检测（1）

```

UPDATE SC
SET Grade=80
WHERE Sno='200215121' AND Cno='1'
SELECT *
FROM Student

```

查询结果如图所示，通过将李勇的 1 号课程分数修改为 80，奖学金自动变化为“否”，则说明了定义的触发器执行的正确性。

| | Sno | Sname | Ssex | Sage | Sdept | Scholarship |
|---|-----------|-------|------|------|-------|-------------|
| 1 | 200215121 | 李勇 | 男 | 23 | CS | 否 |
| 2 | 200215122 | 刘晨 | 女 | 22 | CS | 否 |
| 3 | 200215123 | 王敏 | 女 | 19 | MA | 否 |
| 4 | 200215125 | 张立 | 男 | 20 | IS | 否 |

图 4.5 触发器检测 (2)

(11) 删除刚定义的触发器

```
DROP TRIGGER SC_TR
```

(12) 定义一个存储过程计算 CS 系的课程的平均成绩和最高成绩，在查询分析器或查询编辑器中执行存储过程，查看结果。

```
CREATE PROC CS_PROC
AS
BEGIN
SELECT Cno,AVG(Grade),MAX(Grade)
FROM Student,SC
WHERE Student.Sno=SC.Sno AND Sdept='CS'
GROUP BY Cno
END
```

```
EXEC CS_PROC
```

查询结果如 4.6 所示，

| | Cno | (No column name) | (No column name) |
|---|-----|------------------|------------------|
| 1 | 1 | 93 | 95 |
| 2 | 2 | 87 | 90 |
| 3 | 3 | 84 | 88 |
| 4 | 7 | 78 | 78 |

图 4.6 实验步骤 (12)

(13) 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

```
CREATE PROC GRADE_PROC
@SSno varchar(10)
AS
BEGIN
SELECT @SSno,Sname,Cno,Grade
FROM Student,SC
WHERE Student.Sno=SC.Sno AND Student.Sno=@SSno
END
```

```
EXEC GRADE_PROC '200215121'
```

查询结果如图 4.7 所示，

| | (No column name) | Sname | Cno | Grade |
|---|------------------|-------|-----|-------|
| 1 | 200215121 | 李勇 | 1 | 92 |
| 2 | 200215121 | 李勇 | 2 | 85 |
| 3 | 200215121 | 李勇 | 3 | 88 |
| 4 | 200215121 | 李勇 | 7 | 78 |

图 4.7 实验步骤（13）

（14）把上一题改成函数。再进行验证。

```
CREATE function GRADE_FUNC(@SSno varchar(10))
RETURNS table
RETURN (SELECT Sname,Cno,Grade
        FROM Student,SC
        WHERE Student.Sno=SC.Sno AND Student.Sno=@SSno);
```

```
SELECT * FROM GRADE_FUNC('200215121');
```

查询结果如图 4.8 所示，

| | Sname | Cno | Grade |
|---|-------|-----|-------|
| 1 | 李勇 | 1 | 92 |
| 2 | 李勇 | 2 | 85 |
| 3 | 李勇 | 3 | 88 |
| 4 | 李勇 | 7 | 78 |

图 4.8 实验步骤（14）

（15）在 SC 表上定义一个完整性约束，要求成绩再 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

```
SELECT Sno,Grade
FROM SC
```

```
ALTER TABLE SC
ADD CONSTRAINT Grade CHECK(Grade>=0 AND Grade<=100)
```

```
UPDATE SC
SET Grade=120
WHERE Sno='200215121' AND Cno='1'
```

```
SELECT Sno,Grade
FROM SC
```

