



华中科技大学

数据库系统概论实验报告

姓 名:

学 院: 网络空间安全学院

专 业: 网络空间安全

班 级:

学 号:

指导教师:

分数	
教师签名	

2023 年 11 月 27 日

目 录

1 课程任务概述	1
2 实验一 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	4
3 实验二 SQL 的复杂操作	5
3.1 任务要求	5
3.2 完成过程	5
3.3 任务总结	8
4 实验三 SQL 的高级实验	9
4.1 任务要求	9
4.2 完成过程	9
4.3 任务总结	19
5 实验四 数据库设计	20
5.1 任务要求	20
5.2 完成过程	20
5.3 任务总结	27
6 课程总结	28
6.1 主要工作	28
6.2 心得体会	28
6.3 未来展望	28

1 课程任务概述

本次数据库实验课涵盖了四项主要的实验任务，包含：数据库定义与基本操作（关联到数据库建立、基本的查询命令）、SQL 的复杂操作（涉及到高级 SQL 检索、数据插入与数据更新办法）、SQL 的高级实验（包涵权限配置、函数、触发机制及存储过程等内容）以及数据库设计（借助高级编程语言建立数据库前端，执行数据库查询和管理任务）。

本实验完成的环境：

Windows 11 家庭中文版

MySQL 8.0

Navicat Premium 16

PyCharm 2023.3.1

Python 3.11.5 | packaged by Anaconda, Inc.

ER 图和关系模型：

数据库 s_t 共有三张表分别是 SC，Student，Course，ER 图如下图 1.1 所示。

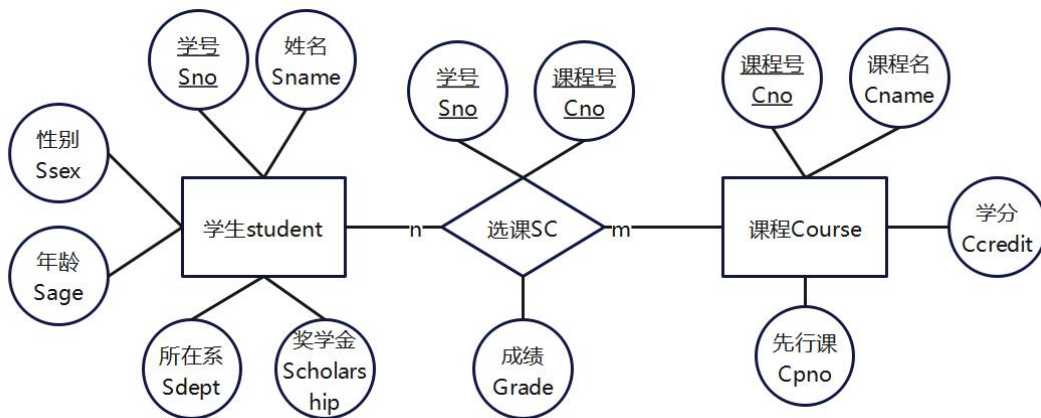


图 1.1 ER 图

关系模型：

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

2 实验一 数据库定义与基本操作

2.1 任务要求

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

2.2 完成过程

完成 MySQL 的安装和 navicat 的安装，创建数据库 s_t,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 语句和运行结果：

```
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 语句和运行结果：

```
mysql> select * from Student where Sdept in ( '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（≥90 分）或不及格的学生的学号、课程号及成绩。

SQL 语句和运行结果：

```
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 语句和运行结果:

```
mysql> select Sname,Ssex,Sage from Student where Sage not between 19 and 20;
+-----+-----+-----+
| Sname | Ssex | Sage |
+-----+-----+-----+
| 王敏  | 女   | 18   |
+-----+-----+-----+
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 语句和运行结果:

```
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 语句和运行结果:

```
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 语句和运行结果:

```
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 语句和运行结果:

```
mysql> select max(Grade),min(Grade),avg(Grade) from SC where Sno like '200215121' ;
+-----+-----+-----+
| max(Grade) | min(Grade) | avg(Grade) |
+-----+-----+-----+
| 92         | 85         | 88.3333    |
+-----+-----+-----+
1 row in set (0.00 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 语句和运行结果：

```
mysql> select Sno,Grade from SC where Cno like '2' order by grade;
+-----+-----+
| 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 语句和运行结果

```
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 语句和运行结果：

```
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 (0.00 sec)
```

图 2.11 实验 1 扩展练习 10 思考

2.3 任务总结

在这次实验中，我学习并实践了基本的 SQL 语句，包括创建表、删除表、修改表，以及使用 LIKE 子句进行模糊查询、利用 ORDER BY 子句对结果排序，以及使用 GROUP BY 子句进行分组查询等。需要注意的是，MySQL 默认对数据库名和数据表名不区分大小写，但对数据表中的属性（列名）区分大小写。另外，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。通过动手实践，我加深了对 SQL 语句的理解，这让我受益匪浅。

3 实验二 SQL 的复杂操作

3.1 任务要求

- (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 语句和运行结果：

```
mysql> select Course.Cno,Course.Cname, SC.Sno,SC.Grade from Course, SC where Course.Cno = SC.Cno;
```

Cno	Cname	Sno	Grade
1	数据库	200215121	92
2	数学	200215121	85
3	信息系统	200215121	88
2	数学	200215122	90
3	信息系统	200215122	80
4	操作系统	200215123	NULL
6	数据处理	200215125	NULL

7 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。

(1) 嵌套查询

SQL 语句 1 和运行结果：

```
mysql> select Sno, Sname, Sage from Student S1 where exists (select * from Student S2 where S2.Sage=
S1.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 3.2 实验 2 扩展练习 2.1

(2) 自身连接

SQL 语句 2 和运行结果：


```
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 3.3 实验 2 扩展练习 2.2

(3) EXIST 谓词

SQL 语句 3 和运行结果:

```
mysql> select Sno, Sname, Sage from Student S1 where exists (select * from Student S2 where S2.Sage=S1.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 3.4 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(80-89 分)的所有学生的学号和姓名。

SQL 语句和运行结果:

```
mysql> select Sno, Sname from Student where exists (select * from SC where Student.Sno = SC.Sno and Cno = 3 and Grade between 80 and 89);
```

Sno	Sname
200215122	刘晨
200215121	李勇

```
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 语句和运行结果:

```
mysql> select Course.Cno, Course.Cname from Student, Course, SC
-> where Student.Sno = 200215122 and SC.Sno = Student.Sno 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?

说明: 在“SELECT”子句中加上属性“Grade”即可。

SQL 语句和运行结果:

```
mysql> select Course.Cno, Course.Cname, SC.Grade from Student, Course, SC
-> where Student.Sno = 200215122 and SC.Sno = Student.Sno 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。

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。

SQL 语句和运行结果：

```
mysql> SELECT Sno,Cno from SC x WHERE Grade+5<(SELECT AVG(Grade) FROM SC y WHERE x.Sno=y.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 语句和运行结果：

```
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 语句和运行结果：

```
mysql> SELECT Sname,Sdept  
-> FROM Student  
-> WHERE EXISTS  
-> (SELECT *  
-> FROM SC  
-> WHERE SC.Sno=Student.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。

SQL 语句和运行结果：

```
mysql> UPDATE Student  
-> SET Sage= Sage+2  
-> WHERE Sno in (select Sno from SC where Grade between 80 and 89);  
Query OK, 2 rows affected (0.01 sec)  
Rows matched: 2 Changed: 2 Warnings: 0  
  
mysql> select Student.Sno,Student.Sname,Student.Sage from Student, SC where SC.Sno = Student.Sno and  
Grade between 80 and 89;  
+-----+-----+-----+  
| Sno      | Sname  | Sage |  
+-----+-----+-----+  
| 200215121 | 李勇   | 24   |  
| 200215121 | 李勇   | 24   |  
| 200215122 | 刘晨   | 23   |  
+-----+-----+-----+  
3 rows in set (0.00 sec)
```

图 3.11 实验 2 扩展练习 8

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。

SQL 语句和运行结果：

