



华中科技大学

数据库系统概论实验报告

姓 名：黄浩岩
学 院：网络空间安全学院
专 业：信息安全
班 级：
学 号：
指导教师：路松峰

分数	
教师签名	

2020 年 11 月 23 日

目 录

1 课程任务概述	1
2 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.....	10
3 SQL 的复杂操作	11
3.1 任务要求.....	11
3.2 完成过程.....	11
3.3 任务总结.....	22
4 SQL 的高级实验	23
4.1 任务要求.....	23
4.2 完成过程.....	23
4.3 任务总结.....	30
5 数据库设计	31
5.1 任务要求.....	31
5.2 完成过程.....	31
5.3 任务总结.....	34
6 课程总结	36
附录.....	37

1 课程任务概述

- (1) 熟练掌握 SQL 的数据定义语句 CREATE、ALTER、DROP、Select。
- (2) 掌握 SQL 语言的数据多表查询语句和更新操作。
- (3) 掌握 SQL 语言的视图、触发器、存储过程、安全等功能。
- (4) 掌握数据库设计和开发技巧。

2 数据库定义与基本操作

2.1 任务要求

- (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

2.2 完成过程

2.2.1 安装数据库

- (1) 安装数据库管理系统 DBMS

首先下载安装数据库 MySQL，然后下载安装 Navicat 15 for MySQL。通过 MySQL 的账号连接 Navicat 和 MySQL，即可通过 Navicat 操作 MySQL 数据库。

- (2) 基于可视化界面或者命令行窗口创建数据库，命名为 CSEDB_学号

选中左侧连接右键选择“新建数据库”，填入数据库名即可创建数据库。如图 2.1 所示。

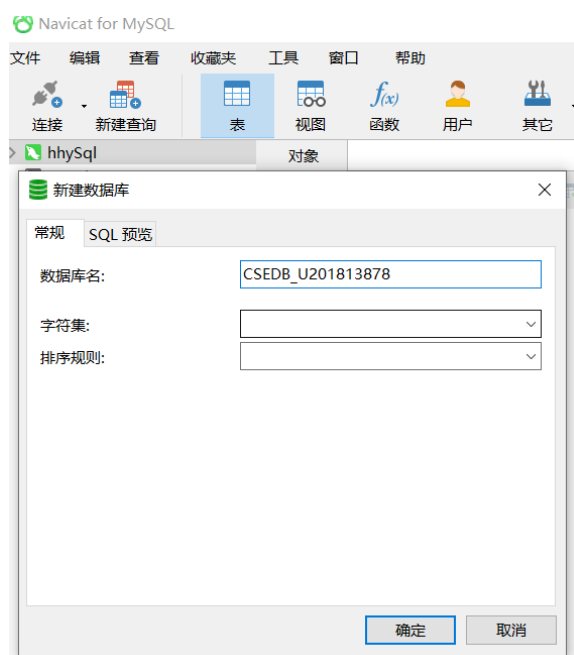


图 2.1 使用 Navicat 新建数据库

2.2.2 基本表操作

- (1) 创建、删除表

使用如下 SQL 语句创建 student 表、SC 表，创建后效果如图 2.2。

```
CREATE TABLE student (Sno CHAR(5) NOT NULL UNIQUE, Sname  
CHAR(20) UNIQUE, Ssex CHAR(1) , Sage INT, Sdept CHAR(15), Scholarship  
CHAR(2));
```

CREATE TABLE sc(Sno CHAR(5), Cno CHAR(3), Grade int, Primary key (Sno, Cno));



图 2.2 创建表 student 和 sc

使用如下 SQL 语句删除数据库 student，删除后效果如图 2.3。

DROP TABLE Student

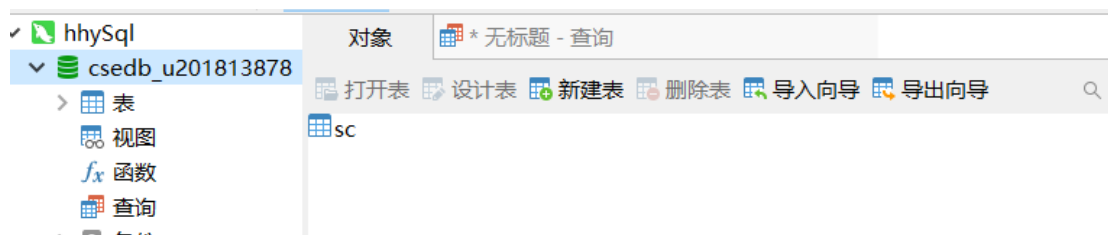


图 2.3 删除表 student

(2) 查看、修改表的定义

使用如下 SQL 语句给添加列 “Scome”，以及修改 “Sage” 列的数据类型为 SMALLINT。

```
ALTER TABLE student ADD Scome DATETIME;
ALTER TABLE student ALTER COLUMN Sage SMALLINT;
```

