华中科技大学

数据库系统概论实验检查

姓　　名：　　袁　也

学　　院：　　网络空间安全学院

专　　业：　　网络空间安全专业

班　　级：　　网安 1902 班

学　　号：　　U201911808

指导教师：　　路松峰

分数	
教师	

2021 年 12 月 24 日

目　　录

2 数据库定义与基本操作

2.2 完成过程

2.2.1 安装数据库

1） 安装并启动数据库，如图 2.1 所示：

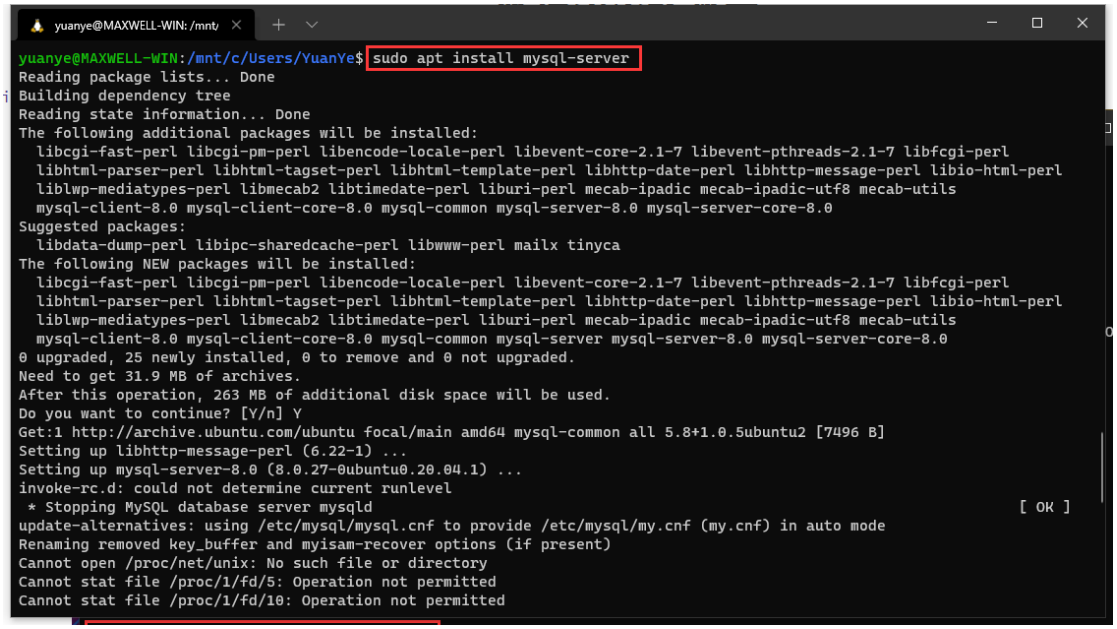

图 2.1 安装并启动数据库

2） 创建名为 CSEDB_U201911808 的数据库，如图 2.2 所示：

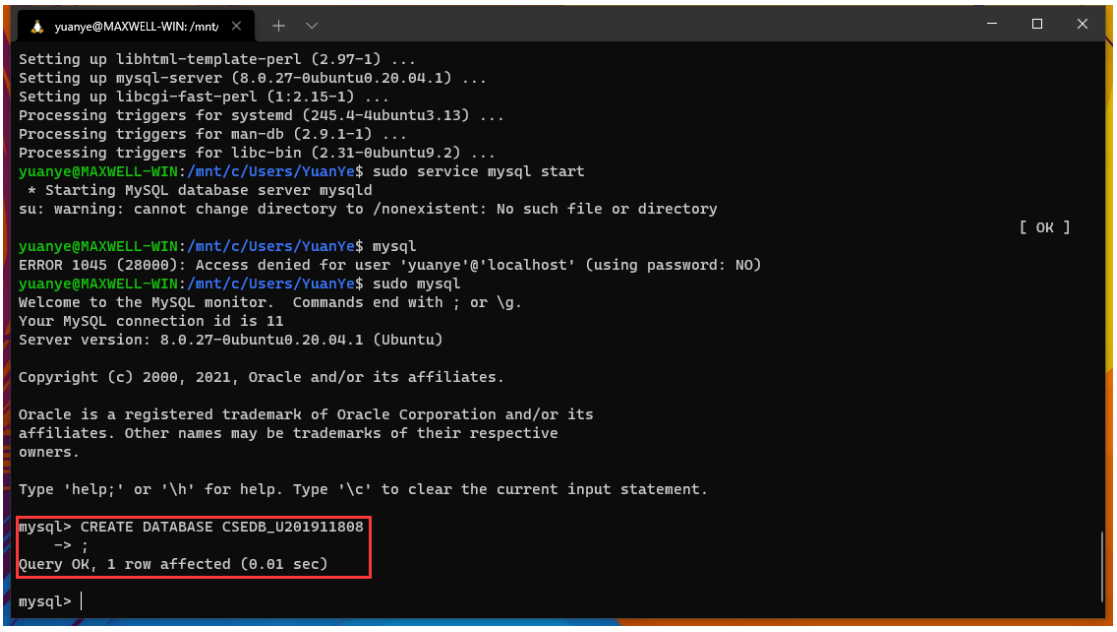

图 2.2 创建数据库

2.2.2 基本表操作

3） 在数据库中按照要求创建三个表，如图 2.3 所示：

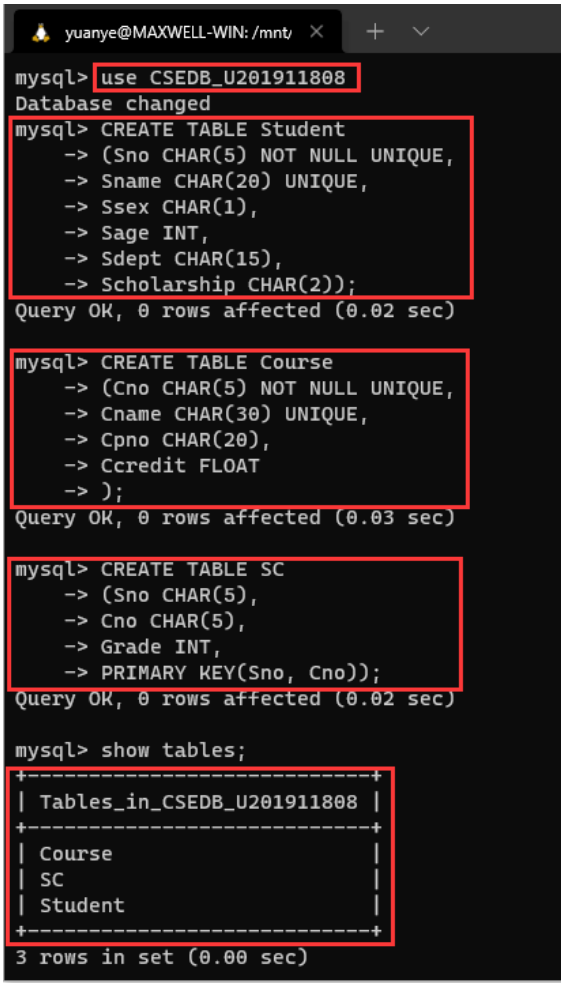

图 2.3 创建三个表

4） 使用 DROP 语句删除表，如图 2.4 所示：

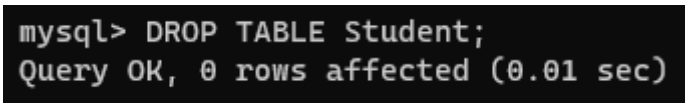

图 2.4 删除表

5） 练习创建和删除索引操作，如图 2.5 所示：

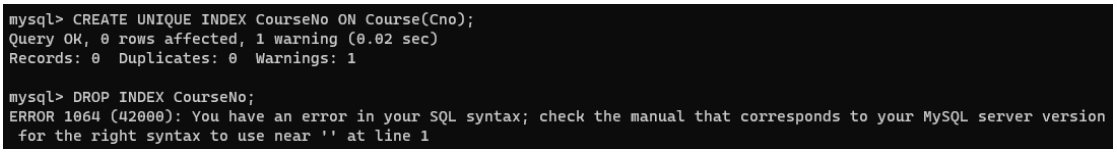

图 2.5 创建和删除索引

6） 使用 ALTER 指令，在表格中增加一列，如图 2.6 所示：

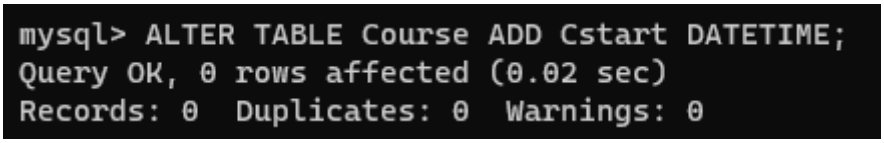

图 2.6 表格增加一列

2.2.3 删除数据库

7） 使用 DROP 语句删除数据库，如图 2.7 所示:

```
mysql> DROP DATABASE CSEDB_U201911808
    -> ;
Query OK, 1 row affected (0.01 sec)

mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
4 rows in set (0.00 sec)

mysql>
```

图 2.7 删除数据库

完成上述联系后，重新创建新的数据库，为接下来的实验做准备。

2.2.4 创建示例数据库

1） 创建新的数据库 S_T_U201911808，并在其中定义三个基本表，Student, Course, SC. 如图 2.8 所示：

```
mysql> CREATE DATABASE S_T_U201911808;
Query OK, 1 row affected (0.01 sec)

mysql> USE S_T_U201911808;
Database changed
mysql> CREATE TABLE Student
    -> (Sno CHAR(9) PRIMARY KEY,
    -> Sname CHAR(20) UNIQUE,
    -> Ssex CHAR(2),
    -> Sage SMALLINT,
    -> Sdept CHAR(20),
    -> Scholarship CHAR(2));
Query OK, 0 rows affected (0.02 sec)

mysql> CREATE TABLE Course
    -> (Cno CHAR(4) PRIMARY KEY,
    -> Cname CHAR(40),
    -> Cpno CHAR(4),
    -> Ccredit SMALLINT,
    -> FOREIGN KEY (Cpno) REFERENCES Course(Cno));
Query OK, 0 rows affected (0.02 sec)

mysql> 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));
Query OK, 0 rows affected (0.02 sec)
```