```
mysql> INSERT
-> INTO Course(Cno,Cname)
-> VALUES ("8","C语言");
Query OK, 1 row affected (0.01 sec)

mysql> INSERT
-> INTO Course(Cno,Cname)
-> VALUES ("9","人工智能");
Query OK, 1 row affected (0.01 sec)

mysql> select Cno,Cname from Course where Cno in("8","9");
+-----+-----+
| Cno | Cname |
+-----+-----+
| 8   | C语言 |
| 9   | 人工智能 |
+-----+-----+
2 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。

SQL 语句和运行结果：

```
mysql> DELETE
-> FROM Course
-> WHERE Cname="人工智能";
Query OK, 1 row affected (0.01 sec)

mysql>
mysql> select Cno,Cname from Course;
+-----+-----+
| Cno | Cname |
+-----+-----+
| 1   | 数据库 |
| 2   | 数学 |
| 3   | 信息系统 |
| 4   | 操作系统 |
| 5   | 数据结构 |
| 6   | 数据处理 |
| 7   | java |
| 8   | 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

在这次实验中，我学习并实践了 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等。比实验一做起来稍难一点，但是结合实验指导书和所学内容最终也顺利完成，我也通过查阅资料搞清楚了左连接、右连接等概念，同时加深和巩固了课本知识的学习掌握。

4 实验三 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
- (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';

运行结果(可以在数据库左侧视图栏看到对应视图):

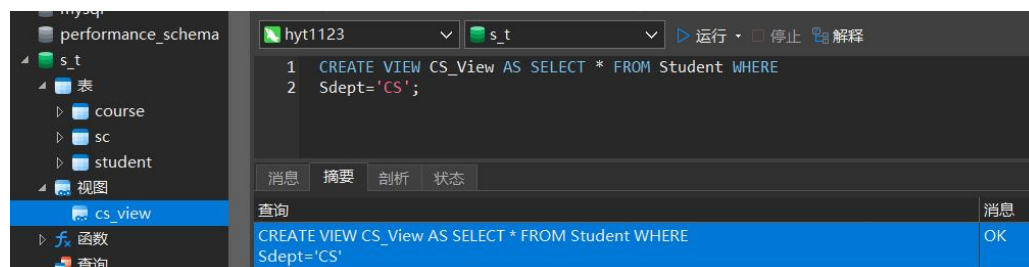


图 4.1 创建视图 CS_View

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.Sname FROM CS_View natural join SC WHERE Cno=1;

运行结果:

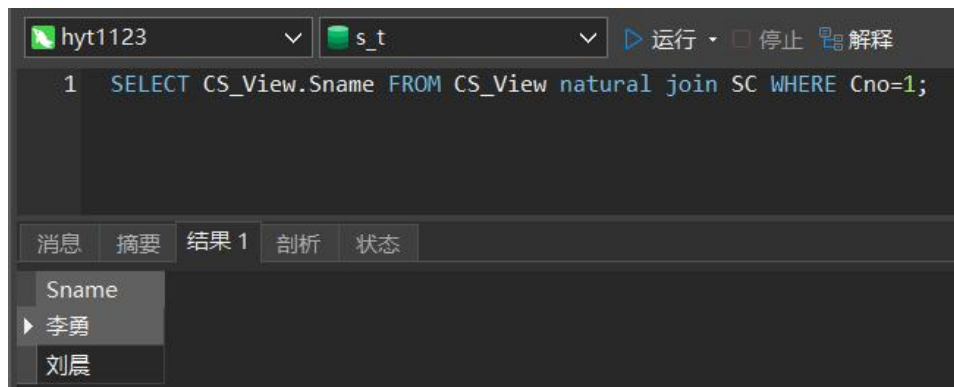


图 4.2 运行结果

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 Sno, Sname, Ssex, Sage, Sdept, Scholarship FROM Student natural join SC WHERE Sdept='IS' AND Grade>80;
运行结果(可以在数据库左侧视图栏看到对应视图):

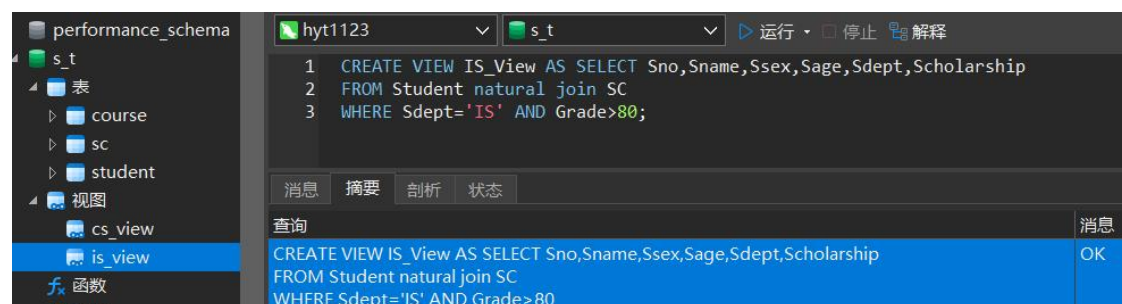


图 4.3 创建视图 IS_View

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;
说明: 由于 IS 系没有大于 80 的学生中, 因此视图为空。
运行结果:

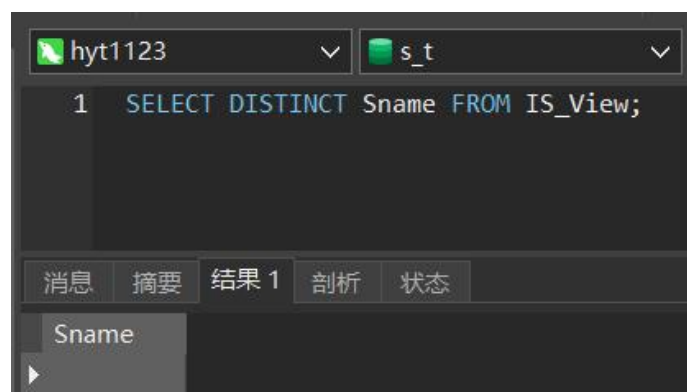


图 4.4 运行结果

4.2.5 删除视图。

SQL 语句:DROP VIEW IS_View;

运行结果(可以在数据库左侧视图栏看到对应视图已被删除):

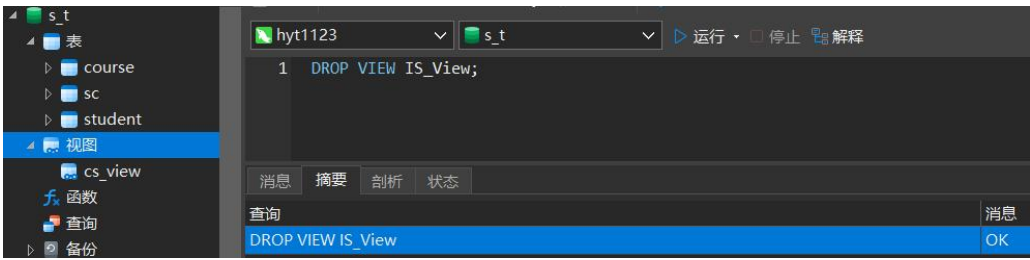


图 4.5 运行结果

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

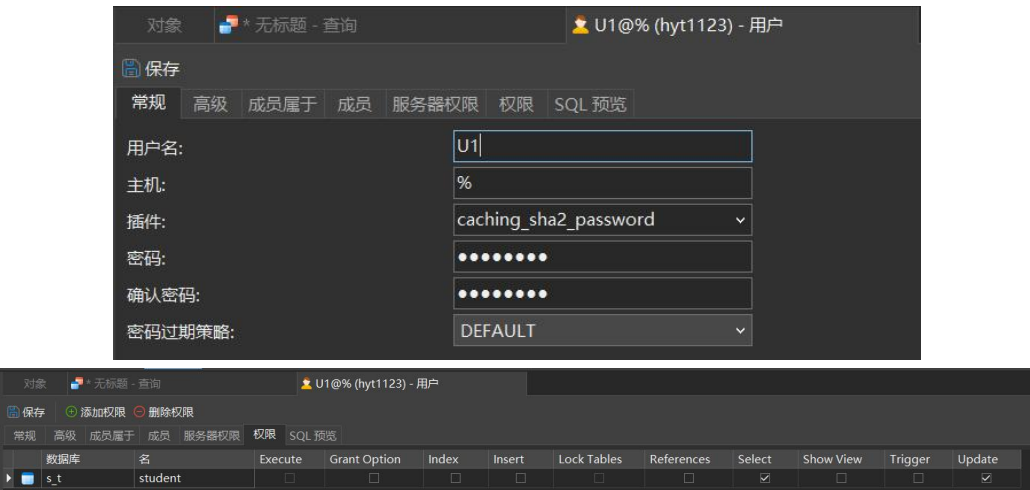