(3) 创建和删除索引

使用如下 SQL 语句分别为 student 表的 “Sno” 项、Course 表的 “Cno” 项和 sc 表中的 “Sno” 项和 “Cno” 项设置索引。

```
CREATE UNIQUE INDEX Stusno ON Student(Sno);
CREATE UNIQUE INDEX Coucno ON Course(Cno);
CREATE UNIQUE INDEX SCno ON SC(Sno ASC, Cno DESC);
```

2.2.3 删除数据库

使用以下 SQL 语句删除数据库 CSEDB_U201813878.

```
DROP DATABASE CSEDB_U201813878;
```

2.2.4 创建示例数据库 S_T_U201813878

如上述 2.2.2 中操作相同，在连接右键 “新建数据库”，在数据库名中输入 “S_T_U201813878” 即可创建该数据库。

2.2.5 在数据库 S_T_学号中创建 3 个表

(1) 创建学生表 Student、课程表 Course 和选修表 SC

利用 SQL 语句中的 Create Table 命令/或者可视化环境创建表，输入以下 SQL 语句创建表，创建效果如图 2.4.

```
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) );
```

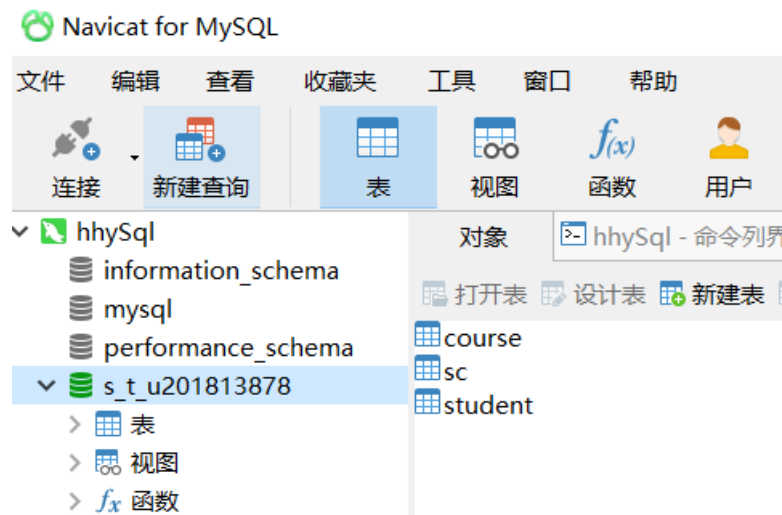


图 2.4 创建 S_T_U201813878 的 3 个表

(2) 在 3 个表中添加示例数据。

输入以下 SQL 语句分别为 3 个表添加数据，最后三个表的数据如图 2.5~2.7.

```
use s_t_u201813878; /*将 S_T 设为当前数据库*/
```

```
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');
```

```
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','PASCAL 语言', NULL,4);
```

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

```
update Course set Cpno = '5' 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';
```

```
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);

```

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

图 2.5 student 表示例数据

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

图 2.6 course 表示例数据

Sno	Cno	Grade
200215121	1	92
200215121	2	85
200215121	3	88
200215122	2	90
200215122	3	80

图 2.7 sc 表实例数据

2.2.6 对 3 个表进行查询

(1) 基本练习

① SELECT 语句的基本用法

查询全体学生的详细记录:

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

运行结果:

```

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

```

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 分以上的所有学生的学号、姓名：

SQL 语句：SELECT student.Sno,Sname FROM sc,student WHERE Cno='2' AND Grade>90 and sc.Sno=student.Sno;

运行结果：Empty set

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：

SQL 语句：SELECT Sname,Ssex FROM student WHERE Sdept IN ('IS', 'MA', 'CS');

运行结果：

```
+-----+-----+
| Sname | Ssex |
+-----+-----+
| 李勇  | 男   |
| 刘晨  | 女   |
| 王敏  | 女   |
| 张立  | 男   |
+-----+-----+
```

查询年龄在 20~23 岁（包括 20 岁和 23 岁）之间的学生的姓名、系别和年龄。

SQL 语句：SELECT Sname,Sdept,Sage FROM student WHERE Sage BETWEEN 20 AND 23;

运行结果：

```
+-----+-----+-----+
| Sname | Sdept | Sage |
+-----+-----+-----+
| 李勇  | CS    | 20   |
+-----+-----+-----+
```

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。

SQL 语句：SELECT Sname,Sno,Ssex FROM student WHERE Sname LIKE '刘%';

运行结果：

```
+-----+-----+-----+
| Sname | Sno      | Ssex |
+-----+-----+-----+
| 刘晨  | 200215122 | 女   |
+-----+-----+-----+
```