图 2.8 创建数据库并定义三个基本表

2.2.5 创建基本表并添加数据

2） 使用可视化数据库管理软件——NaviCat 向刚刚定义的三个基本表中添加数据，如图 2.9 所示：

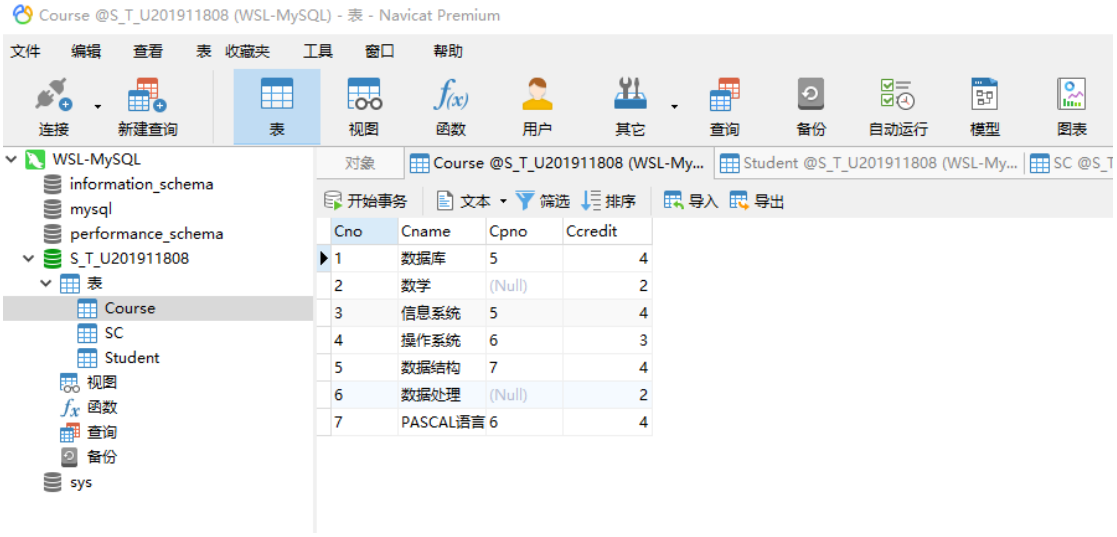

图 2.9 使用可视化软件添加数据

3） 回到命令行界面，使用指令查看并当前表格中现有的数据，如图 2.10 所示：

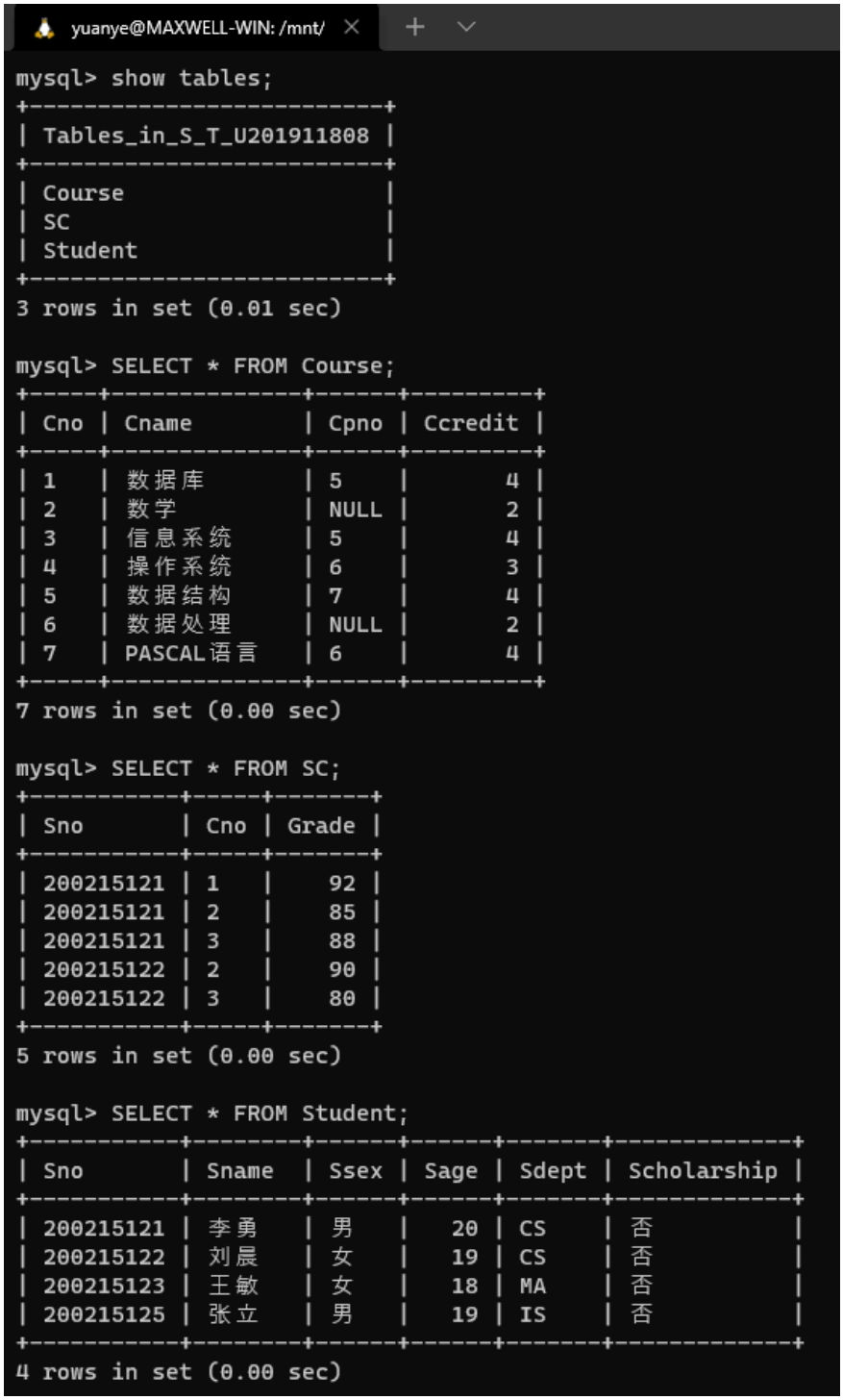

图 2.10 查看当前表格中全部数据

2.2.6 对基本表进行查询

4） 练习使用 SELECT 语句查询，如图 2.11 所示的是查询 student 表格中的全部学生信息，如图 2.11 所示：

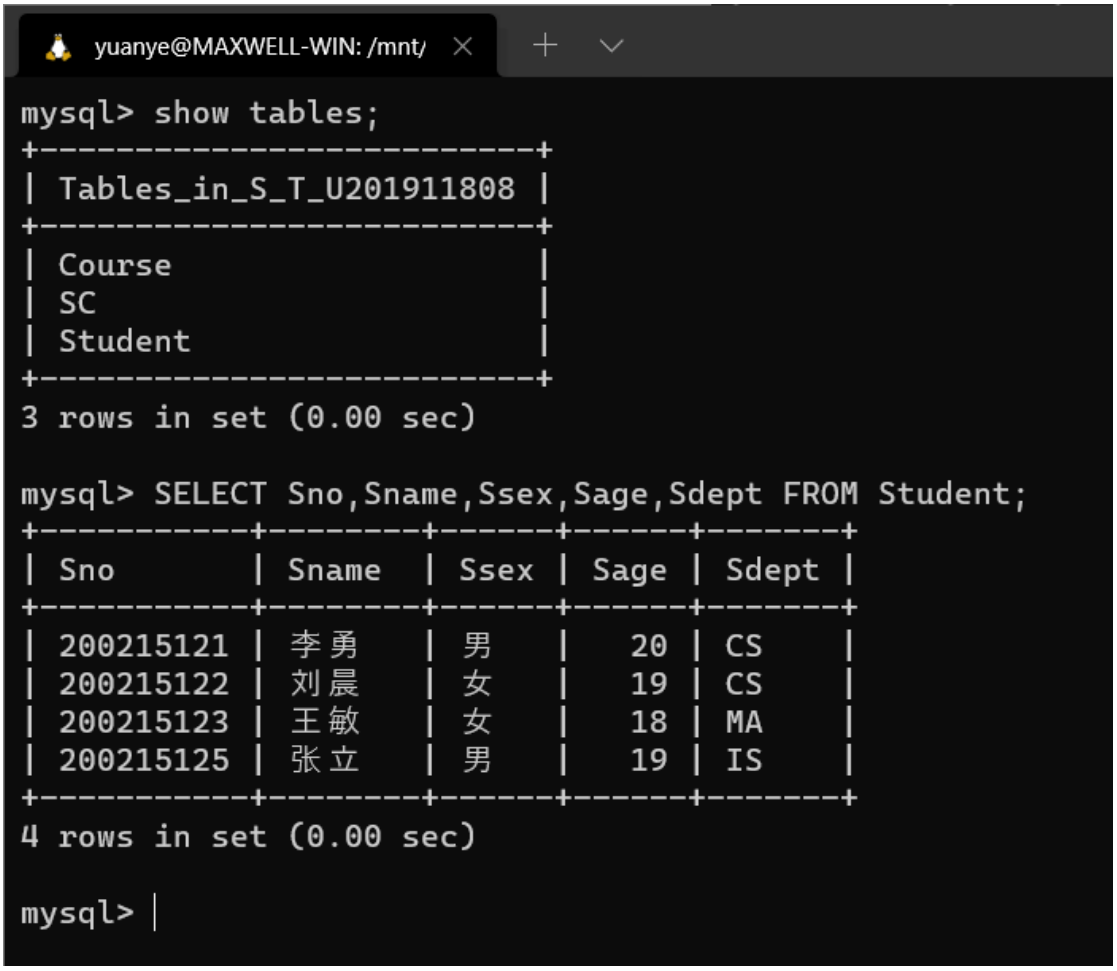

图 2.11 查询 srudent 表信息

5） 接下来，使用可视化界面查询选修 2 号课程并且成绩在 90 分以上的全部同学学号、姓名，查询结果为空，如图 2.12 所示：

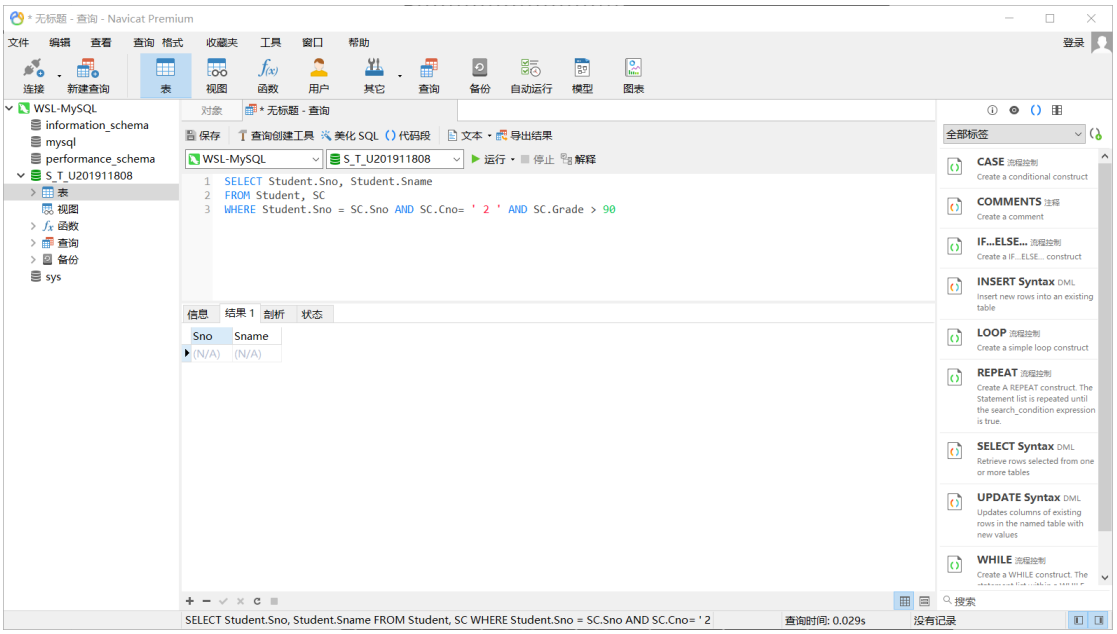

图 2.12 使用可视化管理软件查询信息

6） 接下来，依然利用可视化软件练习谓词查询操作，如图 2.13 所

示：

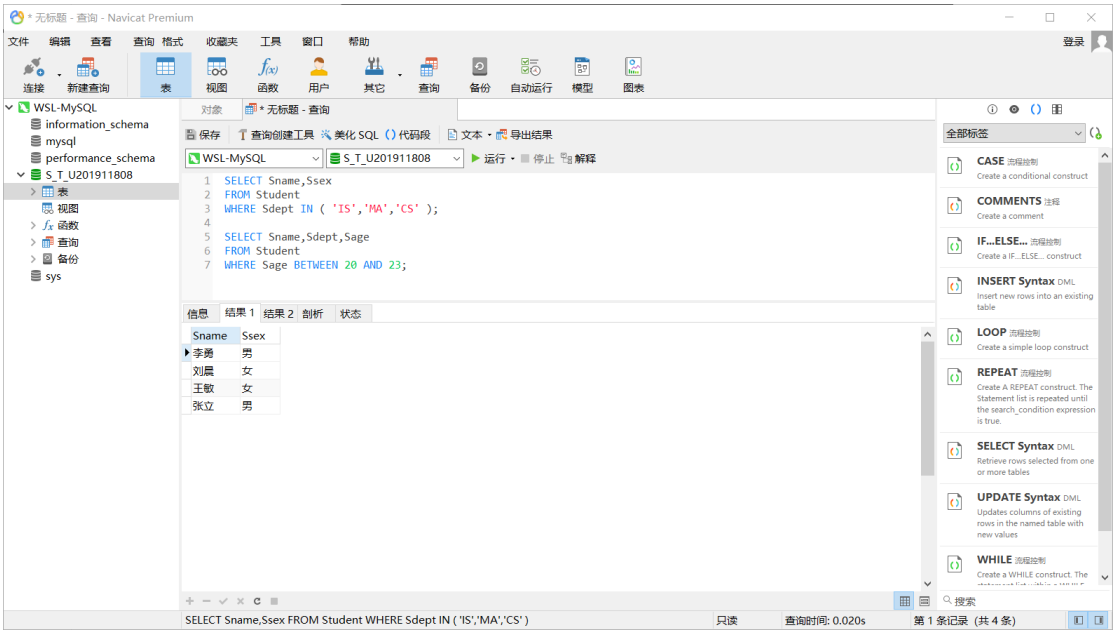