图 4.6 创建用户 U1 并授权

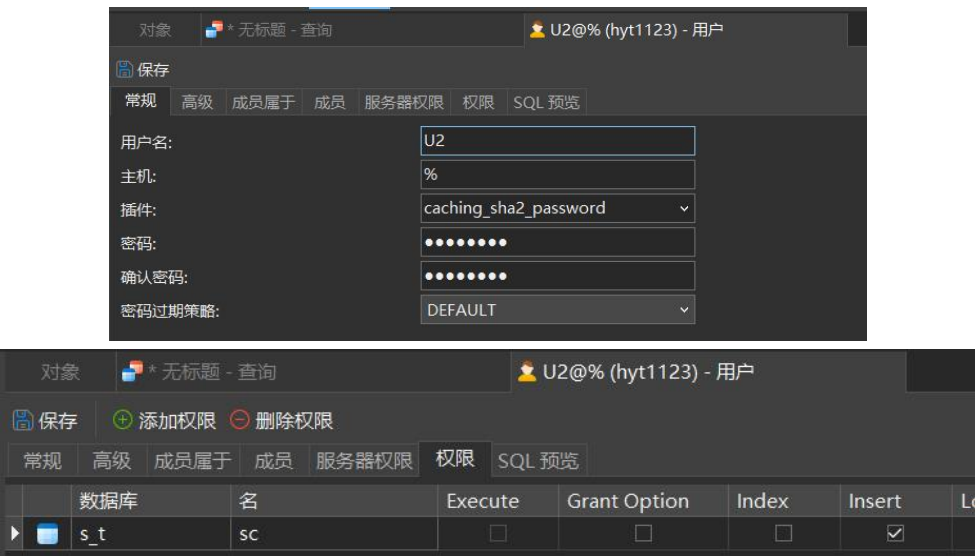


图 4.7 创建用户 U2 并授权

(2) 用 U1 登录，
 查询学生表的信息；
 SQL 语句:select * from student;
 运行结果:

Sno	Sname	Ssex	Sage	Sdept	Scholarship
200215121	李勇	男	24	CS	否
200215122	刘晨	女	23	CS	否
200215123	王敏	女	18	MA	否
200215125	张立	男	19	IS	否

图 4.8 运行结果

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:

Sno	Sname	Ssex	Sage	Sdept	Scholarship
200215121	李勇	男	25	CS	否
200215122	刘晨	女	24	CS	否
200215123	王敏	女	19	MA	否
200215125	张立	男	20	IS	否

图 4.9 运行结果

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:

消息	摘要	状态
delete from student where Sdept='IS'	1142 - DELETE command denied to user 'U1'@'localhost' for table 'student'	0.002s

图 4.10 运行结果

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:

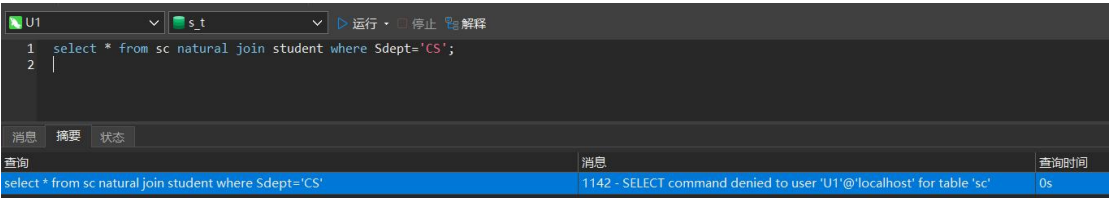


图 4.11 运行结果

(2) 用 U2 登录,

在 SC 表中插入 1 条记录 ('200215122' , '1' , 75);

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

运行结果:

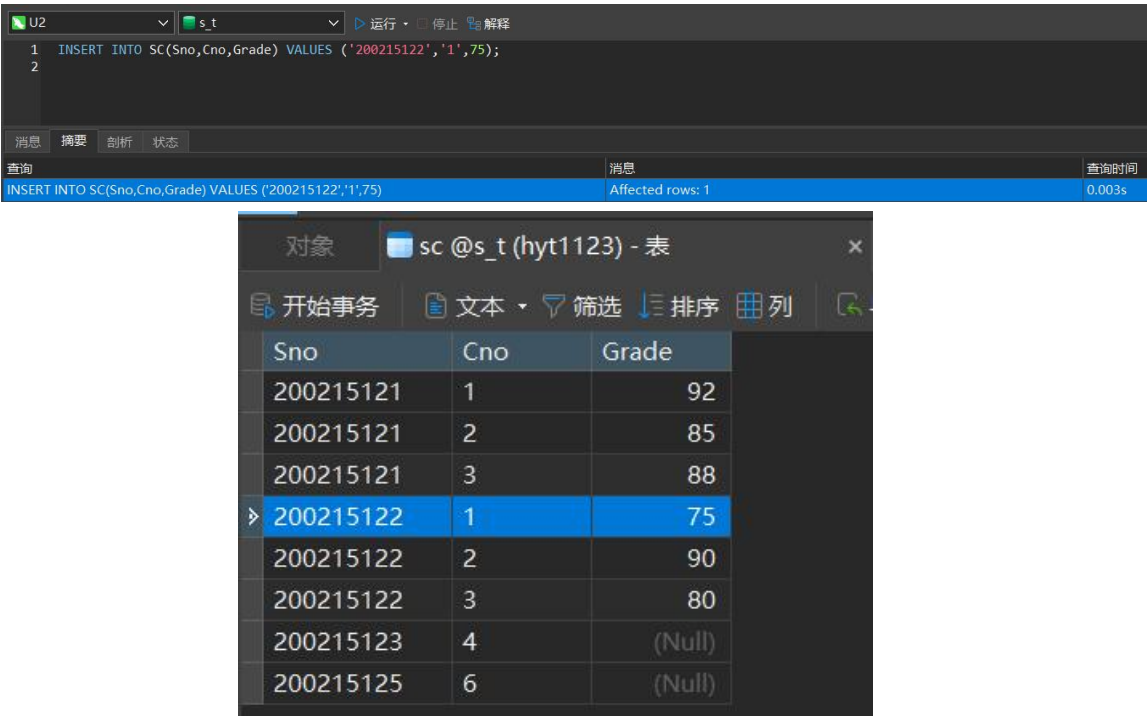


图 4.12 运行结果

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:

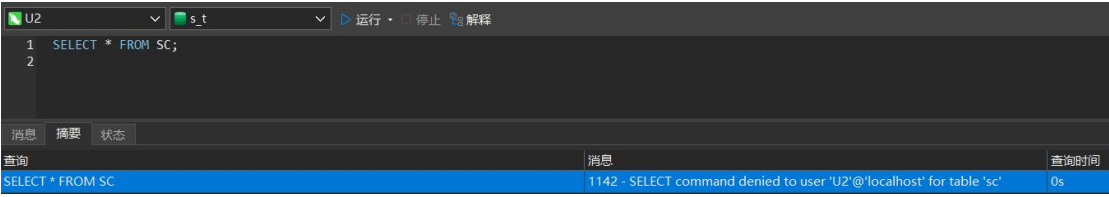


图 4.13 运行结果

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:

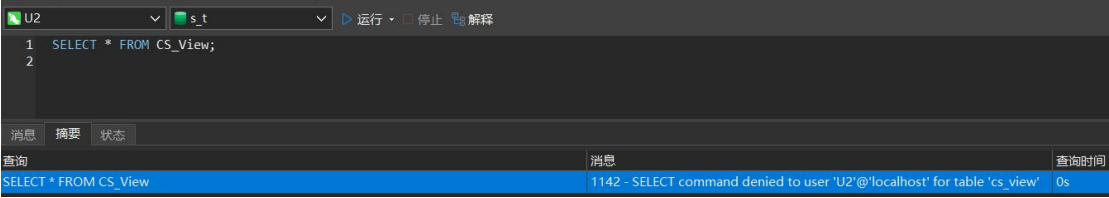


图 4.14 运行结果

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（可见 U1 权限栏为空）：

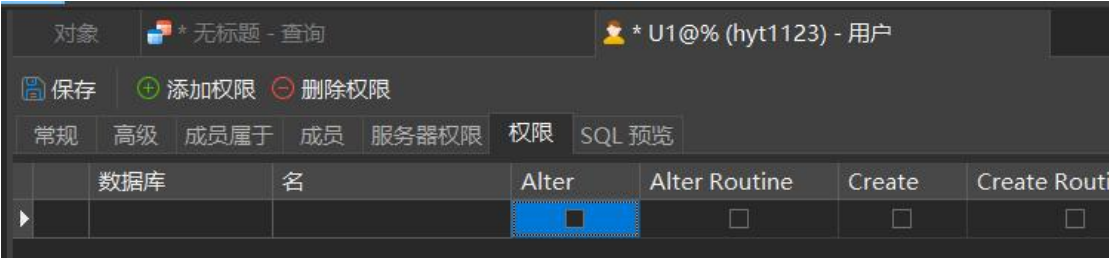


图 4.15 运行结果

4.2.8 用 U1 登录，查询学生表的信息。

SQL 语句:use s_t;

说明：由于当权限撤回后 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 数据库时就已经被拒绝了。

运行结果:

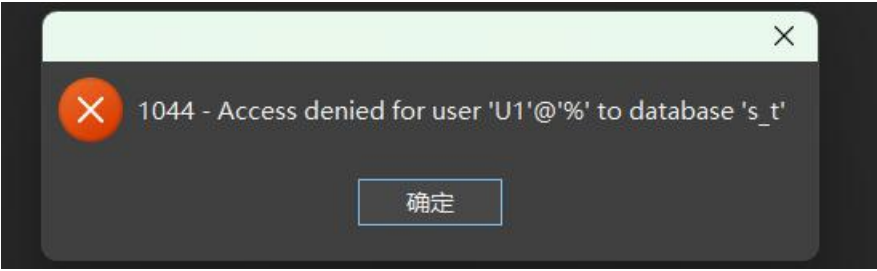


图 4.16 运行结果

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 t_sc after update on SC
```

```
for each row
```

```
begin
```

```
if(new.grade>=95) then update student set scholarship='是' where Sno=new.Sno and  
scholarship='否';
```

```
elseif(new.grade<95 and (select max(grade) from SC where Sno=new.Sno)<95 and  
old.grade>=95) then update student set scholarship='否' where Sno=new.Sno;
```

```
end if;
```

```
end
```