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。

SQL 语句: SELECT Sno,Grade FROM sc WHERE Cno='3' ORDER BY Grade DESC;

运行结果:

```
+-----+-----+
| Sno      | Grade |
+-----+-----+
| 200215121 | 88    |
| 200215122 | 80    |
+-----+-----+
```

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:

SQL 语句: SELECT AVG(Grade) FROM sc WHERE Cno='1';

运行结果:

```
+-----+
| AVG(Grade) |
+-----+
| 92.0000    |
+-----+
```

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:

SQL 语句: SELECT Sno FROM sc GROUP BY Sno HAVING COUNT(*)>3;

运行结果: Empty set

(2) 扩展练习

① 查询全体学生的学号、姓名和年龄;

SQL 语句: select Sno,Sname,Sage from student;

运行结果:

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

② 查询所有计算机系学生的详细记录;

SQL 语句: select * from student where Sdept='CS';

运行结果:

```
+-----+-----+-----+-----+-----+-----+
```

Sno	Sname	Ssex	Sage	Sdept	Scholarship
-----	-------	------	------	-------	-------------

200215121	李勇	男	20	CS	否
-----------	----	---	----	----	---

200215122	刘晨	女	19	CS	否
-----------	----	---	----	----	---

200215122	刘晨	女	19	CS	否
-----------	----	---	----	----	---

200215122	刘晨	女	19	CS	否
-----------	----	---	----	----	---

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;

SQL 语句: select Sno,Cno,Grade from sc where Grade>=90 or Grade<60;

运行结果:

Sno	Cno	Grade
-----	-----	-------

200215121	1	92
-----------	---	----

200215122	2	90
-----------	---	----

200215122	2	90
-----------	---	----

200215122	2	90
-----------	---	----

200215122	2	90
-----------	---	----

④ 查询年龄不在 19~20 岁之间的学生姓名、性别和年龄;

SQL 语句: select Sname,Ssex,Sage from student where Sage not between 19 and 20;

运行结果:

Sname	Ssex	Sage
-------	------	------

王敏	女	18
----	---	----

王敏	女	18
----	---	----

王敏	女	18
----	---	----

王敏	女	18
----	---	----

⑤ 查询数学系 (MA)、信息系 (IS)的学生的姓名和所在系;

SQL 语句: select Sname,Sdept from student where Sdept in ('MA', 'IS');

运行结果:

Sname	Sdept
-------	-------

王敏	MA
----	----

王敏	MA
----	----

王敏	MA
----	----

王敏	MA
----	----

王敏	MA
----	----

⑥ 查询名称中包含“数据”的所有课程的课程号、课程名及其学分;

SQL 语句: select Cno,Cname,Ccredit from course where Cname like '%数据%';

运行结果:

Cno	Cname	Ccredit
-----	-------	---------

王敏	MA
----	----

王敏	MA
----	----

王敏	MA
----	----

1	数据库	4
5	数据结构	4
6	数据处理	2
+-----+-----+-----+		

⑦ 找出所有没有选修课成绩的学生学号和课程号；

SQL 语句：select Sno,Cno from sc where Grade is null;

运行结果：Empty set

⑧ 查询学生 200215121 选修课的最高分、最低分以及平均成绩；

SQL 语句：select MAX(Grade),MIN(Grade),AVG(Grade) from sc where Sno='200215121';

运行结果：

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

⑨ 查询选修了 2 号课程的学生的学号及其成绩，查询结果按成绩升序排列；

SQL 语句：select Sno,Grade from sc where Cno='2' order by Grade ASC;

运行结果：

+-----+-----+	
Sno	Grade
+-----+-----+	
200215121	85
200215122	90
+-----+-----+	

⑩ 查询每个系名及其学生的平均年龄。

SQL 语句：select Sdept,AVG(Sage) from student group by Sdept;

运行结果：

+-----+-----+	
Sdept	AVG(Sage)
+-----+-----+	
CS	19.5000
MA	18.0000
IS	19.0000
+-----+-----+	

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

SQL 语句：select Sdept,AVG(Sage) from student group by Sdept having

AVG(Sage)<=19;

说明：此处学生的评价年龄是对不同的系别进行计算的，因此使用“group by Sdept”按系别分组，使用“having”子句来限制平均年龄，因为函数作为条件不能放在“where”子句而需放在“having”子句。

运行结果：

```
+-----+-----+
| Sdept | AVG(Sage) |
+-----+-----+
| MA    | 18.0000    |
| IS    | 19.0000    |
+-----+-----+
```