图 2.13 利用可视化软件进行谓词查询

7） 练习模糊查询，利用 LIKE 子句实现模糊查询。例如：查询所有姓刘学生的姓名、学号和性别，其结果如图 2.14 所示：

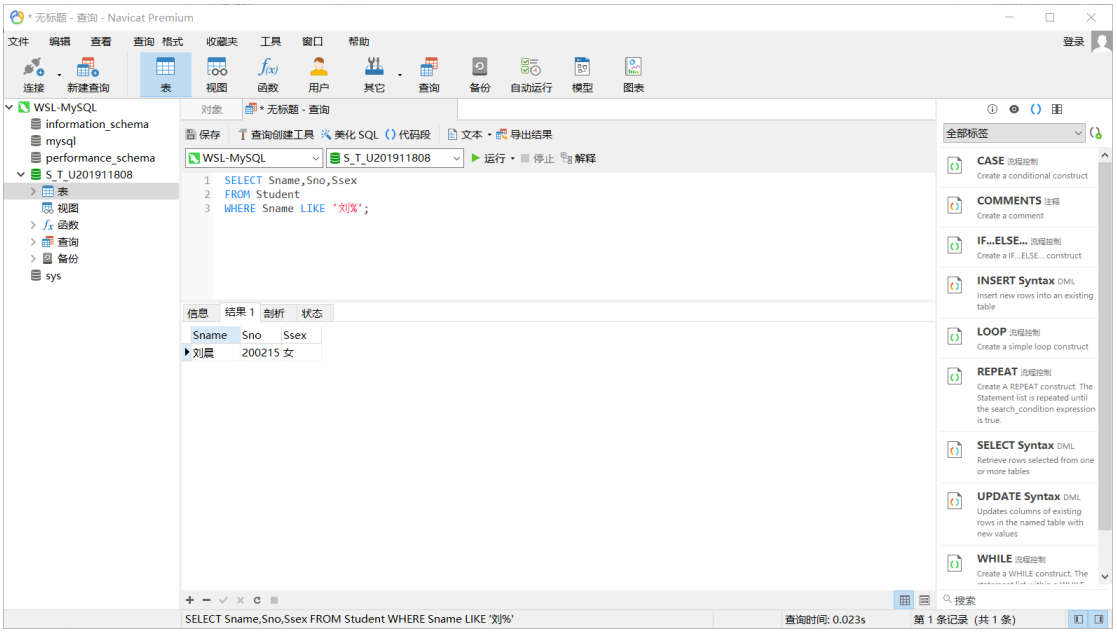

图 2.14 模糊查询练习

8） 利用 ORDER 子句为结果排序，例如：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，结果如图 2.15 所示：

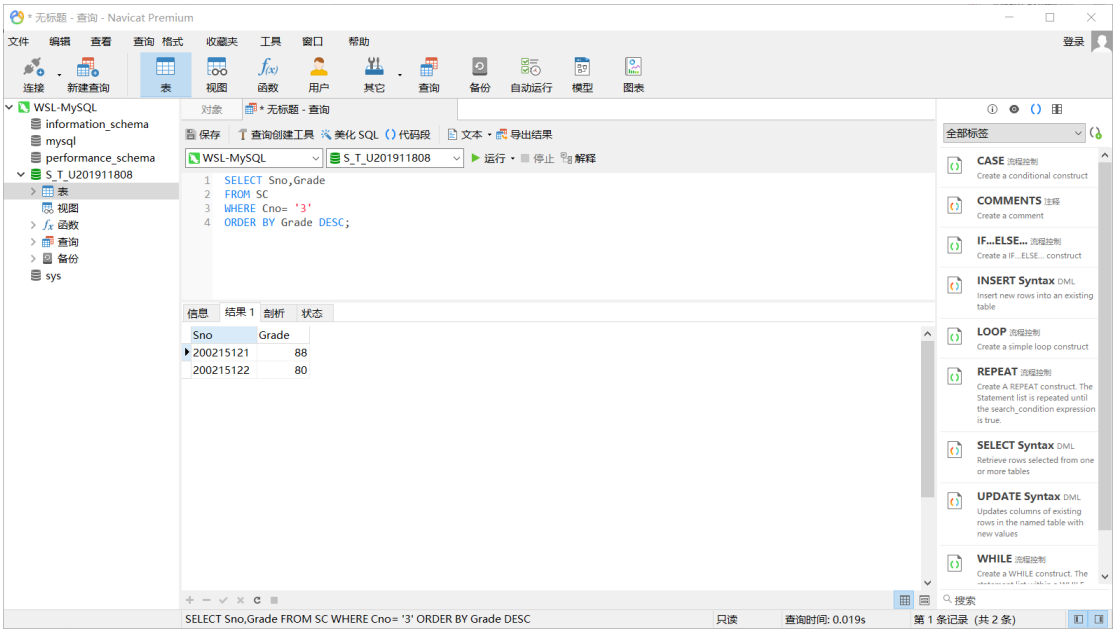

图 2.15 排序查询练习

9） 用 SQL Server 的统计函数进行统计计算，例如：计算 1 号课程的学生平均成绩，其结果如图 2.16 所示：

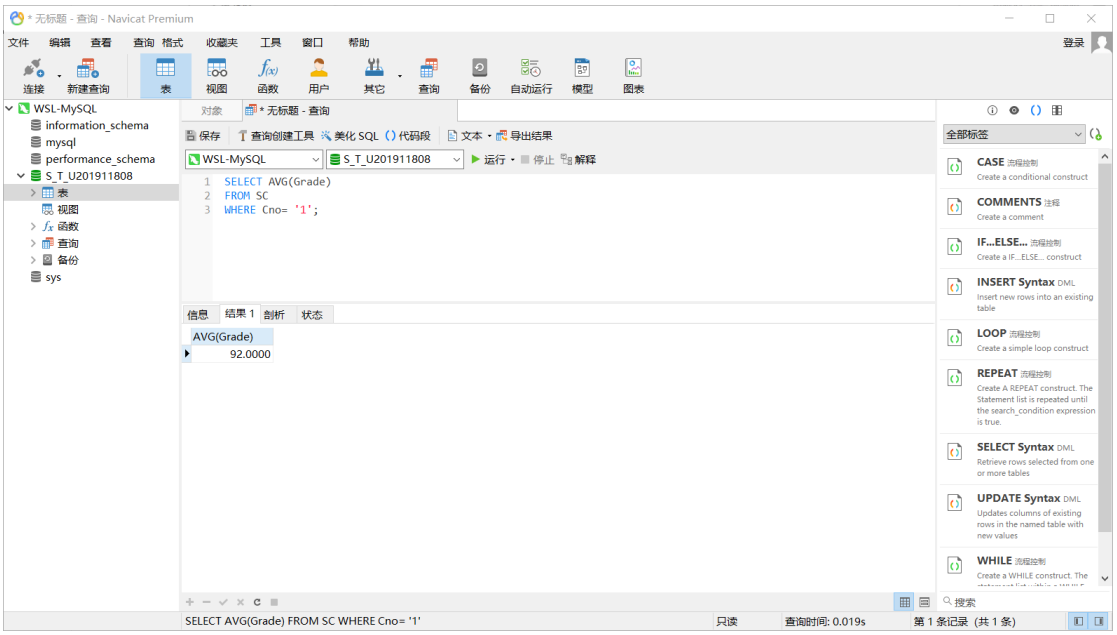

图 2.16 统计计算函数

10） 使用 Group By 子句查询选修了 3 门课以上课程的学生学号，查询结果为空，如图 2.17 所示：

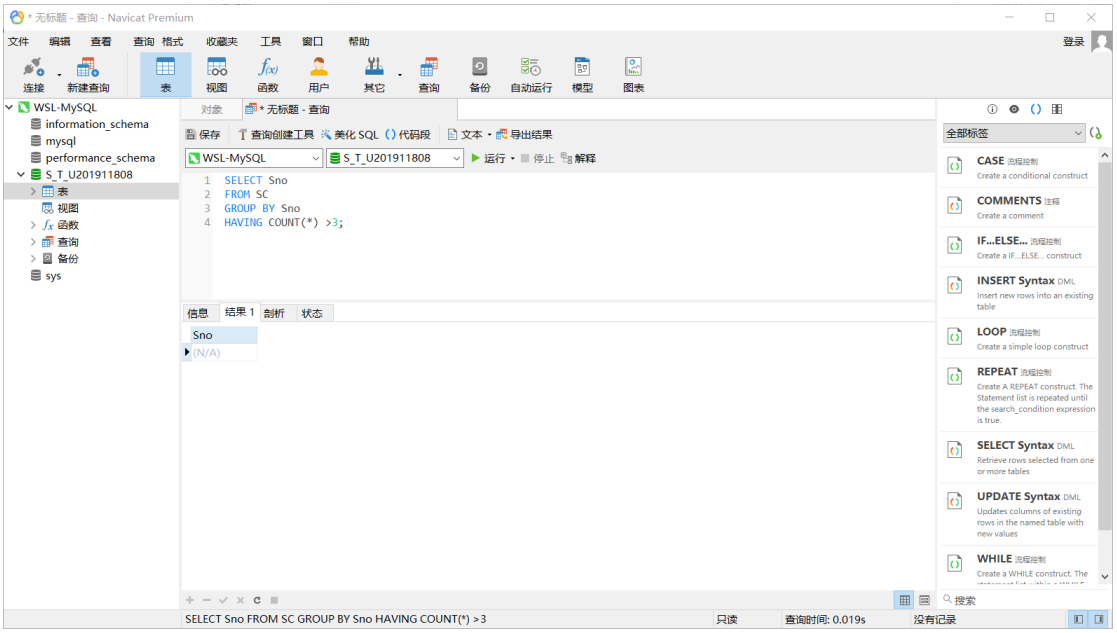

图 2.17 分组查询练习

2.2.7 扩展练习

（1）查询全体学生的学号、姓名和年龄，如图 2.18 所示：

```
1  SELECT Sno, Sname, Sage
2  FROM Student;
3
```