运行结果:

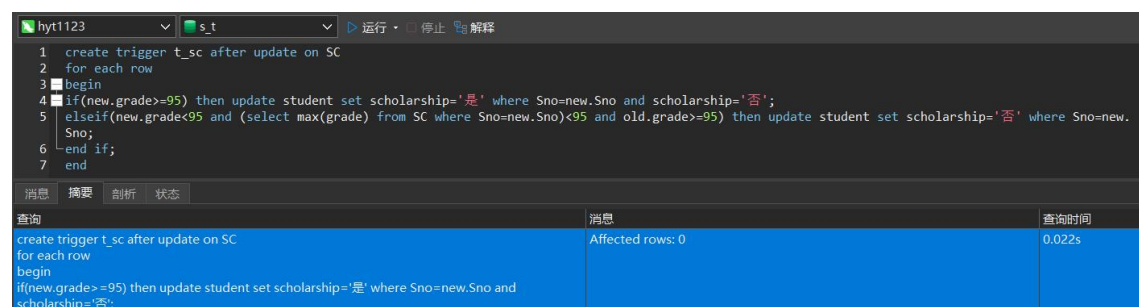


图 4.17 运行结果

(1) 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 Student 表和 SC 表，以供对照，要修改的学生为学号 200215121，课程号为 1 的成绩。

Sno	Sname	Ssex	Sage	Sdept	Scholarship
200215121	李勇	男	25	CS	否

Sno	Cno	Grade
200215121	1	92

图 4.18 未修改的 Student 表和 SC 表

SQL 语句:

```
UPDATE SC SET Grade=98 WHERE Sno='200215121' AND Cno='1';
```

```
SELECT sno,sname,scholarship FROM student WHERE Sno=200215121;
```

运行结果:

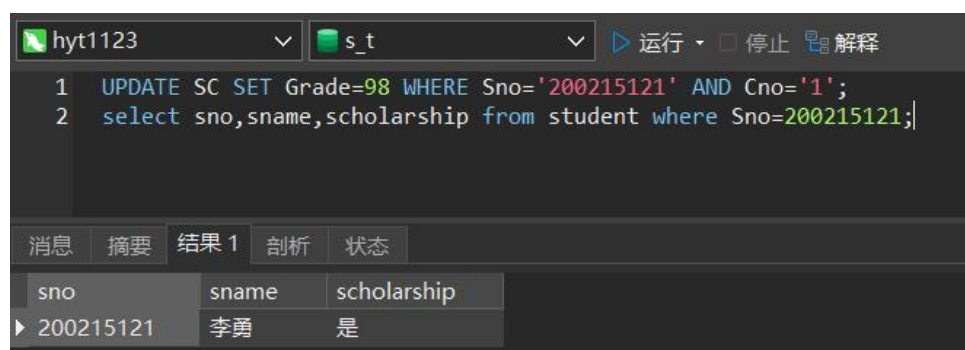


图 4.19 scholarship 属性变为 ‘是’