查询结果如图 4.9 所示，

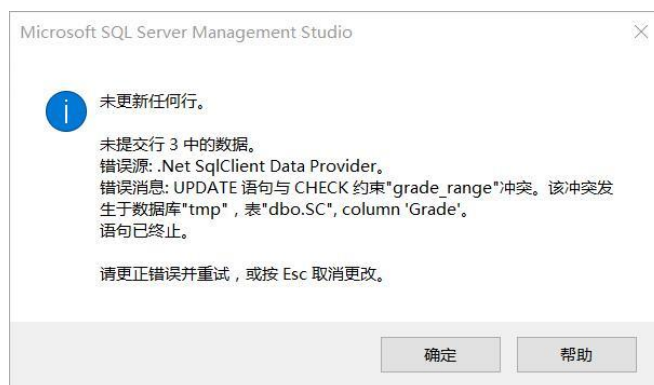


图 4.9 实验步骤（15）

4.3 任务总结

首先，学习了创建表的视图，能够通过视图直接查询表中的部分内容。我学会了创建触发器，当插入或者更新的数据违反触发器的定义时，程序会阻止更新。我学会了创建存储过程和函数，以及对存储过程和函数的调用。我学会了如何创建新的用户，并且让管理员管理新用户的权限，包括赋予权限和收回权限。最后，我通过实现定义完整性实现和创建触发器类似的功能。

其次，在这次的实验过程中，因为我对 Microsoft SQL Server Management Studio 这个软件的使用还不是很熟悉，所以不知道如何创建新用户并且让新用户能够拥有和管理员一样的表。后来通过和同学讨论以及询问路老师，最终成功创建了新用户并实现了权限的赋予和回收。在创建存储过程和函数的过程中，因为书上创建的定义和实际操作有一定的偏差，所以我通过在网上查阅资料，在学会了创建存储过程和函数的语法后，完成了任务。

5 实验任务四 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。

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

1、与数据库建立连接

- (1) 在 python 中引入库 pymssql

```
import pymssql
```

- (2) 设置登录用户名和密码

```
userName = 'sa'
```

```
passWord = '123456'
```

- (3) 与数据库建立连接并获取 cursor，其中第一个参数是服务器名，第二个参数是用户名，第三个参数是数据库名，第四个参数是使用的语言。

```
conn = pymssql.connect(serverName,userName,passWord,"S_T1",charset='utf8')
```

```
cursor = conn.cursor()
```

2、系统功能实现

- (1) 新生入学信息增加，学生信息修改。

如果选择为“学生信息增加”，则输入学生的学号、姓名、性别、年龄、院系。然后编写 SQL 语句，通过 `cursor.execute(sql)` 执行 SQL 语句，通过 `conn.commit()` 提交执行结果，具体实现过程为：

if(choice=='学生信息增加'):

```
message = ['学号', '姓名', '性别', '年龄', '院系']
```

```
s = t.multenterbox('输入学生信息', '学生信息增加', message)
```

```
sno=s[0]
```

```
sname=s[1]
```

```
ssex=s[2]
```

```
sage=s[3]
```

```
sdept=s[4]
```

```
sql = "INSERT INTO Student
```

```
VALUES('%s','%s','%s','%s','%s','%s')"%(sno,sname,ssex,sage,sdept,'否')
```

```
cursor.execute(sql)#执行
```

```
conn.commit()#提交
```

(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)。

输入信息和执行 SQL 语句与上边类似

增加课程的 SQL 语句为:

```
sql="UPDATE Student
```

```
SET Sname='%s',Ssex='%s',Sage='%s',Sdept='%s',Scholarship='%s'
```

```
WHERE Sno='%s' " %(sname,ssex,sage,sdept,scholarship,sno)
```

修改课程的 SQL 语句为:

```
sql= "UPDATE Course SET Cname='%s',Cpno='%s',Ccredit='%s'
```

```
WHERE Cno='%s' "%(cname,cpno,credit,cno)
```

删除课程的 SQL 语句为:

```
sql= "DELETE FROM Course WHERE Cno ='%s' "%(cno)
```

(3) 录入学生成绩, 修改学生成绩。

录入学生成绩的 SQL 语句为:

```
sql= "INSERT INTO SC VALUES('%s','%s','%s')"%(sno,cno,grade)
```

修改学生成绩的 SQL 语句为:

```
sql= "UPDATE SC SET Grade='%s' WHERE Sno='%s' AND
```

```
Cno='%s' "%(grade,sno,cno)
```