信息 | 结果 1 | 剖析 | 状态

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

图 2.18 查询信息

（2）查询所有计算机系学生的详细记录，如图 2.19 所示：

```
1  SELECT *
2  FROM Student
3  WHERE Sdept = 'CS';
```

信息 | 结果 1 | 剖析 | 状态

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

图 2.19 查询计算机系信息

（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，如图 2.20 所示：

```
1  SELECT Sno, Cno, Grade
2  FROM SC
3  WHERE Grade >= 90 OR Grade < 60;
4
```

信息 | 结果 1 | 剖析 | 状态

Sno	Cno	Grade
200215121	1	92
200215122	2	90

图 2.20 查询优秀信息

（4）查询年龄不在 19~20 岁之间的学生姓名、性别和年龄，如图 2.21 所示：

```
1  SELECT Sname, Ssex, Sage
2  FROM Student
3  WHERE Sage NOT BETWEEN 19 AND 20;
4
```

信息 | 结果 1 | 剖析 | 状态

Sname	Ssex	Sage
王敏	女	18

图 2.21 年龄筛选查询

（5）查询数学系（MA）、信息系（IS)的学生的姓名和所在系，如图 2.22 所示：

```
1  SELECT Sname, Sdept
2  FROM Student
3  WHERE Sdept IN ( 'MA', 'IS' );
```

信息 | 结果 1 | 剖析 | 状态

Sname	Sdept
王敏	MA
张立	IS

图 2.22 按系别筛选查询

（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，如图 2.23 所示：

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

信息 | 结果 1 | 剖析 | 状态

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

图 2.23 模糊查询实现

（7) 找出所有没有选修课成绩的学生学号和课程号，如图 2.24 所示：

```
SELECT Sno, Cno
FROM SC
WHERE Grade IS NULL;
```

信息 | 结果 1 | 剖析 | 状态

Sno	Cno
(N/A)	(N/A)

图 2.24 筛选空置的信息

（8）查询学生 200215121 选修课的最高分、最低分以及平均成绩，如图 2.25 所示：

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

信息 | 结果 1 | 剖析 | 状态

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

图 2.25 按照学号筛选查询

（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，如图 2.26 所示：

```
1  SELECT Sno, Grade
2  FROM SC
3  WHERE Cno = '2'
4  ORDER BY Grade;
```

信息 | 结果 1 | 剖析 | 状态

Sno	Grade
200215121	85
200215122	90

图 2.26 查询特定课程学生信息并排序

（10）查询每个系名及其学生的平均年龄，如图 2.27 所示：

```
1  SELECT Sdept, AVG(Sage)
2  FROM Student
3  GROUP BY Sdept;
```

信息 | 结果 1 | 剖析 | 状态

Sdept	AVG(Sage)
CS	19.5000
MA	18.0000
IS	19.0000

图 2.27 查询综合函数

（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？），如图 2.28 所示：

```
1  SELECT Sdept, AVG(Sage)
2  FROM Student
3  GROUP BY Sdept
4  HAVING AVG(Sage) <= 19;
5
```

信息 | 结果 1 | 剖析 | 状态

Sdept	AVG(Sage)
MA	18.0000
IS	19.0000

图 2.28 思考题

3　SQL 的复杂操作

3.2 完成过程

3.2.1 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）。

```
1  SELECT Course.Cno, Cname, Sno, Grade
2  FROM SC RIGHT JOIN Course ON (Course.Cno = SC.Cno);
```

信息　结果 1　剖析　状态

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)

图 3.1 查询 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 1：嵌套搜索

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

信息　结果 1　剖析　状态

Sno	Sname	Sage
200215122	刘晨	19
200215125	张立	19

图 3.2 嵌套搜索

方法 2：自连接

```
1  SELECT S1.Sno, S1.Sname, S1.Sage
2  FROM Student S1, Student S2
3  WHERE S2.Sname = '张立' AND S1.Sage = S2.Sage;
```

信息 | 结果 1 | 剖析 | 状态

Sno	Sname	Sage
200215122	刘晨	19
200215125	张立	19

图 3.3 自连接

方法 3：EXIST 子句

```
1  SELECT Sno, Sname, Sage
2  FROM Student S1
3  WHERE EXISTS ( SELECT * FROM Student S2 WHERE S2.Sage = S1.Sage AND S2.Sname = '张立' );
```

信息 | 结果 1 | 剖析 | 状态

Sno	Sname	Sage
200215122	刘晨	19
200215125	张立	19

图 3.4 EXIST 子句

（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。

```
1  SELECT Student.Sno, Sname
2  FROM Student, SC
3  WHERE Student.Sno = SC.Sno AND Cno = 3 AND Grade BETWEEN 80 AND 89;
```

信息 | 结果 1 | 剖析 | 状态

Sno	Sname
200215121	李勇
200215122	刘晨

图 3.5 查询 2

（4）查询学生 200215122 选修的课程号、课程名

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

信息 | 结果 1 | 剖析 | 状态

Cno	Cname
2	数学
3	信息系统

图 3.6 查询 3

（思考：如何查询学生 200215122 选修的课程号、课程名及成绩？）

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

信息 | 结果 1 | 剖析 | 状态

Cno	Cname	Grade
2	数学	90
3	信息系统	80

图 3.7 查询 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 SC1
WHERE Grade - ( SELECT AVG(Grade) FROM SC SC2 GROUP BY Sno HAVING SC1.Sno = SC2.Sno ) < - 5;
```

信息 | 结果 1 | 剖析 | 状态

Sno	Cno
(N/A)	(N/A)

图 3.8 查询 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' );
```

信息 | 结果 1 | 剖析 | 状态

Sno	Sname	Sage
200215123	王敏	18

图 3.9 查询 ALL

（7）查询所有选修了 2 号课程的学生姓名及所在系。

```sql
SELECT Sname, Sdept
FROM Student, SC
WHERE Student.Sno = SC.Sno AND Cno = 2;
```

信息 | 结果 1 | 剖析 | 状态

Sname	Sdept
李勇	CS
刘晨	CS

图 3.10 查询 AND

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

SELECT *
FROM Student;
```

信息 | 结果 1 | 剖析 | 状态

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

图 3.11 查询嵌套 IN

（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
INSERT
INTO Course
VALUES
  (8, 'C语言', NULL, 4),
  (9, '人工智能', 2, 2);

SELECT *
FROM Course;
```

信息 | 结果 1 | 剖析 | 状态

Cno	Cname	Cpno	Ccredit
1	数据库	5	4
2	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	4
8	C语言	(Null)	4
9	人工智能	2	2

图 3.12 使用 INSERT 插入课程

（10）使用 delete 语句把人工智能课程删除，并查询出来。

```
1  DELETE
2  FROM Course
3  WHERE Cname = '人工智能';
```

信息 | 剖析 | 状态

```
DELETE
FROM Course
WHERE Cname = '人工智能'
> Affected rows: 1
> 时间: 0.004s
```

图 3.13 首先执行删除操作

Cno	Cname	Cpno	Ccredit
1	数据库	5	4
2	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	4
8	C语言	(Null)	4

图 3.14 然后再次查询表，发现已经被删除

4　SQL 的高级实验

4.2 完成过程

4.2.1 扩展练习

(1) 创建 CS 系的视图 CS_View

```
CREATE VIEW CS_View AS SELECT
*
FROM
  Student
WHERE
  Sdept = 'CS';

SELECT *
FROM CS_VIEW;
```

图 4.1 创建视图操作

Sno	Sname	Ssex	Sage	Sdept	Scholarship
200215121	李勇	男	22	CS	否
200215122	刘晨	女	21	CS	否

图 4.2 查询视图查看创建结果

(2) 在视图 CS_View 上查询 CS 系选修了 1 号课程的学生

```
SELECT CS_View.Sno, Sname, Ssex, Sage, Sdept, Scholarship
FROM CS_View, SC
WHERE CS_View.Sno = SC.Sno AND Cno = 1;
```

信息　结果 1　剖析　状态

Sno	Sname	Ssex	Sage	Sdept	Scholarship
200215121	李勇	男	22	CS	否

图 4.3 在视图上进行查询

(3) 创建 IS 系成绩大于 80 的学生的视图 IS_View

```
1  CREATE VIEW IS_View ( Sno, Sname, Ssex, Sage, Sdept, Scholarship ) AS SELECT
2  Student.Sno,
3  Sname,
4  Ssex,
5  Sage,
6  Sdept,
7  Scholarship
8  FROM
9    Student,
10   SC
11 WHERE
12   Student.Sno = SC.Sno
13   AND Sdept = 'IS'
14   AND Grade > 80;
```

信息 剖析 状态

```
Ssex,
Sage,
Sdept,
Scholarship
FROM
        Student,
        SC
WHERE
        Student.Sno = SC.Sno
        AND Sdept = 'IS'
        AND Grade > 80
> OK
> 时间: 0.007s
```

图 4.4 新建关于 IS 系得视图

(4) 在视图 IS_View 查询 IS 系成绩大于 80 的学生

```
1  SELECT *
2  FROM IS_View;
3
```

信息 结果 1 剖析 状态

Sno	Sname	Ssex	Sage	Sdept	Scholarship
(N/A)	(N/A)	(N/A)	(N/A)	(N/A)	(N/A)

图 4.5 查询视图，无结果

(5) 删除视图 IS_View

```
1  DROP
2  VIEW IS_View;
```

信息 剖析 状态

```
DROP
VIEW IS_View
> OK
> 时间: 0.007s
```

图 4.6 删除视图

(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。

```
mysql> show grants for 'U1'@'localhost';
+-----------------------------------------------------------------------------------------+
| Grants for U1@localhost                                                                 |
+-----------------------------------------------------------------------------------------+
| GRANT USAGE ON *.* TO `U1`@`localhost`                                                  |
| GRANT SELECT, UPDATE, ALTER ON `S_T_U201911808`.`Student` TO `U1`@`localhost`           |
+-----------------------------------------------------------------------------------------+
2 rows in set (0.01 sec)

mysql> show grants for 'U2'@'localhost';
+-----------------------------------------------------------+
| Grants for U2@localhost                                   |
+-----------------------------------------------------------+
| GRANT USAGE ON *.* TO `U2`@`localhost`                    |
| GRANT INSERT ON `S_T_U201911808`.`SC` TO `U2`@`localhost` |
+-----------------------------------------------------------+
2 rows in set (0.00 sec)
```