2.3 任务总结

实验中学习并实践了基本的一些 SQL 语句，包括建立表，删除表、修改表，以及一些基本的查询操作。

本次实验相对简单，基本没有遇到问题。

需要注意的就是书写 SQL 语句是的大小写问题：在实验中发现，MySQL 创建的数据库名和数据表名默认是字母是小写的，尽管输入的时候可能使用的是大写字母，可见 MySQL 对数据库名和数据表明是不区分字母大小写的，此外对应 SQL 语句中的关键字同样也不区分字母大小写。而对于数据表中的属性，即列名，则是区分大小写的。

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

3 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

3.2 完成过程

3.2.1 对 3 个表进行多表查询

(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。

SQL 语句：select student.*,sc.* from student,sc where student.Sno=sc.Sno;

运行结果：

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

查询每个学生及其选修课的情况（去掉重复列）。

SQL 语句：select student.Sno,Sname,Ssex,Sage,Sdept,Scholarship,Cno,Grade from student,sc where student.Sno=sc.Sno;

运行结果：

Sno	Sname	Ssex	Sage	Sdept	Scholarship	Cno	Grade
-----	-------	------	------	-------	-------------	-----	-------

200215121	李勇	男	20	CS	否	1	92
200215121	李勇	男	20	CS	否	2	85
200215121	李勇	男	20	CS	否	3	88
200215122	刘晨	女	19	CS	否	2	90
200215122	刘晨	女	19	CS	否	3	80

② 自身连接查询

查询每一门课的间接先修课。

SQL 语句: `select c1.Cno,c2.cpno from course c1,course c2 where c1.Cpno=c2.Cno;`

说明: 使用自身连接时需要为同一个表起 2 个别名予以区分

运行结果:

Cno	cpno
1	7
3	7
4	NULL
7	NULL
5	6

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情况)

SQL 语句 1: `select student.*,Cno,Grade from student left outer join sc on(student.Sno=sc.Sno);`

SQL 语句 2: `select student.*,Cno,Grade from sc right outer join student on(student.Sno=sc.Sno);`

说明: 外连接包括左外连接(列出左边关系的所有元组)和右外连接(列出右边关系的所有元组)。要包含所有未选课的学生,则可以如语句 1 把 student 放在 join 左边进行左外连接,也可以如语句 2 将学生表放在 join 右边进行右外连接,两语句是等价的。

运行结果:

Sno	Sname	Ssex	Sage	Sdept	Scholarship	Cno	Grade
200215121	李勇	男	20	CS	否	1	92
200215121	李勇	男	20	CS	否	2	85

200215121	李勇	男		20	CS	否		3		88	
200215122	刘晨	女		19	CS	否		2		90	
200215122	刘晨	女		19	CS	否		3		80	
200215123	王敏	女		18	MA	否		NULL		NULL	
200215125	张立	男		19	IS	否		NULL		NULL	

+-----+-----+-----+-----+-----+-----+-----+-----+

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。

SQL 语句: select sc.Sno,Sname from sc,student where Cno='2' and Grade>=90 and sc.Sno=student.Sno;

运行结果:

+-----+-----+
Sno Sname
+-----+-----+
200215122 刘晨
+-----+-----+

查询每个学生的学号、姓名、选修的课程名及成绩。

SQL 语句: select student.Sno,Sname,Cname,Grade from student,sc,course where student.Sno=sc.Sno and sc.Cno=course.Cno;

运行结果:

+-----+-----+-----+-----+
Sno Sname Cname Grade
+-----+-----+-----+-----+
200215121 李勇 数据库 92
200215121 李勇 数学 85
200215121 李勇 信息系统 88
200215122 刘晨 数学 90
200215122 刘晨 信息系统 80
+-----+-----+-----+-----+

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。

SQL 语句 1: select Sno,Sname,Sdept from student where Sdept = (select Sdept from student where Sname='刘晨');

SQL 语句 2: select s2.Sno,s2.Sname,s2.Sdept from student s1, student s2 where s1.Sname='刘晨' and s1.Sdept=s2.Sdept;

SQL 语句 3: select Sno,Sname,Sdept from student s1 where exists (select * from student s2 where s2.Sname='刘晨' and s1.Sdept=s2.Sdept);

说明：在元组的个数为 1 的情况下，“=”和“in”是可互换的，如语句 1。语句 2 使用了表的自身连接；语句 3 使用了“exist”谓词来判断存在性。

运行结果：

```

+-----+-----+-----+
| Sno      | Sname | Sdept |
+-----+-----+-----+
| 200215121 | 李勇  | CS    |
| 200215122 | 刘晨  | CS    |
+-----+-----+-----+

```

查询选修了课程名为“信息系统”的学生号和姓名。

SQL 语句：select Sno,Sname from student where Sno in (select Sno from sc where Cno in (select Cno from course where Cname='信息系统'));

运行结果：

```

+-----+-----+
| Sno      | Sname |
+-----+-----+
| 200215121 | 李勇  |
| 200215122 | 刘晨  |
+-----+-----+