(2) 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';
```

```
SELECT sno,sname,scholarship FROM student WHERE Sno=200215121;
```

运行结果（scholarship 属性变为‘否’）：

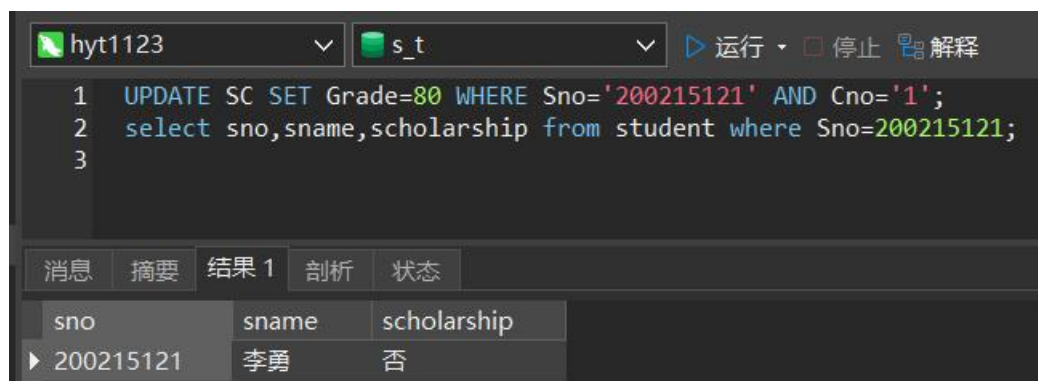


图 4.20 运行结果

4.2.10 删除刚定义的触发器。

SQL 语句：drop trigger t_sc;

运行结果：

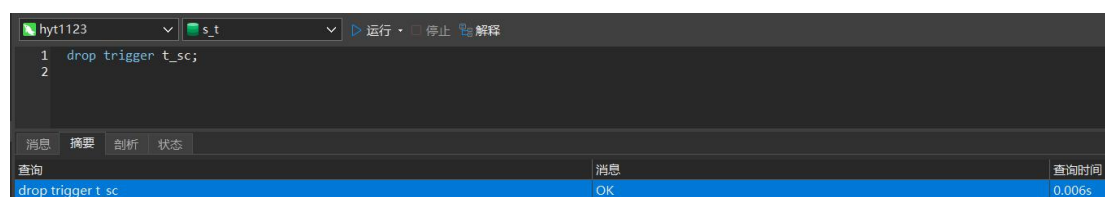


图 4.21 运行结果

4.2.11 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。

SQL 语句：

```
CREATE PROCEDURE cs_avg_max()
```

```
BEGIN
```

```
declare csavg float;
```

```
declare csmax int;
```

```
select avg(grade) into csavg from sc where sno in (select sno from student where sdept='cs');
```

```
select max(grade) into csmax from sc where sno in (select sno from student where sdept='cs');
```

```
select csavg,csmax;
```

```
END;
```

```
CALL cs_avg_max();
```

运行结果：

```

1 CREATE PROCEDURE cs_avg_max()
2 BEGIN
3   declare csavg float;
4   declare csmax int;
5   select avg(grade) into csavg from sc where sno in (select sno from student where sdept='cs');
6   select max(grade) into csmax from sc where sno in (select sno from student where sdept='cs');
7   select csavg,csmax;
8 END;
9 CALL cs_avg_max();

```

csavg	csmax
83	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 gra_sna(stu_no CHAR(9))
BEGIN
SELECT grade,sname FROM student natural join sc WHERE sno=stu_no;
END;
CALL gra_sna('200215122');

```

运行结果:

```

1 CREATE PROCEDURE gra_sna(stu_no CHAR(9))
2 BEGIN
3   select grade,sname from student natural join sc where sno=stu_no;
4 END;
5 CALL gra_sna('200215122');

```

grade	sname
75	刘晨
90	刘晨
80	刘晨

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 get_student_name(stu_no CHAR(9))
RETURNS CHAR(20) DETERMINISTIC
BEGIN
  DECLARE student_name CHAR(20);
  SELECT Sname INTO student_name FROM student WHERE Sno=stu_no;
  RETURN student_name;
END;

```

```
SELECT Grade, Sname FROM sc JOIN student ON sc.Sno = student.Sno
WHERE student.Sname = get_student_name('200215122');
```

说明：由于 MySQL 中的存储过程和函数主要用于返回单一的值（标量值），而并不直接支持返回表（table）这种数据类型，所以定义单一返回值来记录。

运行结果：

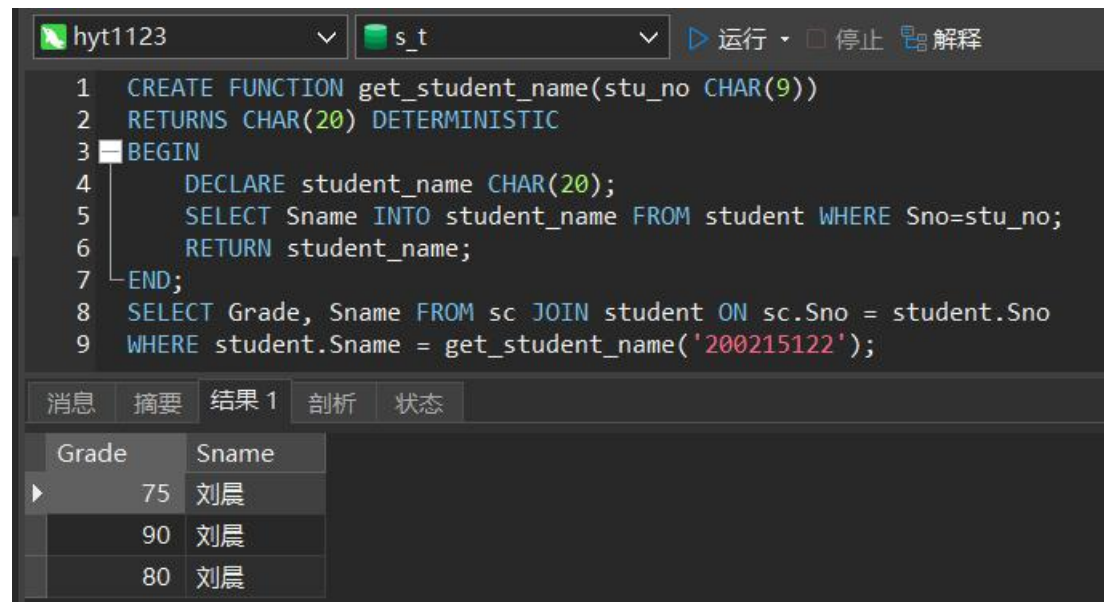


图 4.24 运行结果

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。

（1）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。

说明：将学号为‘200215122’的学生，课程号为‘2’的成绩改为 120，然后再改成原本的 90。

SQL 语句：