图 4.7 在命令行中检查权限授予情况

然后用 U1 登录，分别

1）查询学生表的信息；

```
1  SELECT *
2  FROM Student;
```

信息 | 结果 1 | 剖析 | 状态

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

图 4.8 使用 U1 查询 Student 表

2）把所有学生的年龄增加 1 岁，然后查询；

```
1  UPDATE Student
2  SET Sage = Sage + 1;
3
4  SELECT *
5  FROM Student;
```

信息 | 结果 1 | 剖析 | 状态

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

图 4.9 修改年龄并查询

3）删除 IS 系的学生；

```
1  DELETE
2  FROM Student
3  WHERE Sdept = 'IS';
4
```

信息 状态

```
DELETE
FROM Student
WHERE Sdept = 'IS'
> 1142 - DELETE command denied to user 'U1'@'localhost' for table 'Student'
> 时间: 0s
```

图 4.10 无权限删除

4）查询 CS 系的选课信息。

```
1  SELECT Student.Sno, Sname, Cno, Cname
2  FROM Student, SC
3  WHERE Student.Sno = SC.Sno AND Student.Sdept = 'CS';
4
```

信息 状态

```
SELECT Student.Sno, Sname, Cno, Cname
FROM Student, SC
WHERE Student.Sno = SC.Sno AND Student.Sdept = 'CS'
> 1142 - SELECT command denied to user 'U1'@'localhost' for table 'SC'
> 时间: 0s
```

图 4.11 无权查询 CS 系选课信息

用 U2 登录，分别

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

```
1  INSERT INTO SC
2  VALUES ('200215122', '1', 75);
3
```

信息 剖析 状态

```
INSERT INTO SC
VALUES ('200215122', '1', 75)
> Affected rows: 1
> 时间: 0.002s
```

图 4.12 U2 可以向 SC 中插入数据

2）查询 SC 表的信息，

```
1  SELECT *
2  FROM SC;
```

信息 状态

```
SELECT *
FROM SC
> 1142 - SELECT command denied to user 'U2'@'localhost' for table 'SC'
> 时间: 0s
```

图 4.13 U2 无权查询 SC 表得信息

3）查询视图 CS_View 的信息。

```
1  SELECT *
2  FROM CS_View;
```

信息 状态

```
SELECT *
FROM CS_View
> 1142 - SELECT command denied to user 'U2'@'localhost' for table 'CS_View'
> 时间: 0s
```

图 4.14 U2 无权查询 CS 视图

(7) 用系统管理员登录，收回 U1 的所有权限

```
1  REVOKE ALL
2  ON *.*
3  FROM 'U1'@'localhost';
```

信息 剖析 状态

```
REVOKE ALL
ON *.*
FROM 'U1'@'localhost'
> OK
> 时间: 0.005s
```

图 4.15 撤销权限

(8) 用 U1 登录，查询学生表的信息

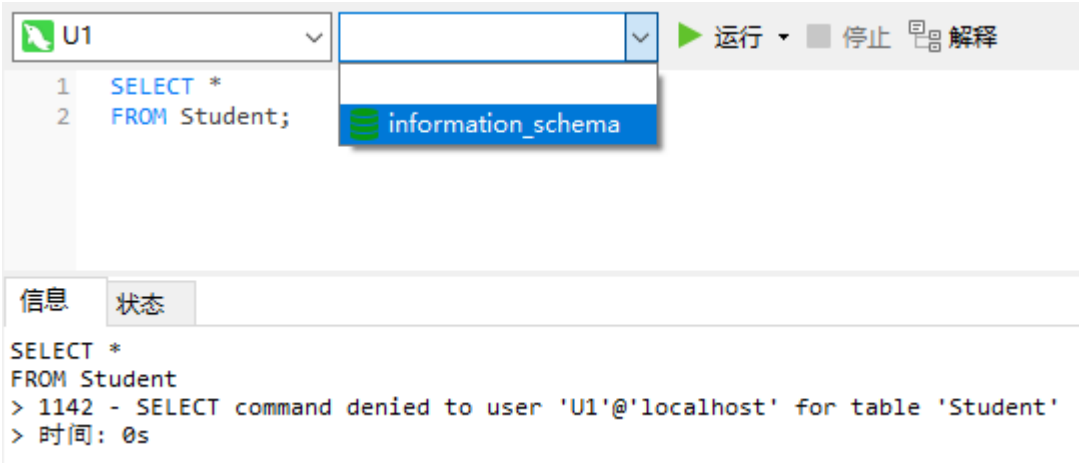

图 4.16 U1 已经无权限

(9) 用系统管理员登录

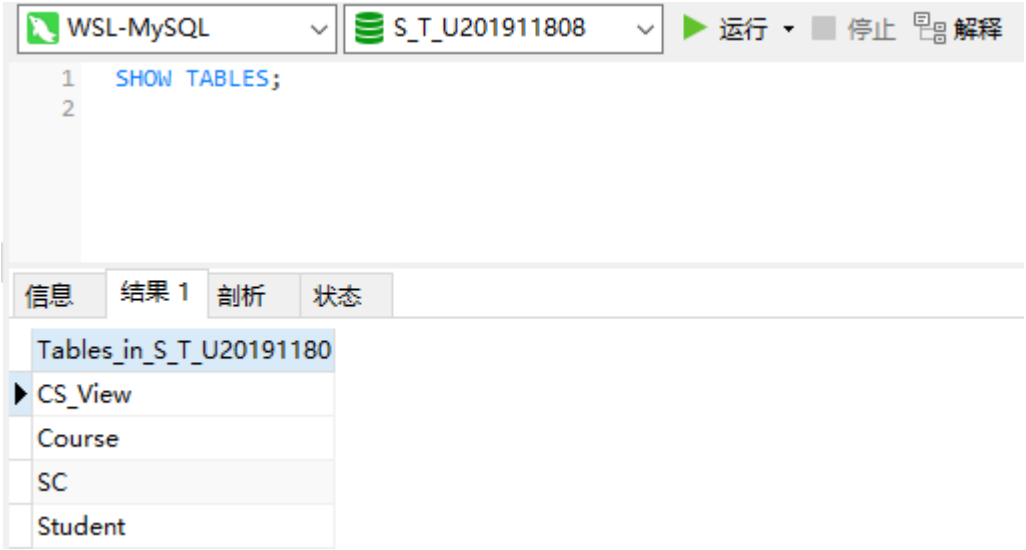

图 4.17 重新用系统管理员登录

(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”。

```
CREATE TRIGGER SCHO_JDG AFTER UPDATE ON SC FOR EACH ROW
BEGIN
  IF
    NEW.Grade >= 95 THEN
      UPDATE Student
      SET Scholarship = '是'
    WHERE
      Scholarship = '否' AND NEW.Sno = Student.Sno;

    ELSEIF NEW.Grade < 95 AND NOT EXISTS ( SELECT * FROM SC WHERE NEW.Sno = SC.Sno AND SC.Grade >= 95
    )
    AND OLD.Grade >= 95 THEN
      UPDATE Student
      SET Scholarship = '否';

    END IF;
END;
```

信息 剖析 状态

```
ELSEIF NEW.Grade < 95 AND NOT EXISTS ( SELECT * FROM SC WHERE NEW.Sno = SC.Sno AND SC.Grade >= 95
)
AND OLD.Grade >= 95 THEN
        UPDATE Student
        SET Scholarship = '否';

END IF;
END
> Affected rows: 0
> 时间: 0.004s
```

图 4.18 定义触发器

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。

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

SELECT Sno, Sname, Scholarship
FROM Student
WHERE Sno = '200215121';
```

信息 结果 1 剖析 状态

Sno	Sname	Scholarship
200215121	李勇	是

图 4.19 查看触发器效果

2）再把刚才的成绩修改为 80，再查询其奖学金。

```
UPDATE SC
SET Grade = 80
WHERE Sno = '200215121' AND Cno = '1';

SELECT Sno, Sname, Scholarship
FROM Student
WHERE Sno = '200215121';
```

信息 结果 1 剖析 状态

Sno	Sname	Scholarship
200215121	李勇	否

图 4.20 查看触发器效果

（11）删除刚定义的触发器

```
1  DROP TRIGGER SCHO_JDG;
```

信息 | 剖析 | 状态

```
DROP TRIGGER SCHO_JDG
> OK
> 时间: 0.004s
```

图 4.21 删除触发器

（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。

```
1  CREATE PROCEDURE CALC_GRADE () BEGIN
2    SELECT
3      MAX( SC.Grade ),
4      AVG( SC.Grade )
5    FROM
6      SC,
7      Student
8    WHERE
9      Student.Sdept = 'CS'
10     AND Student.Sno = SC.Sno;
11
12 END;
13 CALL CALC_GRADE ();
```

信息 | 结果 1 | 剖析 | 状态

MAX(SC.Grade)	AVG(SC.Grade)
90	80.8333

图 4.22 定义存储过程

（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 PROCEDURE SNO_GRADE (
  IN SID CHAR ( 9 )) BEGIN
  SELECT
    Student.Sno,
    Student.Sname,
    Course.Cname,
    SC.Grade
  FROM
    Student,
    SC,
    Course
  WHERE
    Student.Sno = SID
    AND Student.Sno = SC.Sno
    AND SC.Cno = Course.Cno;

END;
CALL SNO_GRADE ( '200215121' );
```

信息 | 结果 1 | 剖析 | 状态

Sno	Sname	Cname	Grade
200215121	李勇	数据库	80
200215121	李勇	数学	80
200215121	李勇	信息系统	80

图 4.23 利用存储过程进行查询

（14）把上一题改成函数。再进行验证。

本题在 MySQL 中无法实现

（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。

在定义完整性约束前，先修改学号为 200215122 同学 1 号课程的成绩为 120 分，操作过程及修改后的结果如下：

```
1  UPDATE SC
2  SET Grade = 120
3  WHERE
4    Sno = '200215122'
5    AND Cno = '1';
6  SELECT
7    *
8  FROM
9    SC;
```

信息 | 结果 1 | 剖析 | 状态

Sno	Cno	Grade
200215121	1	80
200215121	2	80
200215121	3	80
200215122	1	120
200215122	2	90
200215122	3	80

图 4.24 添加完整性约束

新增约束条件，如下图所示：

```
1  ALTER TABLE SC
2  ADD CONSTRAINT CON_GRADE
3  CHECK (Grade BETWEEN 0 AND 100);
```

信息 | 剖析 | 状态

```
ALTER TABLE SC
ADD CONSTRAINT CON_GRADE
CHECK (Grade BETWEEN 0 AND 100)
> OK
> 时间: 0.085s
```

图 4.25 向 SC 表中新增完整性约束条件

之后，再次尝试修改学生成绩为 120 分，报错，如下图所示：

```
1  UPDATE SC
2  SET Grade = 120
3  WHERE
4    Sno = '200215122'
5    AND Cno = '1';
6  SELECT
7    *
8  FROM
9    SC;
```

信息 | 状态

```
UPDATE SC
SET Grade = 120
WHERE
        Sno = '200215122'
        AND Cno = '1'
> 3819 - Check constraint 'CON_GRADE' is violated.
> 时间: 0s
```