(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。

获取平均成绩、最好成绩、最差成绩:

```
sql= "SELECT AVG(SC.Grade),MAX(SC.Grade),MIN(SC.Grade)
```

```
FROM SC,Student
```

```
WHERE SC.Sno=Student.Sno AND Student.Sdept='%s' "%(sdept)
```

通过 fetchall()获得所有的记录列表保存在 results 中

```
results=cursor.fetchall()
```

获取该院系的人数:

```
sql= "SELECT COUNT(DISTINCT SC.Sno) FROM SC,Student
```

```

WHERE SC.Sno=Student.Sno AND Student.Sdept='%s'%(sdept)
cursor.execute(sql)
sum=cursor.fetchall()
获取该院系优秀即成绩大于 80 的人数:
sql= "SELECT COUNT(DISTINCT SC.Sno) FROM SC,Student
WHERE SC.Sno=Student.Sno AND Student.Sdept='%s' AND
SC.Grade>80"%(sdept)
cursor.execute(sql)
good=cursor.fetchall()
求出优秀率 percent:
percent=int((good[0])[0])/int((sum[0])[0])
获取不及格人数:
sql= "SELECT COUNT(DISTINCT SC.Sno) FROM SC,Student
WHERE SC.Sno=Student.Sno AND Student.Sdept='%s' AND
SC.Grade<60"%(sdept)
cursor.execute(sql)
bad=cursor.fetchall()
(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。
按系对学生成绩进行排名
sql= "SELECT Student.Sno,Student.Sname,Course.Cno,Course.Cname,SC.Grade
FROM SC,Student,Course
WHERE SC.Sno=Student.Sno AND SC.Cno=Course.Cno AND
Student.Sdept='%s'ORDER BY Grade DESC"%(sdept)
将结果保存在 results 中, 并且通过循环, 将输出结果保存到字符串 st 中
try:

```

```

    cursor.execute(sql)
    results = cursor.fetchall()
    st="学号\t姓名\t课程号\t课程名\t成绩\n"
    for row in results:
        sno = row[0]
        sname = row[1]
        cno = row[2]
        cname = row[3]
        grade = row[4]
    st=st+sno+"\t"+sname+"\t"+cno+"\t"+cname+"\t"+str(grade)+"\n"
    t.msgbox(st,'按系排名','YES')
except:
    t.msgbox("Error: unable to fetch data",'按系排名','YES')

```

(6) 输入学号, 显示该学生的基本信息和选课信息。

获取指定学号的所有信息：

```
sql="SELECT
Student.Sno,Student.Sname,Student.Ssex,Student.Sage,Student.Sdept,Student.Scholar
ship,Course.Cno,Course.Cname,SC.Grade
FROM SC,Student,Course
WHERE SC.Sno=Student.Sno AND SC.Cno=Course.Cno AND
Student.Sno='%s'"%(sno)
```

将结果保存在 results 中，并且通过循环，将输出结果保存到字符串 st 中用来输出。

try:

```
        cursor.execute(sql)
        results = cursor.fetchall()
        st="学号\t姓名\t性别\t年龄\t院系\t奖学金\t课程号\t课程名
\t成绩\n"
```

```
        for row in results:
```

```
            sno = row[0]
            sname = row[1]
            ssex = row[2]
            sage = row[3]
            sdept = row[4]
            scholarship=row[5]
            cno=row[6]
            cname=row[7]
            grade=row[8]
```

```
st=st+sno+"\t"+sname+"\t"+ssex+"\t"+str(sage)+"\t"+sdept+"\t"+scholarship+"\t"+c
no+"\t"+cname+"\t"+str(grade)+"\n"
```

```
        t.msgbox(st,'按学号查询','YES')
```

```
    except:
```

```
        t.msgbox("Error: unable to fetch data",'按学号查询','YES')
```

3、设计图形界面

(1) 连接库 easygui 并使它作用于 t

```
import easygui as t
```

(2) 设计选择按钮

先将要显示的内容存储在一个列表中，比如学生主选择页面：

```
list=('学生信息增加','学生信息修改','课程信息维护','录入/修改成绩','按系统
计','按系排名','按学号查询','退出')
```

通过调用 buttonbox 来显示内容，并且将选择结果返回到 choice 中：

choice=t.buttonbox('请选择操作','数据库实验',list)