```
UPDATE SC SET GRADE=120/90 WHERE SNO='200215122' AND CNO=2;
SELECT * FROM SC WHERE SNO='200215122' AND CNO=2;
```

运行结果：

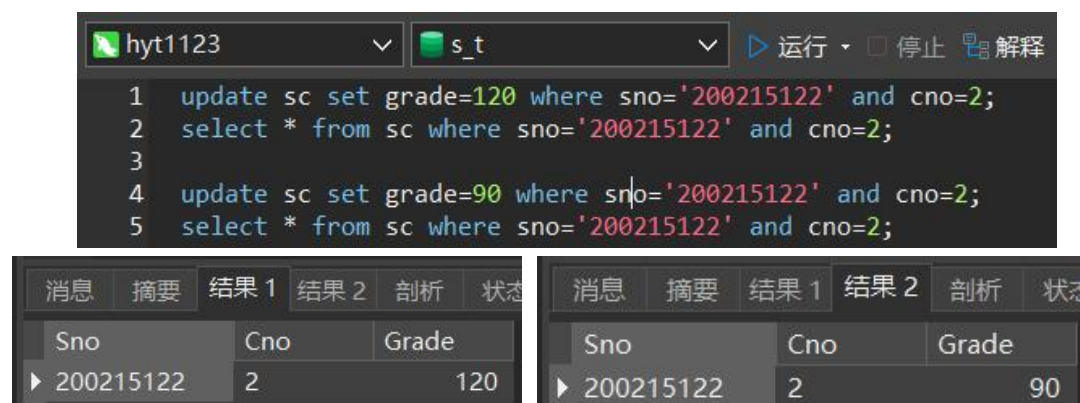


图 4.25 运行结果

（2）定义约束后，再把该学生成绩修改为 120，然后进行查询。

定义约束的 SQL 语句：ALTER TABLE SC ADD CONSTRAINT gracst
check(Grade BETWEEN 0 AND 100);
说明：定义约束后，将学号为‘200215122’的学生，课程号为‘2’的成绩
更新为 120，发现拒绝更新。
SQL 语句及运行结果：

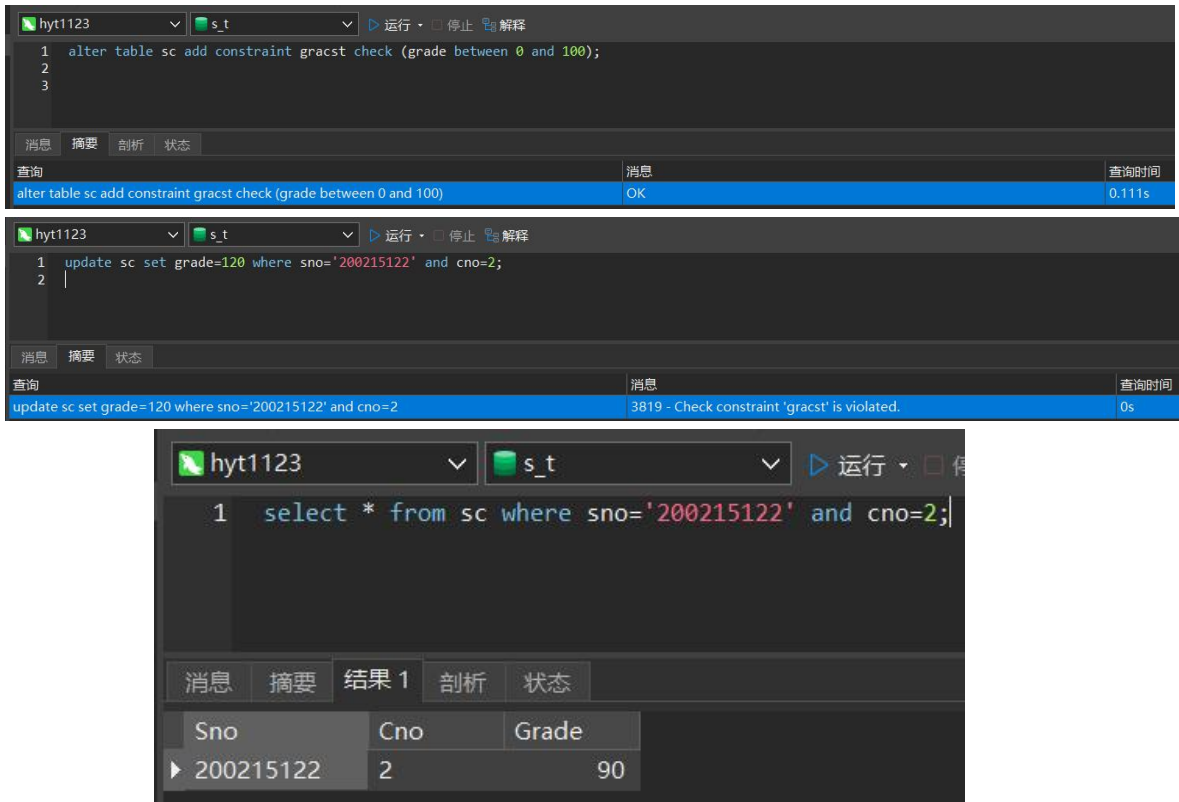


图 4.26 运行结果

4.3 任务总结

我在实验中实现了一些高级操作，如创建视图、授权、触发器、存储过程、函数以及完整性约束等。这一实验的 sql 语句段明显比之前两个实验的变得更长且复杂，一开始全部任务均通过命令行进行完成，会发现在建立触发器，存储过程以及函数段如果使用‘;’则无法完整输入完 sql 语句段，而需要使用 delimiter 语句先将结束符号修改。后面尝试全程使用可视化窗口就无需考虑到这一点，也可以直接在 navicat 进行用户的创建，授权以及收回权限，也简便了登录和连接的过程，但是我还是在报告中将需要的 sql 语句进行呈现，增强可读性。

感触比较深的是不同的数据库的语法可能有所差别，所以不能局限在课本上所学习的内容，还要结合具体使用的数据库语法来进行 sql 语句的编写。比如，在改写函数那一题，通过查阅资料了解到由于 MySQL 中的存储过程和函数主要用于返回单一的值，所以不能返回表（table）这种数据类型。故不能固步自封需要充分利用能够查询到的资料对知识进行一个广度和深度的延深。

5 实验四 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，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 conda 虚拟环境下 python3.11.5 在 pycharm 中进行编程

安装 MySQL 驱动：

打开 PyCharm 终端，执行以下命令：`pip install mysql-connector-python`

在 PyCharm 中配置数据库连接：

选择 Data Source -> MySQL 来添加一个新的 MySQL 连接。

导入库为 `import mysql.connector`

5.2.2 相关函数说明

1. main():

提供了一个执行上述功能的文本界面。用户可通过输入对应的功能序号来选择他们想要执行的数据库操作。程序确保在操作过程中如有异常发生将回滚任何未完成的事务，以保持数据库数据的一致性。程序结尾还确保了不论执行过程中是否发生错误，数据库连接和游标最终都会被正确关闭。

2. add_student(sno, sname, ssex, sage, sdept, scholarship):

这个函数用于向数据库中添加一个新的学生记录或更新已有的学生记录。如

果插入时检测到学号（sno）已存在，则会更新对应学生的信息。函数参数分别表示学号、姓名、性别、年龄、系别和奖学金情况。

3. add_course(cursor, cnx):

这个函数用于在数据库中添加一个新课程。它会提示用户输入课程号、课程名称、先修课程号以及课程学分，并将这些信息插入到数据库中。

4. update_course(cursor, cnx):

用于修改现有课程的信息。用户会被提示输入想要修改的课程号及新的课程信息（名称、先修课程号、学分），然后更新数据库。

5. delete_unenrolled_courses(cursor, cnx):

这个函数负责删除没有学生选修的课程信息。移除在学生选课记录中未被选修过的课程的数据记录，确保所有列表出的课程都有学生选修记录。

6. enter_grade(cursor, cnx):

用于为特定的学生和课程录入成绩。用户需要输入学生学号、课程号和成绩信息，这些数据将被插入到数据库中。

7. update_grade(cursor, cnx):

允许修改特定学生某门课程的成绩。用户需输入学生学号、课程号以及新的成绩，然后更新数据库记录。

8. calculate_department_statistics(cursor, cnx):

这个函数用来按系计算其中学生的成绩统计信息。它会从数据库中检索所有成绩，然后计算出平均成绩、最高成绩、最低成绩、优秀率（90 分以上为优秀）和不及格人数（60 分以下为不及格）。

9. rank_students_by_department(cursor, cnx):

该函数会按照系别对学生成绩进行排名，并输出每个学生的系别、学号、姓名、所选课程号、课程名称和成绩。

10. show_student_info(cursor, cnx, student_id):

当用户输入一个学生学号时，此函数查找并显示这个学生的基本信息（如学号、姓名、性别等）以及他/她所选的课程信息和成绩。

5.2.3 思路概述

编写语句的方法同之前所学的内容这里便不再赘述，操作总体来说分为四类，插入类操作使用，修改类操作使用，删除类操作使用以及查询类使用。

在 add_student 函数和 enter_grade 函数中使用了 ON DUPLICATE KEY UPDATE 语句，允许在尝试插入具有重复主键的记录时，更新现有的信息，以免录入重复的信息。在 delete_unenrolled_courses 函数中使用“SET Foreign_key_checks = 0”语句，用来处理外键依赖并安全删除课程信息。在 show_student_info 和 rank_students_by_department 函数中使用了 INNER JOIN 来合并学生、课程和成绩到一个查询中。在 calculate_department_statistics 函数中使

用 ROUND 函数用于将优秀率保留两位小数。当编写实际的应用程序时，应确保处理所有可能的异常情况，并且在执行数据库更新操作前进行充分的数据校验。在数据库操作完成后，需调用 commit()方法来确保更改被保存，并且在发生错误时调用 rollback()方法来撤销变更。

编写好上面的函数并确保单独均可实现功能后，然后创建一个交互式的用户界面来执行数据库相关的功能，并允许用户输入功能序号以选择执行的操作，可以通过一个简单的 while 循环和 if-elif 语句组合来实现。首先输入数据库信息，然后再通过输入一个数字 1-6 来分别实现各个功能，并且能循环执行，直到输入数字 0 则退出程序。然后将主体执行代码都放在功能函数，在主函数只需调用和传参即可。同时，为了确保数据的一致性和安全性，在执行插入、更新或删除操作时要进行正确的异常处理，并在操作完成后提交事务。

5.2.4 功能实现效果

(1) 建立数据库连接和主菜单界面。