图 4.26 增加完整性约束条件后，不合法数据修改将会报错

5.2.8 系统测试

测试环境：操作系统：Windows 10；数据库版本：8.0.27-0ubuntu0.20.04.1

1. 应用程序启动与连接数据库测试

首先按回车采取默认值，当然如果需要连接其他数据库，也可以输入相应的值，如图 5.6 所示。

```
D:\Anaconda\python.exe C:/Users/YuanYe/Desktop/数据库/expr4.py
Input the IP Address (default: localhost):
Username (default: yuanye):
Password (default: yuanye):
Select one database to operate (default: S_T_U201911808):
```

图 5.6

然后会显示登录信息、数据库版本信息以及初始主菜单界面，如图 5.7 所示。

```
 ... Connecting to the Database ...
---------------- Student Management System ----------------
        Author: 网安1902班   袁也   U201911808
        MySQL Version: 8.0.27-0ubuntu0.20.04.1
 User: yuanye@localhost   Database: S_T_U201911808

   ************************************ Menu ************************************
   1. Insert New Student Info                2. Modify Existing Student Info
   3. Insert New Course Info                 4. Modify Existing Course Info
   5. Delete Existing Course Info            6. Record Student Grade
   7. Modify Student Grade                   8. Print Analytical Info by Dept
   9. Print Grade Rank by Dept               10. Print Student Info

                                    0. Exit
   ******************************************************************************

Choose what you want (0~10):
```

2. 测试插入学生信息

选择操作 1，并输入相应的插入数据，如图 5.8 所示。

```
Choose what you want (0~10): 1
================== Mode 1: Insert New Student Info ==================
Please input the following information!
Student ID: 200215129
Name: 张三
Gender: 男
Age: 21
Student's Department: EE
Have Scholarship? (y/n)n
Re-check the student's information below:
+----+-----------+---------+-------+-------+--------+--------------+
|    |       sno | sname   | sex   |   age | dept   | scholarship  |
|----+-----------+---------+-------+-------+--------+--------------|
|  0 | 200215129 | 张三    | 男    |    21 | EE     | 否           |
+----+-----------+---------+-------+-------+--------+--------------+
ARE YOU SURE TO INSERT THE NEW INFORMATION? (y/n)y
Successfully Inserted!
```

在 navicat 中查询 student 表的信息，如图 5.9 所示，可以看到已经能够查询到相应的学生信息。

Sno	Sname	Ssex	Sage	Sdept	Scholarship
200215121	李勇	男	23	CS	否
200215122	刘晨	男	20	CS	是
200215123	王敏	女	19	MA	否
200215125	张立	男	20	IS	否
200215129	张三	男	21	EE	否

3. 测试修改学生信息

选择操作 2，进行修改操作，如图 5.10 所示。

```
Choose what you want (0~10): 2
================== Mode 2: Modify Existing Student Info ==================
Please input the Student ID to search the student!
Student ID: 200215129
Student founded! Here is the information of this person:
+----+-----------+---------+-------+-------+--------+--------------+
|    |       sno | sname   | sex   |   age | dept   | scholarship  |
|----+-----------+---------+-------+-------+--------+--------------|
|  0 | 200215129 | 张三    | 男    |    21 | EE     | 否           |
+----+-----------+---------+-------+-------+--------+--------------+
Select column 1~6 to modify, select 0 to exit: 4
Input the new data: 24
Successfully Updated!
Select column 1~6 to modify, select 0 to exit: 6
Input the new data: 是
Successfully Updated!
Select column 1~6 to modify, select 0 to exit: 0
```

Student 表如图 5.11 所示：

Sno	Sname	Ssex	Sage	Sdept	Scholarship
200215121	李勇	男	23	CS	否
200215122	刘晨	男	20	CS	是
200215123	王敏	女	19	MA	否
200215125	张立	男	20	IS	否
200215129	张三	男	24	EE	是

4. 插入课程信息

选择模式 3，输入相应的数据，如图 5.12 所示：

```
Choose what you want (0~10): 3
================== Mode 3: Insert New Course Info ==================
Please input the following information!
Course ID: 10
Course Name: 大学物理
Previous Course ID: 2
Credits: 5
Re-check the course's information below:
+----+-------+----------+--------+-----------+
|    |   cno | cname    |   cpno |   ccredit |
|----+-------+----------+--------+-----------|
|  0 |    10 | 大学物理 |      2 |         5 |
+----+-------+----------+--------+-----------+
ARE YOU SURE TO INSERT THE NEW INFORMATION? (y/n)y
Successfully Inserted!
```

查询 course 表，如图 5.13 所示。

Cno	Cname	Cpno	Ccredit
0	入学考试	(Null)	5
1	数据库	5	4
10	大学物理	2	5
2	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	4
8	C语言	(Null)	4
9	Py语言	0	2

5. 修改课程信息

选择模式 4，输入修改的信息，如图 5.14 所示。

```
Choose what you want (0~10): 4
================== Mode 4: Modify Existing Course Info ==================
Please input the Course ID to search the course!
Course ID: 10
Course founded! Here is the information of this course:
+----+-------+----------+--------+-----------+
|    |   cno | cname    |   cpno |   ccredit |
|----+-------+----------+--------+-----------|
|  0 |    10 | 大学物理 |      2 |         5 |
+----+-------+----------+--------+-----------+
Select column 1~4 to modify, select 0 to exit: 4
Input the new data: 3
Successfully Updated!
Select column 1~4 to modify, select 0 to exit: 0
```

修改后的 course 表如图 5.15 所示：

Cno	Cname	Cpno	Ccredit
0	入学考试	(Null)	5
1	数据库	5	4
10	大学物理	2	3
2	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	4
8	C语言	(Null)	4
9	Py语言	0	2

6. 删除课程信息

选择模式 5，删除大学物理课程信息，如图 5.16 所示。

```
Choose what you want (0~10): 5
================== Mode 5: Delete Existing Course Info ==================
Please input the Course ID to search the course!
Course ID: 10
Course founded! Here is the information of this course:
+----+-------+----------+--------+-----------+
|    |   cno | cname    |   cpno |   ccredit |
|----+-------+----------+--------+-----------|
|  0 |    10 | 大学物理 |      2 |         3 |
+----+-------+----------+--------+-----------+
ARE YOU SURE TO DELETE THIS COURSE? (y/n): y
Successfully Deleted!
```

删除后的 course 表如图 5.17 所示。

Cno	Cname	Cpno	Ccredit
0	入学考试	(Null)	5
1	数据库	5	4
2	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	4
8	C语言	(Null)	4
9	Py语言	0	2

7. 登记学生成绩

选择模式 6，输入相应的数据，如图 5.18 所示。

```
Choose what you want (0~10): 6
================== Mode 6: Record Student's Grade ==================
Please input the following information!
Student ID: 200215129
Course ID: 9
Student's Grade: 76
Re-check the information below:
+----+-----------+-------+---------+
|    |       sno |   cno |   grade |
|----+-----------+-------+---------|
|  0 | 200215129 |     9 |      76 |
+----+-----------+-------+---------+
ARE YOU SURE TO INSERT THE NEW INFORMATION? (y/n)y
Successfully Inserted!
```

添加后的 SC 表如图 5.19 所示。

Sno	Cno	Grade
200215121	1	80
200215121	2	80
200215121	3	80
200215122	1	80
200215122	2	90
200215122	3	80
200215122	5	96
200215123	9	98
200215129	9	76

8. 修改学生成绩

选择模式 7，输入相应的信息查询并修改学生成绩，如图 5.20 所示。

```
Choose what you want (0~10): 7
================== Mode 7: Modify Student's Grade ==================
Please input the Student ID and Course ID to search the grade record!
Student ID: 200215129
Course ID: 9
Course founded! Here is the information of this course:
+----+-----------+-------+---------+
|    |       sno |   cno |   grade |
|----+-----------+-------+---------|
|  0 | 200215129 |     9 |      76 |
+----+-----------+-------+---------+
Are you sure to modify this score? (y/n): y
Input the new grade: 87
Successfully Updated!
```

修改后的 SC 表如图 5.21 所示。

Sno	Cno	Grade
200215121	1	80
200215121	2	80
200215121	3	80
200215122	1	80
200215122	2	90
200215122	3	80
200215122	5	96
200215123	9	98
200215129	9	87

9. 按系打印统计信息

选择模式 8，可以看到统计信息如图 5.22 所示。

```
Choose what you want (0~10): 8
================== Mode 8: Print Analytical Info by Dept ==================
Department List: CS,EE,IS,MA
Here is the information for your inquire:
+----+--------+---------+-------+-------+----------+---------+
|    | dept   |     avg |   max |   min |   a_rate |   f_num |
|----+--------+---------+-------+-------+----------+---------|
|  0 | CS     | 83.7143 |    96 |    80 | 0.285714 |       0 |
|  1 | EE     | 87      |    87 |    87 | 0        |       0 |
|  2 | IS     |         |   nan |   nan | 0        |       0 |
|  3 | MA     | 98      |    98 |    98 | 1        |       0 |
+----+--------+---------+-------+-------+----------+---------+
Press Enter to continue...
```

10. 学生成绩排名

选择模式 9，输入项查询的院系信息，查询结果如图 5.23 所示。

```
Choose what you want (0~10): 9
================== Mode 9: Print Department Student Ranking ==================
Department List: CS,EE,IS,MA
Please select one department to inquire the student's ranking: CS
Here is the ranking of CS department:
+----+-----------+---------+-------+---------+-----------+
|    |       sno | sname   |   avg |   c_num |   credits |
|----+-----------+---------+-------+---------+-----------|
|  0 | 200215122 | 刘晨    |  86.5 |       4 |        14 |
|  1 | 200215121 | 李勇    |  80   |       3 |        10 |
+----+-----------+---------+-------+---------+-----------+
Do you want to search other department's rank? (y/n)y
Department List: CS,EE,IS,MA
Please select one department to inquire the student's ranking: IS
Here is the ranking of IS department:
+-------+---------+-------+---------+-----------+
| sno   | sname   | avg   | c_num   | credits   |
```

11. 查询学生信息

选择模式 10，输入需要查询的学生学号，查询结果如图 5.24 所示。

```
Choose what you want (0~10): 10
================== Mode 10: Lookup Student Basic Info ==================
Please input the Student ID to search the student!
Student ID: 200215129
Student founded! Here is the information of this person:
+----+-----------+---------+-------+-------+--------+---------------+
|    |       sno | sname   | sex   |   age | dept   | scholarship   |
|----+-----------+---------+-------+-------+--------+---------------|
|  0 | 200215129 | 张三    | 男    |    24 | EE     | 是            |
+----+-----------+---------+-------+-------+--------+---------------+
Student's Course Info:
+----+-------+---------+---------+-----------+--------+
|    |   cno | cname   |   grade |   credits |   cpno |
|----+-------+---------+---------+-----------+--------|
|  0 |     9 | Py语言  |      87 |         2 |      0 |
+----+-------+---------+---------+-----------+--------+
Press Enter to continue ...
```