```

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

SQL 语句：select Sno,Cno from sc x where Grade>(select AVG(Grade) from sc y where x.Sno=y.Sno);

运行结果：

```

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

```

⑦ 嵌套查询（带有 ANY 或 ALL 谓词的子查询）

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

SQL 语句 1：select Sname,Sage from student where Sdept!='CS' and Sage<any(select Sage from student where Sdept='CS');

SQL 语句 2：select Sname,Sage from student where Sdept!='CS' and Sage<(select MAX(Sage) from student where Sdept='CS');

说明：比系中某个学生的年龄小等价于比最大的学生年龄小，因此“<any”等同于“<MAX”

运行结果：

```
+-----+-----+
| Sname | Sage |
+-----+-----+
| 王敏   | 18   |
| 张立   | 19   |
+-----+-----+
```

查询其他系中比计算机系所有学生年龄都小的学生的姓名和年龄。

SQL 语句 1: select Sname,Sage from student where Sdept!='CS' and Sage<all(select Sage from student where Sdept='CS');

SQL 语句 2: select Sname,Sage from student where Sdept!='CS' and Sage<(select MIN(Sage) from student where Sdept='CS');

说明：同上，比系中所有学生年龄小等价于比年龄最小的学生小，因此，“<any”等同于“<MIN”。

运行结果：

```
+-----+-----+
| Sname | Sage |
+-----+-----+
| 王敏   | 18   |
+-----+-----+
```

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。

SQL 语句: select Sname from student where exists (select * from sc where Cno='1' and Sno=student.Sno);

说明：使用“exist”则理解为：结果中存在学生他选修了 1 号课程。

运行结果：

```
+-----+
| Sname |
+-----+
| 李勇   |
+-----+
```

查询所有未选修 1 号课程的学生姓名。

SQL 语句: select Sname from student where not exists (select * from sc where Cno='1' and Sno=student.Sno);

说明：使用“exist”则理解为：结果中不存在学生他选修了 1 号课程。

运行结果：

```
+-----+
| Sname |
```

```

+-----+
| 刘晨  |
| 张立  |
| 王敏  |

```

```

+-----+

```

查询选修了全部课程的学生姓名。

SQL 语句: select Sname from student where not exists (select * from course where not exists (select * from sc where Sno = student.Sno and Cno=course.Cno));

说明: 使用 “exist” 则理解为: 在结果中不存在这样的学生: 不存在一门课程在 course 里但是没有该学生选。

运行结果: Empty set

查询至少选修了学生 200215122 选修的全部课程的学生号码。

SQL 语句: select distinct Sno from sc sc1 where not exists (select * from sc sc2 where sc2.Sno='200215122' and not exists (select * from sc sc3 where sc2.Cno=sc3.Cno and sc1.Sno=sc3.Sno));

说明: 使用 “exist” 则理解为: 在结果中不存在学生满足: 不存在一门课程 122 学生选了而这个学生没选。

运行结果:

```

+-----+
| Sno      |
+-----+
| 200215121 |
| 200215122 |

```

```

+-----+

```

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。

SQL 语句 1: select * from student where Sdept='CS' union select * from student where Sage<=19;

SQL 语句 2: select * from student where Sdept='CS' or Sage<=19;

说明: 集合并运算相当于条件的或运算。

运行结果:

```

+-----+-----+-----+-----+-----+-----+
| 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 |
```

```
+-----+-----+-----+-----+-----+-----+
```

查询既选修了课程 1 又选修了课程 2 的学生（交集运算）。

SQL 语句 1: select Sno from sc where Cno='1' intersect select Sno from sc where Cno='2'; //MySQL 不支持

SQL 语句 2: select Sno from sc where Cno='1' and Sno in (select Sno from sc where Cno='2');

说明：集合的交运算相当于条件的与运算。

运行结果：

```
+-----+
```

```
| Sno |
```

```
+-----+
```

```
| 200215121 |
```

```
+-----+
```

查询计算机系的学生与年龄不大于 19 岁的学生的差集。

SQL 语句: select * from student where Sdept='CS' except select * from student where Sage<=19; //MySQL 不支持

SQL 语句: select * from student where Sdept='CS' and Sage>19;

说明：集合的差运算，也可以通过条件的与和非运算等同。

运行结果：

```
+-----+-----+-----+-----+-----+-----+
```

```
| Sno | Sname | Ssex | Sage | Sdept | Scholarship |
```

```
+-----+-----+-----+-----+-----+-----+
```

```
| 200215121 | 李勇 | 男 | 20 | CS | 否 |
```

```
+-----+-----+-----+-----+-----+-----+
```

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。

SQL 语句: update student set Sage=Sage+1 where Sdept='IS';

运行结果：如图 3.1.