效果如图 5.1 所示:



图 5.1 系统初始界面

(3) 设计多输入文本框

先将需要输入的信息名称保存在列表 message 中，比如插入学生信息：

```
message = ['学号','姓名','性别','年龄','院系']
```

通过调用 multenterbox 来显示输入项并且将结果返回到 s，其中 s[i]代表第 i 个返回值：

```
s = t.multenterbox('输入学生信息','学生信息增加', message)
```

效果如图 5.2 所示:

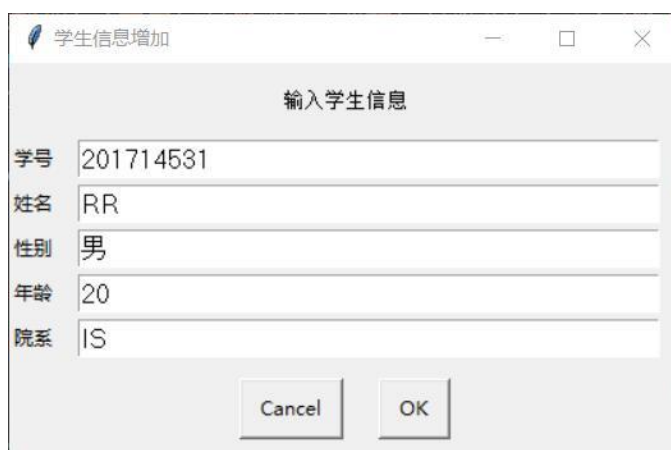


图 5.2 输入学生信息

4、执行结果验证

(1) 插入学生信息

| | Sno | Sname | Ssex | Sage | Sdept | Scholarship |
|----|-----------|-------|------|------|-------|-------------|
| | 200215121 | 李勇 | 男 | 20 | CS | 否 |
| | 200215122 | 刘晨 | 女 | 19 | CS | 否 |
| | 200215123 | 王敏 | 女 | 18 | MA | 否 |
| | 200215125 | 张立 | 男 | 19 | IS | 否 |
| | 201714535 | 李华 | 男 | 18 | CS | 是 |
| ▶* | NULL | NULL | NULL | NULL | NULL | NULL |

图 5.3 插入前学生信息

插入学生李明的信息如图 5.4 所示。

图 5.4 插入的信息

插入后学生 Student 表如图 5.5 所示。

| | Sno | Sname | Ssex | Sage | Sdept | Scholarship |
|---|-----------|-------|------|------|-------|-------------|
| ▶ | 200215121 | 李勇 | 男 | 20 | CS | 否 |
| | 200215122 | 刘晨 | 女 | 19 | CS | 否 |
| | 200215123 | 王敏 | 女 | 18 | MA | 否 |
| | 200215125 | 张立 | 男 | 19 | IS | 否 |
| | 201714535 | 李华 | 男 | 18 | CS | 是 |
| | 201814531 | 李明 | 男 | 20 | IS | 否 |
| * | NULL | NULL | NULL | NULL | NULL | NULL |

图 5.5 插入后的信息

(2) 修改学生成绩

修改成绩前学号为“200215121”的李勇 1 号课程成绩为 90，如图 5.6 所示。

| | Sno | Cno | Grade |
|---|-----------|------|-------|
| ▶ | 200215121 | 1 | 90 |
| | 200215121 | 2 | 85 |
| | 200215121 | 3 | 88 |
| | 200215122 | 2 | 90 |
| | 200215122 | 3 | 80 |
| | 201714535 | 1 | 100 |
| * | NULL | NULL | NULL |