A 附录

A.1 数据库管理系统源码

```
1. #!/usr/bin/python
2. # -*- coding: UTF-8 -*-
3.
4. import os
5. import pymysql
6. import pandas as pd
7. from tabulate import tabulate
8.
9.
10. def print_menu():
11.     print('''''
12.     ********************************** Menu ********************** ***************
13.     1. Insert New Student Info                  2. Modify Existing St udent Info
14.     3. Insert New Course Info                   4. Modify Existing Co urse Info
15.     5. Delete Existing Course Info              6. Record Student Gra de
16.     7. Modify Student Grade                     8. Print Analytical I nfo by Dept
17.     9. Print Grade Rank by Dept                 10. Print Student Inf o
18.
19.                                         0. Exit
20.     ********************************************************************* ***************
21.     ''')
22.     return
23.
24.
25. def insert_student(db):
26.     cursor = db.cursor()
27.     dic_student = {'sno': [], 'sname': [], 'sex': [], 'age': [], 'dep t': [], 'scholarship': []}
28.     print("================== Mode 1: Insert New Student Info ======= ===========")
29.     print("Please input the following information! ")
30.
31.     dic_student['sno'].append(input("Student ID: "))
32.     dic_student['sname'].append(input("Name: "))
33.     dic_student['sex'].append(input("Gender: "))
34.     dic_student['age'].append(input("Age: "))
35.     dic_student['dept'].append(input("Student's Department: "))
36.     dic_student['scholarship'].append("是
    " if input("Have Scholarship? (y/n)") == 'y' else "否")
37.
38.     df = pd.DataFrame(dic_student)
39.
40.     print("Re-check the student's information below: ")
41.     print(tabulate(df, headers='keys', tablefmt='psql'))
42.
43.     op = input("ARE YOU SURE TO INSERT THE NEW INFORMATION? (y/n)")
44.     if op == 'n':
45.         print("rollback successfully! ")
46.         return
47.     else:
48.         pass
49.
```

```
    sql = "INSERT INTO Student " \
          "VALUES('%s', '%s', '%s', %s, '%s', '%s')" % (
              dic_student['sno'][0], dic_student['sname'][0], dic_student['sex'][0], dic_student['age'][0],
              dic_student['dept'][0],
              dic_student['scholarship'][0])

    try:
        cursor.execute(sql)
        db.commit()
        print("Successfully Inserted! ")
    except:
        print("An unexpected error occurred and we rollback all the changes. ")
        db.rollback()

    return


def update_student(db):
    cursor = db.cursor()
    dic_student = {'sno': [], 'sname': [], 'sex': [], 'age': [], 'dept': [], 'scholarship': []}
    print("================== Mode 2: Modify Existing Student Info ==================")
    print("Please input the Student ID to search the student! ")

    dic_student['sno'].append(input("Student ID: "))

    sql = "SELECT * FROM Student WHERE Sno = '%s'" % dic_student['sno'][0]

    try:
        cursor.execute(sql)
        results = cursor.fetchall()
        for index, row in enumerate(results):
            # dic_student['sno'].append(row[0])
            dic_student['sname'].append(row[1])
            dic_student['sex'].append(row[2])
            dic_student['age'].append(row[3])
            dic_student['dept'].append(row[4])
            dic_student['scholarship'].append(row[5])

        print("Student founded! Here is the information of this person: ")
        df = pd.DataFrame(dic_student)
        print(tabulate(df, headers='keys', tablefmt='psql'))
    except:
        print("Error: unable to fetch data")
        return

    header_list = ['', 'Sno', 'Sname', 'Ssex', 'Sage', 'Sdept', 'Scholarship']

    while True:
        modify_op = input("Select column 1~6 to modify, select 0 to exit: ")
        if modify_op == '0':
            break
        else:
            pass
        new_data = input("Input the new data: ")

        sql = "UPDATE Student SET %s = " % header_list[int(modify_op)]
        if modify_op != 4:
```

```
107.                 sql += "'"%s'"
108.             sql += new_data
109.             if modify_op != 4:
110.                 sql += "'"
111.             sql += "WHERE Sno = '%s'" % dic_student['sno'][0]
112.
113.             try:
114.                 print("Successfully Updated! ")
115.                 cursor.execute(sql)
116.                 db.commit()
117.             except:
118.                 print("update error! ")
119.                 db.rollback()
120.                 return
121.
122.         return
123.
124.
125.     def insert_course(db):
126.         cursor = db.cursor()
127.         dic_course = {'cno': [], 'cname': [], 'cpno': [], 'ccredit': []
    }
128.         print("================== Mode 3: Insert New Course Info ======
    ============")
129.         print("Please input the following information! ")
130.
131.         dic_course['cno'].append(input("Course ID: "))
132.         dic_course['cname'].append(input("Course Name: "))
133.         dic_course['cpno'].append(input("Previous Course ID: "))
134.         dic_course['ccredit'].append(input("Credits: "))
135.
136.         df = pd.DataFrame(dic_course)
137.
138.         print("Re-check the course's information below: ")
139.         print(tabulate(df, headers='keys', tablefmt='psql'))
140.
141.         op = input("ARE YOU SURE TO INSERT THE NEW INFORMATION? (y/n)")

142.         if op == 'n':
143.             print("rollback successfully! ")
144.             return
145.         else:
146.             pass
147.
148.         sql = "INSERT INTO Course " \
149.               "VALUES('%s', '%s', '%s', %s)" % (
150.                   dic_course['cno'][0], dic_course['cname'][0], dic_cou
    rse['cpno'][0], dic_course['ccredit'][0])
151.
152.         try:
153.             cursor.execute(sql)
154.             db.commit()
155.             print("Successfully Inserted! ")
156.         except:
157.             print("An unexpected error occurred and we rollback all the
     changes. ")
158.             db.rollback()
159.
160.         return
161.
162.
163.     def modify_course(db):
164.         cursor = db.cursor()
165.         dic_course = {'cno': [], 'cname': [], 'cpno': [], 'ccredit': []
    }
```

```
166.       print("================== Mode 4: Modify Existing Course Info =
    ================")
167.       print("Please input the Course ID to search the course! ")
168.
169.       dic_course['cno'].append(input("Course ID: "))
170.
171.       sql = "SELECT * FROM Course WHERE Cno = '%s'" % dic_course['cno
    '][0]
172.
173.       try:
174.           cursor.execute(sql)
175.           results = cursor.fetchall()
176.           for index, row in enumerate(results):
177.               # dic_course['cno'].append(row[0])
178.               dic_course['cname'].append(row[1])
179.               dic_course['cpno'].append(row[2])
180.               dic_course['ccredit'].append(row[3])
181.
182.           print("Course founded! Here is the information of this cour
    se: ")
183.           df = pd.DataFrame(dic_course)
184.           print(tabulate(df, headers='keys', tablefmt='psql'))
185.       except:
186.           print("Error: unable to fetch data")
187.           return
188.
189.       header_list = ['', 'Cno', 'Cname', 'Cpno', 'Ccredit']
190.
191.       while True:
192.           modify_op = input("Select column 1~4 to modify, select 0 to
     exit: ")
193.           if modify_op == '0':
194.               break
195.           else:
196.               pass
197.           new_data = input("Input the new data: ")
198.
199.           sql = "UPDATE Course SET %s = " % header_list[int(modify_op
    )]
200.           if modify_op != 4:
201.               sql += "'""'"
202.           sql += new_data
203.           if modify_op != 4:
204.               sql += "'"
205.           sql += "WHERE Cno = '%s'" % dic_course['cno'][0]
206.
207.           try:
208.               print("Successfully Updated! ")
209.               cursor.execute(sql)
210.               db.commit()
211.           except:
212.               print("update error! ")
213.               db.rollback()
214.               return
215.
216.       return
217.
218.
219.   def delete_course(db):
220.       cursor = db.cursor()
221.       dic_course = {'cno': [], 'cname': [], 'cpno': [], 'ccredit': []
    }
222.       print("================== Mode 5: Delete Existing Course Info =
    ================")
223.       print("Please input the Course ID to search the course! ")
224.
```

```
225.       dic_course['cno'].append(input("Course ID: "))
226.
227.       sql = "SELECT * FROM Course WHERE Cno = '%s'" % dic_course['cno
    '][0]
228.
229.       try:
230.           cursor.execute(sql)
231.           results = cursor.fetchall()
232.           for index, row in enumerate(results):
233.               # dic_course['cno'].append(row[0])
234.               dic_course['cname'].append(row[1])
235.               dic_course['cpno'].append(row[2])
236.               dic_course['ccredit'].append(row[3])
237.
238.           print("Course founded! Here is the information of this cour
    se: ")
239.           df = pd.DataFrame(dic_course)
240.           print(tabulate(df, headers='keys', tablefmt='psql'))
241.       except:
242.           print("Error: unable to fetch data")
243.           return
244.
245.       header_list = ['', 'Cno', 'Cname', 'Cpno', 'Ccredit']
246.
247.       confirm_op = input("ARE YOU SURE TO DELETE THIS COURSE? (y/n):
    ")
248.       if confirm_op == 'n':
249.           return
250.       else:
251.           pass
252.
253.       sql = "DELETE FROM Course WHERE Cno = '%s'" % dic_course['cno']
    [0]
254.
255.       try:
256.           print("Successfully Deleted! ")
257.           cursor.execute(sql)
258.           db.commit()
259.       except:
260.           print("delete error! ")
261.           db.rollback()
262.           return
263.
264.       return
265.
266.
267.   def record_grade(db):
268.       cursor = db.cursor()
269.       dic_sc = {'sno': [], 'cno': [], 'grade': []}
270.       print("================== Mode 6: Record Student's Grade ======
    ============")
271.       print("Please input the following information! ")
272.
273.       dic_sc['sno'].append(input("Student ID: "))
274.       dic_sc['cno'].append(input("Course ID: "))
275.       dic_sc['grade'].append(input("Student's Grade: "))
276.
277.       df = pd.DataFrame(dic_sc)
278.
279.       print("Re-check the information below: ")
280.       print(tabulate(df, headers='keys', tablefmt='psql'))
281.
282.       op = input("ARE YOU SURE TO INSERT THE NEW INFORMATION? (y/n)")

283.       if op == 'n':
284.           print("rollback successfully! ")
```

```
285.             return
286.         else:
287.             pass
288.
289.         sql = "INSERT INTO SC " \
290.               "VALUES('%s', '%s', %s)" % (
291.                   dic_sc['sno'][0], dic_sc['cno'][0], dic_sc['grade'][0
    ])
292.
293.         try:
294.             cursor.execute(sql)
295.             db.commit()
296.             print("Successfully Inserted! ")
297.         except:
298.             print("An unexpected error occurred and we rollback all the
     changes. ")
299.             db.rollback()
300.
301.         return
302.
303.
304.     def modify_grade(db):
305.         cursor = db.cursor()
306.         dic_sc = {'sno': [], 'cno': [], 'grade': []}
307.         print("================== Mode 7: Modify Student's Grade ======
    ============")
308.         print("Please input the Student ID and Course ID to search the
    grade record! ")
309.
310.         dic_sc['sno'].append(input("Student ID: "))
311.         dic_sc['cno'].append(input("Course ID: "))
312.
313.         sql = "SELECT * FROM SC WHERE Sno = '%s' AND Cno = '%s'" % (dic
    _sc['sno'][0], dic_sc['cno'][0])
314.
315.         try:
316.             cursor.execute(sql)
317.             results = cursor.fetchall()
318.             for index, row in enumerate(results):
319.                 # dic_sc['sno'].append(row[0])
320.                 # dic_sc['cno'].append(row[1])
321.                 dic_sc['grade'].append(row[2])
322.
323.             print("Course founded! Here is the information of this cour
    se: ")
324.             df = pd.DataFrame(dic_sc)