student @s_t_u201813878 (hhySql) - 表

文件 编辑 查看 窗口 帮助						
开始事务 文本 筛选 排序 导入 导出						
Sno	Sname	Ssex	Sage	Sdept	Scholarship	
200215121	李勇	男	21	CS	否	
200215122	刘晨	女	20	CS	否	
200215123	王敏	女	19	MA	否	
200215125	张立	男	20	IS	否	

图 3.1 所有学生年龄+1 后结果

⑪ delete 语句用于对表进行删除

删除学号为 95019 的学生记录。

SQL 语句: delete from student where Sno='95019';

运行结果: 无该学号同学, 因此使用后无效果

⑫ insert 语句用于对表进行插入

插入一条选课记录('95020', '1')。

SQL 语句: insert into sc(Sno,Cno) values('95020','1');

说明: 由于 sc 表存在外键的约束, 要求插入表中的元组的 Sno 和 Cno 必须在 student 表和 course 表中出现, 此处 '95020' 未在 student 表出现, 因此无法被插入。

运行结果:

1452 - Cannot add or update a child row: a foreign key constraint fails (`s_t_u201813878`.`sc`, CONSTRAINT `sc_ibfk_1` FOREIGN KEY (`Sno`) REFERENCES `student` (`Sno`))

(2) 拓展练习

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)。

SQL 语句: select course.Cno,Cname,Sno,Grade from course left outer join sc on (course.Cno=sc.Cno);

说明: 通过要求可以知道, 需要列出 course 表中的全部课程, 因此使用外连接来处理。

运行结果:

Cno	Cname	Sno	Grade
1	数据库	200215121	92
2	数学	200215121	85
2	数学	200215122	90
3	信息系统	200215121	88
3	信息系统	200215122	80
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

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）

SQL 语句 1: select Sno,Sname,Sage from student where Sage= (select Sage from student where Sname='张立');

SQL 语句 2: select s1.Sno,s1.Sname,s1.Sage from student s1, student s2 where s1.Sage=s2.Sage and s2.Sname='张立';

SQL 语句 3: select Sno,Sname,Sage from student s1 where exists (select * from student s2 where s1.Sage=s2.Sage and s2.Sname='张立');

说明：语句 1 使用嵌套查询的方法，子查询查询出张立同学的年龄，外查询查询与该年龄相同的同学；语句 2 使用的是表 student 的自身连接；语句 3 使用“exist”谓词，即查询的结果要求存在他的年龄和张立的年龄相同。

运行结果：

Sno	Sname	Sage
200215122	刘晨	19
200215125	张立	19

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。

SQL 语句: select sc.Sno,Sname from sc,student where Cno='3' and Grade between 80 and 90 and sc.Sno=student.Sno;

运行结果：

Sno	Sname
200215121	李勇
200215122	刘晨

④ 查询学生 200215122 选修的课程号、课程名

SQL 语句: select sc.Cno,Cname from sc,course where Sno='200215122' and sc.Cno=course.Cno;

运行结果：

Cno	Cname
2	数学
3	信息系统

+-----+-----+

思考：如何查询学生 200215122 选修的课程号、课程名及成绩？

SQL 语句：select sc.Cno,Cname,Grade from sc,course where Sno='200215122' and sc.Cno=course.Cno;

说明：只需要在“select”子句中加上属性“Grade”。

运行结果：

+-----+-----+-----+

Cno	Cname	Grade
-----	-------	-------

+-----+-----+-----+

2	数学	90
---	----	----

3	信息系统	80
---	------	----

+-----+-----+-----+

⑤ 找出每个学生低于他所选修课程平均成绩 5 分以上的课程号。（输出学号和课程号）

SQL 语句：select Sno,Cno from sc sc1 where Grade+5<(select AVG(Grade) from sc sc2 where sc1.Sno=sc2.Sno group by Sno);

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

运行结果：Empty set

⑥ 查询比所有男生年龄都小的女生的学号、姓名和年龄。

SQL 语句：select Sno,Sname,Sage from student where Ssex='女' and Sage<all (select Sage from student where Ssex='男');

运行结果：

+-----+-----+-----+

Sno	Sname	Sage
-----	-------	------

+-----+-----+-----+

200215123	王敏	18
-----------	----	----

+-----+-----+-----+

⑦ 查询所有选修了 2 号课程的学生姓名及所在系。

SQL 语句：select Sname,Sdept from student,sc where student.Sno=sc.Sno and sc.Cno='2';

运行结果：

+-----+-----+

Sname	Sdept
-------	-------

+-----+-----+

李勇	CS
----	----

刘晨	CS
----	----

+-----+-----+

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:

```
update student set Sage=Sage+2 where Sno in (select distinct Sno from sc where Grade
between 80 and 90);
```

```
select student.Sno,Sage,Grade from student,sc where Grade between 80 and 90 and
student.Sno=sc.Sno;
```