```
D:\anaconda3\python.exe E:\python\pythonProject\database.py
请输入用户名: root
请输入密码: 123456
请输入数据库名称: s_t
数据库连接成功!

+-----+
| 请选择要执行的功能序号:          |
| 1: 新生入学信息增加, 学生信息修改 |
| 2: 课程信息维护                  |
| 3: 录入学生成绩, 修改学生成绩     |
| 4: 按系计算学生平均成绩, 成绩分布 |
| 5: 按系排名学生成绩              |
| 6: 显示学生基本信息和选课信息     |
| 0: 退出程序                      |
+-----+
请输入功能序号:
```

图 5.1 数据库连接和主菜单

(2) 新生入学信息增加，学生信息修改。

1.新生入学信息增加

```
请输入功能序号: 1
请输入学号: 202211915
请输入姓名: 林木木
请输入性别: 女
请输入年龄: 19
请输入系别: CSE
请输入奖学金情况(是/否): 是
学生信息已添加或更新
```

图 5.2 插入学生信息

对象 student @s_t (hyt1123) - 表						
Sno	Sname	Ssex	Sage	Sdept	Scholarship	
200215121	李勇	男	24	CS	否	
200215122	刘晨	女	23	CS	否	
200215123	王敏	女	18	MA	否	
200215125	张立	男	19	IS	否	
202211915	林木木	女	19	CSE	是	

图 5.3 插入后 navicat 中 Student 表

2. 学生信息修改

```

请输入功能序号: 1
请输入学号: 202211915
请输入姓名: 李木子
请输入性别: 男
请输入年龄: 20
请输入系别: CSE
请输入奖学金情况(是/否): 是
学生信息已添加或更新
  
```

图 5.4 修改学生信息

对象 student @s_t (hyt1123) - 表						
Sno	Sname	Ssex	Sage	Sdept	Scholarship	
200215121	李勇	男	24	CS	否	
200215122	刘晨	女	23	CS	否	
200215123	王敏	女	18	MA	否	
200215125	张立	男	19	IS	否	
202211915	李木子	男	20	CSE	是	

图 5.5 修改后 navicat 中 Student 表

(3) 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）。

```

请输入功能序号: 2

功能菜单:
1 - 增加新课程
2 - 修改课程信息
3 - 删除没有选课的课程信息
0 - 返回上级菜单
请输入您的选择: |
  
```

图 5.6 课程信息维护菜单

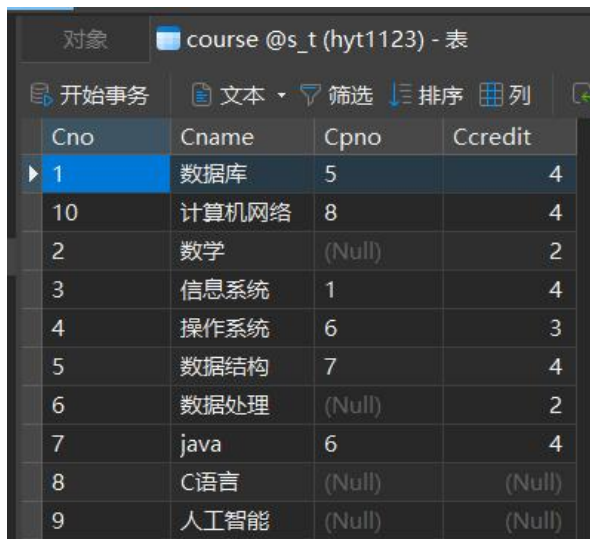
1. 增加新课程

```

请输入您的选择： 1
请输入课程号： 10
请输入课程名称： 计算机网络
请输入先修课程号（如果无先修课程，请输入NULL）： 8
请输入课程学分： 4
新增课程成功。

```

图 5.7 添加课程信息



Cno	Cname	Cpno	Ccredit
1	数据库	5	4
10	计算机网络	8	4
2	数学	(Null)	2
3	信息系统	1	4
4	操作系统	6	3
5	数据结构	7	4
6	数据处理	(Null)	2
7	java	6	4
8	C语言	(Null)	(Null)
9	人工智能	(Null)	(Null)

图 5.8 添加课程信息后的 navicat 中 course 表

2. 修改课程信息

```

请输入您的选择： 2
请输入要修改的课程号： 10
请输入新的课程名称： 大学物理
请输入新的先修课程号（如果无先修课程，请输入NULL）： NULL
请输入新的课程学分： 4
课程信息更新成功。

```

图 5.9 修改课程信息



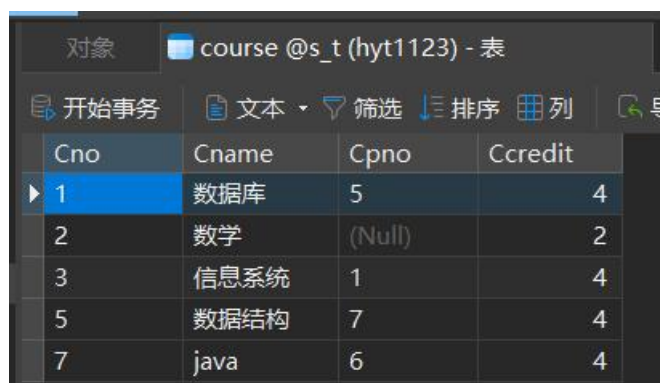
Cno	Cname	Cpno	Ccredit
1	数据库	5	4
10	大学物理	(Null)	4
2	数学	(Null)	2
3	信息系统	1	4
4	操作系统	6	3
5	数据结构	7	4
6	数据处理	(Null)	2
7	java	6	4
8	C语言	(Null)	(Null)
9	人工智能	(Null)	(Null)

图 5.10 修改课程信息后的 navicat 中 course 表

3. 删除没有选课的課程信息

```
请输入您的选择：3
未选修的课程已删除，并更新了相关的先修课程信息。
```

图 5.11 删除未选课的課程信息



Cno	Cname	Cpno	Ccredit
1	数据库	5	4
2	数学	(Null)	2
3	信息系统	1	4
5	数据结构	7	4
7	java	6	4

图 5.12 删除课程信息后的 navicat 中 course 表

(4) 录入学生成绩，修改学生成绩。