325.             print(tabulate(df, headers='keys', tablefmt='psql'))
326.         except:
327.             print("Error: unable to fetch data")
328.             return
329.
330.         header_list = ['', 'Sno', 'Cno', 'Grade']
331.
332.         modify_op = input("Are you sure to modify this score? (y/n): ")
333.         if modify_op == 'n':
334.             return
335.         else:
336.             pass
337.         new_data = input("Input the new grade: ")
338.
339.         sql = "UPDATE SC SET Grade = %s WHERE Sno = '%s' AND Cno = '%s'
    " % (new_data, dic_sc['sno'][0], dic_sc['cno'][0])
340.
341.         try:
342.             print("Successfully Updated! ")
```

```
343.             cursor.execute(sql)
344.             db.commit()
345.         except:
346.             print("update error! ")
347.             db.rollback()
348.             return
349.
350.         return
351.
352.
353.     def analyze_grade(db):
354.         cursor = db.cursor()
355.         dic_sc = {'dept': [], 'avg': [], 'max': [], 'min': [], 'a_rate'
    : [], 'f_num': []}
356.         l_dept = []
357.         print("================== Mode 8: Print Analytical Info by Dept
     ==================")
358.         # print("Please input the Student ID and Course ID to search th
    e grade record! ")
359.
360.         # Step 1. acquire all of the dept names
361.         sql = "SELECT DISTINCT Sdept FROM Student ORDER BY Sdept"
362.         try:
363.             cursor.execute(sql)
364.             results = cursor.fetchall()
365.             for row in results:
366.                 l_dept.append(row[0])
367.         except:
368.             print("Error: unable to fetch department data")
369.             return
370.
371.         print("Department List: " + ",".join(str(x) for x in l_dept))
372.
373.         # Step 2. get the data of each dept
374.         for dept in l_dept:
375.             dic_sc['dept'].append(dept)
376.
377.             # fetch the avg, max, min number
378.             sql = "SELECT AVG(SC.Grade), MAX(SC.Grade), MIN(SC.Grade) F
    ROM Student, SC WHERE Student.Sdept = '%s' AND " \
379.                   "Student.Sno = SC.Sno" % dept
380.             try:
381.                 cursor.execute(sql)
382.                 results = cursor.fetchall()
383.                 for row in results:
384.                     dic_sc['avg'].append(row[0])
385.                     dic_sc['max'].append(row[1])
386.                     dic_sc['min'].append(row[2])
387.             except:
388.                 print("Error: unable to fetch avg/max/min data")
389.                 return
390.
391.             a_number = 0
392.             total_number = 0
393.
394.             # fetch the A number
395.             sql = "SELECT COUNT(*) FROM Student, SC WHERE Student.Sdept
     = '%s' AND Student.Sno = SC.Sno AND SC.Grade >= 90" % dept
396.             try:
397.                 cursor.execute(sql)
398.                 results = cursor.fetchall()
399.                 for row in results:
400.                     a_number = row[0]
401.             except:
402.                 print("Error: unable to fetch avg/max/min data")
403.                 return
```

```
404.
405.             # fetch all dept student number
406.             sql = "SELECT COUNT(*) FROM Student, SC WHERE Student.Sdept
     = '%s' AND Student.Sno = SC.Sno" % dept
407.             try:
408.                 cursor.execute(sql)
409.                 results = cursor.fetchall()
410.                 for row in results:
411.                     total_number = row[0]
412.             except:
413.                 print("Error: unable to fetch total student number data
    ")
414.                 return
415.
416.             if total_number != 0:
417.                 a_rate = a_number / total_number
418.             else:
419.                 a_rate = 0.0
420.             dic_sc['a_rate'].append(a_rate)
421.
422.             fail_num = 0
423.
424.             # fetch the failed students number
425.             sql = "SELECT COUNT(*) FROM Student, SC WHERE Student.Sdept
     = '%s' AND Student.Sno = SC.Sno AND SC.Grade < 60" % dept
426.             try:
427.                 cursor.execute(sql)
428.                 results = cursor.fetchall()
429.                 for row in results:
430.                     fail_num = row[0]
431.             except:
432.                 print("Error: unable to fetch avg/max/min data")
433.                 return
434.
435.             dic_sc['f_num'].append(fail_num)
436.
437.         # Step 3. display the information
438.         print("Here is the information for your inquire: ")
439.         df = pd.DataFrame(dic_sc)
440.         print(tabulate(df, headers='keys', tablefmt='psql'))
441.         input("Press Enter to continue...")
442.         return
443.
444.
445.     def rank_student(db):
446.         cursor = db.cursor()
447.         l_dept = []
448.         print("================== Mode 9: Print Department Student Rank
    ing ==================")
449.         # print("Please input the Student ID and Course ID to search th
    e grade record! ")
450.
451.         # Step 1. acquire all of the dept names
452.         sql = "SELECT DISTINCT Sdept FROM Student ORDER BY Sdept"
453.         try:
454.             cursor.execute(sql)
455.             results = cursor.fetchall()
456.             for row in results:
457.                 l_dept.append(row[0])
458.         except:
459.             print("Error: unable to fetch department data")
460.             return
461.
462.         while True:
463.             dic_sc = {'sno': [], 'sname': [], 'avg': [], 'c_num': [], '
    credits': []}
```

```
464.           print("Department List: " + ",".join(str(x) for x in l_dept
    ))
465.           while True:
466.               sel_dept = input("Please select one department to inqui
    re the student's ranking: ")
467.               if sel_dept in l_dept:
468.                   break
469.               else:
470.                   print('Department %s not found! ' % sel_dept)
471.
472.           # dic_sc = {'sno': [], 'sname': [], 'avg': [], 'c_num': [],
     'credits': []}
473.           # Step 2. get the data of the chosen dept
474.           sql = "SELECT Student.Sno, Student.Sname, AVG(SC.Grade), CO
    UNT(SC.Grade), SUM(Course.Ccredit) FROM Student, SC, " \
475.                 "Course WHERE Student.Sdept = '%s' AND SC.Sno = Stude
    nt.Sno AND Course.Cno = SC.Cno GROUP BY Student.Sno " \
476.                 "ORDER BY AVG(SC.Grade) DESC" % sel_dept
477.           try:
478.               cursor.execute(sql)
479.               results = cursor.fetchall()
480.               for row in results:
481.                   dic_sc['sno'].append(row[0])
482.                   dic_sc['sname'].append(row[1])
483.                   dic_sc['avg'].append(row[2])
484.                   dic_sc['c_num'].append(row[3])
485.                   dic_sc['credits'].append(row[4])
486.           except:
487.               print("Error: unable to fetch student's data")
488.               return
489.
490.           # Step 3. display the information
491.           print("Here is the ranking of %s department: " % sel_dept)

492.           df = pd.DataFrame(dic_sc)
493.           print(tabulate(df, headers='keys', tablefmt='psql'))
494.           sel_op = input("Do you want to search other department's ra
    nk? (y/n)")
495.           if sel_op == 'n':
496.               break
497.           else:
498.               pass
499.
500.       return
501.
502.
503.   def show_stu_info(db):
504.       cursor = db.cursor()
505.       dic_student = {'sno': [], 'sname': [], 'sex': [], 'age': [], 'd
    ept': [], 'scholarship': []}
506.       dic_course = {'cno': [], 'cname': [], 'grade': [], 'credits': [
    ], 'cpno': []}
507.       print("================== Mode 10: Lookup Student Basic Info ==
    ================")
508.       print("Please input the Student ID to search the student! ")
509.
510.       dic_student['sno'].append(input("Student ID: "))
511.
512.       sql = "SELECT * FROM Student WHERE Sno = '%s'" % dic_student['s
    no'][0]
513.
514.       try:
515.           cursor.execute(sql)
516.           results = cursor.fetchall()
517.           for index, row in enumerate(results):
518.               # dic_student['sno'].append(row[0])
```

```
519.                 dic_student['sname'].append(row[1])
520.                 dic_student['sex'].append(row[2])
521.                 dic_student['age'].append(row[3])
522.                 dic_student['dept'].append(row[4])
523.                 dic_student['scholarship'].append(row[5])
524.
525.             print("Student founded! Here is the information of this per
    son: ")
526.             df = pd.DataFrame(dic_student)
527.             print(tabulate(df, headers='keys', tablefmt='psql'))
528.         except:
529.             print("Error: unable to fetch data")
530.             return
531.
532.         # definition: dic_course = {'cno': [], 'cname': [], 'grade': []
    , 'credits': [], 'cpno': []}
533.         sql = "SELECT SC.Cno, Course.Cname, SC.Grade, Course.Ccredit, C
    ourse.Cpno FROM SC, Course WHERE SC.Sno = '%s' AND " \
534.               "SC.Cno = Course.Cno" % dic_student['sno'][0]
535.         try:
536.             cursor.execute(sql)
537.             results = cursor.fetchall()
538.             for index, row in enumerate(results):
539.                 dic_course['cno'].append(row[0])
540.                 dic_course['cname'].append(row[1])
541.                 dic_course['grade'].append(row[2])
542.                 dic_course['credits'].append(row[3])
543.                 dic_course['cpno'].append(row[4])
544.
545.             print("Student's Course Info: ")
546.             df = pd.DataFrame(dic_course)
547.             print(tabulate(df, headers='keys', tablefmt='psql'))
548.             input("Press Enter to continue ... ")
549.         except:
550.             print("Error: unable to fetch course data")
551.             return
552.
553.         return
554.
555.
556.     def main():
557.         ip_addr = input("Input the IP Address (default: localhost): ")
    or "localhost"
558.         usr = input("Username (default: yuanye): ") or "yuanye"
559.         passwd = input("Password (default: yuanye): ") or "yuanye"
560.         select_db = input("Select one database to operate (default: S_T
    _U201911808): ") or "S_T_U201911808"
561.
562.         print(" ... Connecting to the Database ... ")
563.
564.         db = pymysql.connect(
565.             host=ip_addr,
566.             user=usr,
567.             password=passwd,
568.             database=select_db
569.         )
570.
571.         cursor = db.cursor()
572.
573.         cursor.execute("SELECT VERSION()")
574.
575.         db_version = cursor.fetchone()
576.
577.         print("---------------- Student Management System ------------
    ---")
578.         print("         Author:  网安1902班    袁也    U201911808")
```

```
579.       print("         MySQL Version: {}".format(db_version[0]))
580.       print("  User: {}@{}   Database: {}".format(usr, ip_addr, selec
    t_db))
581.
582.       while True:
583.           print_menu()
584.           op = input("Choose what you want (0~10): ")
585.           if op == '0':
586.               break
587.           elif op == '1':
588.               insert_student(db)
589.           elif op == '2':
590.               update_student(db)
591.           elif op == '3':
592.               insert_course(db)
593.           elif op == '4':
594.               modify_course(db)
595.           elif op == '5':
596.               delete_course(db)
597.           elif op == '6':
598.               record_grade(db)
599.           elif op == '7':
600.               modify_grade(db)
601.           elif op == '8':
602.               analyze_grade(db)
603.           elif op == '9':
604.               rank_student(db)
605.           elif op == '10':
606.               show_stu_info(db)
607.           else:
608.               print("Wrong Input! ")
609.
610.       db.close()
611.       print("Thanks for using this database! Bye!")
612.
613.
614.   if __name__ == '__main__':
    615.       main()
```