运行结果:

+-----+-----+-----+

Sno	Sage	Grade
-----	------	-------

+-----+-----+-----+

200215121	22	85
-----------	----	----

200215121	22	88
-----------	----	----

200215122	21	90
-----------	----	----

200215122	21	80
-----------	----	----

+-----+-----+-----+

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:

```
insert into course(Cno,Cname) values('8','C 语言'), ('9','人工智能');
```

```
select Cname from course;
```

运行结果:

+-----+

Cname

+-----+

数据库

数学

信息系统

操作系统

数据结构

数据处理

PASCAL 语言

C 语言

人工智能

+-----+

⑩ 使用 delete 语句把人工智能课程删除，并查询出来。

SQL 语句:

```
delete from course where Cname='人工智能';
```

```
select Cname from course;
```

运行结果：

```
+-----+
| Cname  |
+-----+
| 数据库 |
| 数学   |
| 信息系统 |
| 操作系统 |
| 数据结构 |
| 数据处理 |
| PASCAL 语言 |
| C 语言   |
+-----+
```

3.3 任务总结

该部分实验主要实践了 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。

实验中比较困难的是写嵌套查询中的“相关子查询”语句，需要确定内部查询和外部查询的相关量，同时一般查询的最终结果是作为外部查询，而对结果的一些限制和条件则作为内部查询的结果。此外，含有“exist”谓词的嵌套查询也比较难写，一般需要将要求转换为“存在”或“不存在”组成的句子，必要时还需要借助逻辑中的量词、蕴含等对语句进行转换。

此外，在实验中发现 MySQL 的 SQL 语句不支持集合交谓词“intersect”和集合差谓词“except”，不过可以将集合的运算转换为“where”子句条件的运算，同样可以达到相同的查询目的。

4 SQL 的高级实验

4.1 任务要求

- (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

4.2.1 视图的相关操作

- (1) 创建 CS 系的视图 CS_View

SQL 语句: `create view CS_View as select * from student where Sdept='CS';`

说明: 使用“`create view <视图名> as <子查询>`”语句创建视图。

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

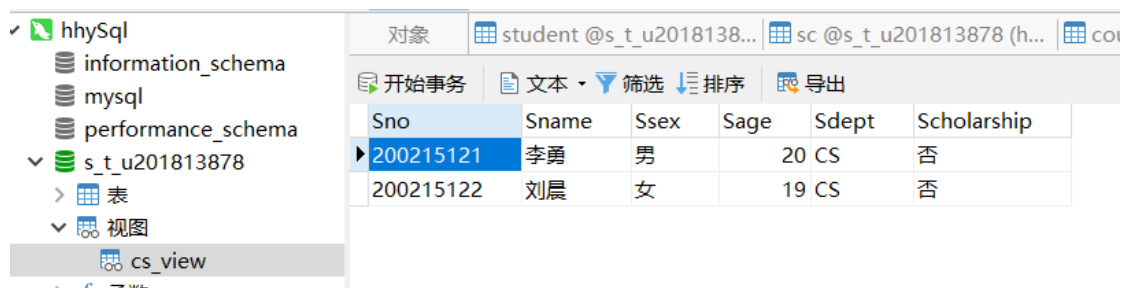


图 4.1 创建视图 CS_View

- (2) 在视图 CS_View 上查询 CS 系选修了 1 号课程的学生

SQL 语句: `select CS_View.Sno,Sname,Ssex,Sage,Sdept,Scholarship,Cno,Grade from CS_View,sc where CS_View.Sno=sc.Sno and Cno='1';`

说明: 对视图的查询语句与对普通表的查询语句是一样的。

运行结果:

```
+-----+-----+-----+-----+-----+-----+-----+
| Sno      | Sname | Ssex | Sage | Sdept | Scholarship | Cno | Grade |
+-----+-----+-----+-----+-----+-----+-----+
| 200215121 | 李勇  | 男   | 20   | CS    | 否          | 1   | 92    |
+-----+-----+-----+-----+-----+-----+-----+
```

- (3) 创建 IS 系成绩大于 80 的学生的视图 IS_View

SQL 语句: `create view IS_View as select student.Sno, Sname, Ssex, Sage, Sdept, Scholarship, Cno, Grade from student,sc where Sdept='IS' and student.Sno=sc.Sno and sc.Grade>80;`

运行结果: 如图 4.2.