图 5.6 修改前的成绩

将该成绩修改为 100，如图 5.7 所示。

图 5.7 修改成绩

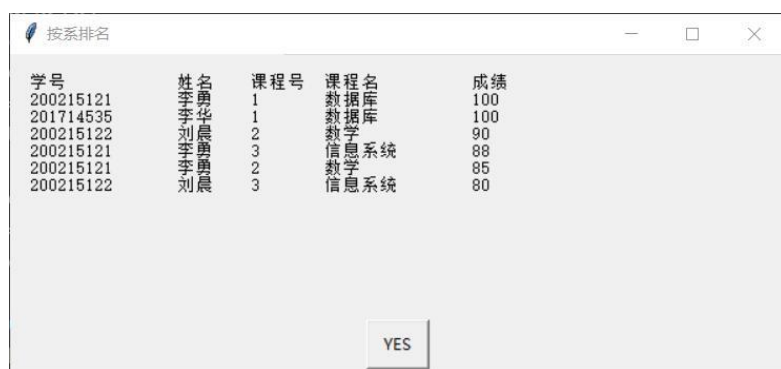
修改后 SC 表如图 5.8 所示。

| | Sno | Cno | Grade |
|---|-----------|------|-------|
| ▶ | 200215121 | 1 | 100 |
| | 200215121 | 2 | 85 |
| | 200215121 | 3 | 88 |
| | 200215122 | 2 | 90 |
| | 200215122 | 3 | 80 |
| | 201714535 | 1 | 100 |
| * | NULL | NULL | NULL |

图 5.8 修改后的 SC 表

(3) 按系排名

输入系名 CS 后结果如图 5.9 所示。



| 学号 | 姓名 | 课程号 | 课程名 | 成绩 |
|-----------|-----|-----|------|-----|
| 200215121 | 李勇华 | 1 | 数据库 | 100 |
| 201714535 | 李华 | 1 | 数据库 | 100 |
| 200215122 | 刘晨 | 2 | 数学 | 90 |
| 200215121 | 李勇 | 3 | 信息系统 | 88 |
| 200215121 | 李勇 | 2 | 数学 | 85 |
| 200215122 | 刘晨 | 3 | 信息系统 | 80 |

图 5.9 按系排名

5.3 任务总结

这次的实验综合性比较强，难度也比较大。通过这次实验，我学会了如何用 python 连接数据库，通过 import 语句连接 pymssql 库，然后通过直接调用 connect 就可以直接连接到数据库，非常的方便。我学会了如何使用 python 编写数据库语言，编写完 sql 语句后，直接通过 cursor.execute(sql)便可以执行，然后通过 conn.commit()进行提交。我还学会了使用 easygui 编写图形化界面，只需要简单的调用 multenterbox 和 buttonbox 便可以实现，与通过 MFC 编写界面相比简单了很多。

在这次实验中，我认为比较难得不是编写代码，而是连接数据库，下载和配置 pymssql 用了很久，最终在同学的帮助下能够成功连接数据库。还有一个难点是如何将数据库中的信息格式化输出，经过思考，我先把所有需要输出的内容存储在一个字符串中，然后通过 easygui 一次性将结果输出到图形化界面中。

6 课程总结

第一次实验和第二次实验让我比较熟练的掌握了数据库的基本语句，同时还熟悉了数据库的嵌套查询操作，知道用什么方式去进行思考解决问题。同时，也让我学会了如何使用 **Microsoft SQL Server Management Studio** 这个软件。第三次实验的语句操作与书上的不同地方太多，而且微软的官方文档也没有清楚的进行说明。我通过在网上查阅了很多资料，发现很多都不能实际运用，最后几位同学一起合作，最终找到有关触发器和函数的使用方法。我认为视图一旦创建完成，就和一个基本表类似。实现了创建触发器、创建存储过程、编写函数、创建新用户并且控制新用户的权限、定义完整性。最后一次实验的综合性偏强，但是只要知道对应编程语言的数据库接口，其实写起来也非常快。第一个难点是数据库的连接，不知道为什么 **Microsoft SQL Server** 特别难连接，而 **MySQL** 却十分容易，所以最后一个实验我果断换成了 **MySQL** 进行操作。其他的图形化界面和输出都是利用接口在做，并没有什么难度。

通过数据库的四次试验，我对数据库编程有了基本的了解，同时也感受到了数据库语言的强大之处。现在掌握 **SQL** 语言在工作上也很占优势，基本上每个部门都会有数据库的建立和管理，所以学好 **SQL** 语言对我来说十分重要，今后我会花更多的时间掌握 **SQL** 语言。