```
请输入功能序号：3

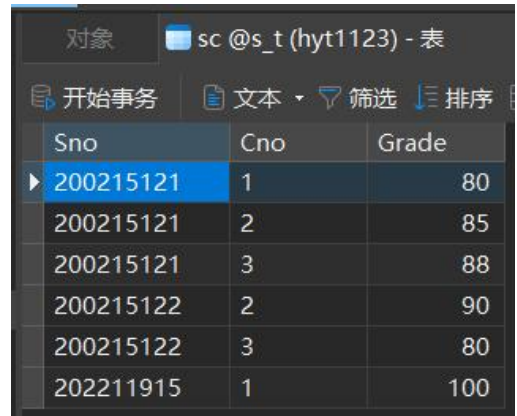
功能菜单：
1 - 录入学生成绩
2 - 修改学生成绩
0 - 返回上级菜单
请输入您的选择：|
```

图 5.13 学生成绩维护菜单

1.录入学生成绩

```
请输入您的选择：1
请输入学生学号：202211915
请输入课程号：1
请输入学生成绩：100
学生成绩录入成功。
```

图 5.14 录入学生成绩



Sno	Cno	Grade
200215121	1	80
200215121	2	85
200215121	3	88
200215122	2	90
200215122	3	80
202211915	1	100

图 5.15 录入学生成绩后 navicat 中 SC 表

2.修改学生成绩


```

请输入您的选择： 2
请输入学生学号： 202211915
请输入课程号： 1
请输入新的学生成绩： 99
学生成绩更新成功。

```

图 5.16 修改学生成绩

对象 sc @s_t (hyt1123) - 表

开始事务 文本 筛选 排序

Sno	Cno	Grade
▶ 200215121	1	80
200215121	2	85
200215121	3	88
200215122	2	90
200215122	3	80
202211915	1	99

图 5.17 修改学生成绩后 navicat 中 SC 表

(5) 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。

说明：因为成绩样本较少，所以通过功能 3 增加了几位学生的成绩。

```

请输入功能序号： 4

按系计算并输出学生成绩的平均值、最高分、最低分、优秀率和不及格人数。
系别 | 平均成绩 | 最高成绩 | 最低成绩 | 优秀率 | 不及格人数
CS | 84.60 | 90 | 80 | 20.00% | 0
MA | 95.00 | 95 | 95 | 100.00% | 0
IS | 93.00 | 93 | 93 | 100.00% | 0
CSE | 99.00 | 99 | 99 | 100.00% | 0

```

图 5.18 学生成绩统计

(6) 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。

```

请输入功能序号： 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：

CS:
200215122 | 刘晨 | 2 | 90
200215121 | 李勇 | 3 | 88
200215121 | 李勇 | 2 | 85
200215121 | 李勇 | 1 | 80
200215122 | 刘晨 | 3 | 80

CSE:
202211915 | 李木子 | 1 | 99

```

图 5.19 学生成绩排名

(7) 输入学号，显示该学生的基本信息和选课信息。

```
请输入功能序号：6

请输入要查询的学生学号：200215121
学生基本信息：
学号：200215121，姓名：李勇，性别：男，年龄：24，系别：CS，奖学金：否

学生选课信息：
课程号 | 课程名称 | 成绩
1 | 数据库 | 80
2 | 数学 | 85
3 | 信息系统 | 88
```

图 5.20 查询学生信息界面

(8) 退出程序并关闭数据库连接。

```
请输入功能序号：0
退出程序。
数据库连接已关闭。

进程已结束，退出代码为 0
```

图 5.21 退出程序界面

5.3 任务总结

实验四主要是学习如何设计编程应用程序来控制数据库。刚看到这个实验任务的时候我的第一想法就是想使用 python 来进行编写，而不是选择更加得心应手的 c 语言，因为近期在课题组的学习也涉及到了大量的 python 实践，也在尝试进行深入学习，同时配置了相关的虚拟环境便想尝试完成任务。

一开始学习了连接环境的配置和 mysql.connector 的几个接口函数，然后先将单个任务函数编写出来，分别进行验证了之后再在主函数里进行连接，这样避免了运行过程中无法确定出错来源的问题，并且对函数进行分装，减小主函数的代码量，同时便于增强代码的可读性。虽然中间遇到了一些编程过程中 sql 语句选取和函数调用的问题，印象比较深的是删除没有选课的课程信息，先需要处理外键的依赖才能进行删除，并且加上自身对 python 的语法了解还是稍浅薄，但最终通过在网上搜集资料和询问同学得到了较好的解决，还是算比较顺利的。

考虑到安全性的问题，对一些细节进行了修改。一开始是直接代码段填写了数据库的相关信息，但是在实际应用场合将密码直接暴露存在安全风险，后面考虑通过输入信息形式来进行数据库连接。计划是编写代码实现在输入密码时显示为某个图标而不是明文文本，有利于信息的保护，但是由于编程技术受限，只能以明码输入，希望通过后续的深入学习可以对方案进行改进。

6 课程总结

6.1 主要工作

- (1) 首先，我系统地学习和巩固了 SQL 语言对数据库的基本操作。我学会了如何创建数据库、创建表格以及对数据进行增加、删除和修改等操作。
- (2) 其次，我深入了解了 SQL 查询语句中的复杂语句。我学习了等值连接（JOIN）、自身连接、子查询、带有 ALL/ANY 等谓词的子查询等概念。
- (3) 然后，我投入时间学习和掌握了 MySQL 数据库和可视化客户端管理工具 Navicat 的使用。通过学习 Navicat 提供的直观界面和功能，我学会了连接到数据库服务器，这使得数据库的管理和操作变得更加便捷和高效。并且学习了对视图、用户、触发器、存储过程、函数以及完整性约束的 sql 语句使用和编写。
- (4) 最后，我学习了使用 python 编程应用程序来控制数据库的信息操作。

6.2 心得体会

通过完成这些数据库管理和 SQL 编程任务的实验，加深了我对课本知识的理解，并掌握熟练运用它们的语法和用法，同时也锻炼了解决实际问题的能力。在网络空间安全领域，数据库管理能力至关重要。

从网络空间安全的角度来看，这些实验体现了数据库在维护和保障信息安全中的作用。例如，通过实现合理的数据访问控制和使用触发器来审计和记录敏感操作，可以预防非法访问和数据篡改，保护数据的完整性和私密性。此外，编写精确的 SQL 查询也帮助我理解了如何高效检索、监控数据，这在检测和防御潜在的网络攻击中扮演着重要角色。

从思政角度来讲，这段实验过程也提醒了我科技发展对于道德责任的要求。比如在处理学生信息时，我意识到作为数据库管理员应当遵行严格的信息安全和隐私保护标准，这涉及到了法律和伦理问题。保护数据安全是网络空间安全人员的重要职责之一，应当防止个人数据泄露或被滥用。

最后，在深入实验的过程中，我也感受到了问题解决的乐趣和挑战。每当我修正一个语法错误、优化一个查询或成功实现一个复杂的功能时，它不仅增强了我的技术能力，也提升了我的自信心和解决问题的决心。我认识到，持续学习、实践和不断探索是在快速发展的网络空间安全领域取得成功的关键。

6.3 未来展望

个人感觉学习时间还是比较紧凑和仓促的，而且课程安排时间较晚，接近于期末月，对报告的完成和完善时间较短，想进行一些拓展的学习和内容的实现，比如人机交互图形界面和其他功能的编写，但是迫于时间的压力无法在提交报告之前实现。但是自己会在课后进行完善和学习，也非常感谢老师们的教导！