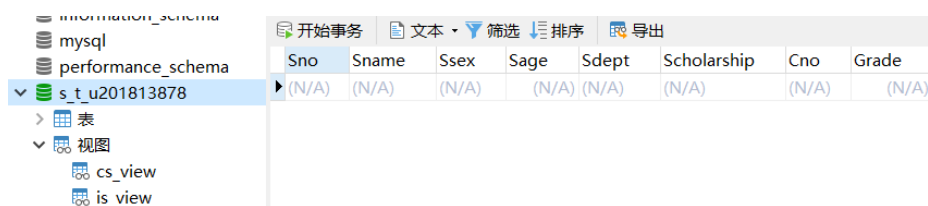


图 4.2 创建视图 IS_View

(4) 在视图 IS_View 查询 IS 系成绩大于 80 的学生

SQL 语句: `select distinct Sname from IS_View;`

运行结果: Empty set

(5) 删除视图 IS_View

SQL 语句: `drop view IS_View;`

说明: 使用语句 “`drop view <视图名>`” 删除视图。

4.2.2 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。

说明: 在 Navicat 的工具栏选择“用户”一项,选择“新建用户”,进行用户创建,如图 4.3。在输入框中填入用户名、主机名(localhost)以及密码即可,如图 4.4 和 4.5。



图 4.3 使用 Navicat 创建用户

常规	高级	成员属于	成员	服务器权限	权限	SQL 预览
用户名:	U1					
主机:	localhost					
插件:	caching_sha2_password					
密码:	●●●●●●●●					
确认密码:	●●●●●●●●					
密码过期策略:	DEFAULT					

图 4.4 创建用户 U1

常规	高级	成员属于	成员	服务器权限	权限	SQL 预览
用户名:	U1					
主机:	localhost					
插件:	caching_sha2_password					
密码:	●●●●●●●●					
确认密码:	●●●●●●●●					
密码过期策略:	DEFAULT					

图 4.5 创建用户 U2

然后使用“grant <权限> on <对象类型> <对象名> to <用户>”语句进行授权，SQL 语句如下：

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

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

(2) 用 U1 登录，分别

1) 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   | 20  | CS    | 否          |
| 200215122 | 刘晨  | 女   | 19  | CS    | 否          |
| 200215123 | 王敏  | 女   | 18  | MA    | 否          |
| 200215125 | 张立  | 男   | 19  | IS    | 否          |
+-----+-----+-----+-----+-----+-----+
```

2) 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   | 21  | CS    | 否          |
| 200215122 | 刘晨  | 女   | 20  | CS    | 否          |
| 200215123 | 王敏  | 女   | 19  | MA    | 否          |
| 200215125 | 张立  | 男   | 20  | IS    | 否          |
+-----+-----+-----+-----+-----+-----+
```

3) 删除 IS 系的学生；

SQL 语句: delete from student where Sdept='IS';

运行结果: 1142 - DELETE command denied to user 'U1'@'localhost' for table 'student'

说明: 由于用户 U1 没有删除 student 表的权限，因此无法执行。

4) 查询 CS 系 的选课信息。

SQL 语句: select * from student,sc where Sdept='CS' and student.Sno=sc.Sno;

运行结果: 1142 - SELECT command denied to user 'U1'@'localhost' for table 'sc'

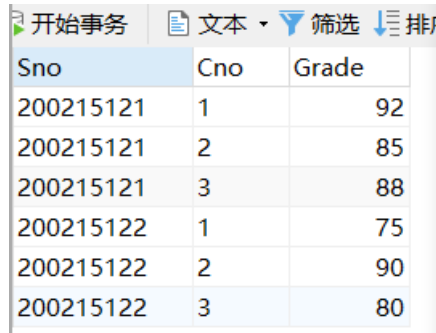
说明：由于用户 U1 没有对表 sc 的权限，因此查询失败。

(3) 用 U2 登录，分别

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

SQL 语句：insert into sc values('200215122','1',75);

运行效果：如图 4.6.



Sno	Cno	Grade
200215121	1	92
200215121	2	85
200215121	3	88
200215122	1	75
200215122	2	90
200215122	3	80

图 4.6 用户 U2 向 sc 表中插入数据

2) 查询 SC 表的信息

SQL 语句：select * from sc;

运行结果：1142 - SELECT command denied to user 'U2'@'localhost' for table '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。

3) 查询视图 CS_View 的信息。

SQL 语句：select * from CS_View;

运行结果：1142 - SELECT command denied to user 'U2'@'localhost' for table 'cs_view'

说明：与 2) 同理，用户 U2 没有对视图 CS_View 的权限，因此无法查询。

(3) 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;

(4) 用 U1 登录，查询学生表的信息

SQL 语句：use s_t_u201813878;

运行结果：1044 - Access denied for user 'U1'@'localhost' to database 's_t_u201813878'

说明：由于用户 U1 仅有对表 student 的权限，当权限撤回后 U1 对该数据库中的表均没有了权限，无法访问该数据库，因此选择使用 S_T_U201813878 数据库时就已经被拒绝了。

(5) 用系统管理员登录

4.2.3 触发器的相关操作

(1) 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”

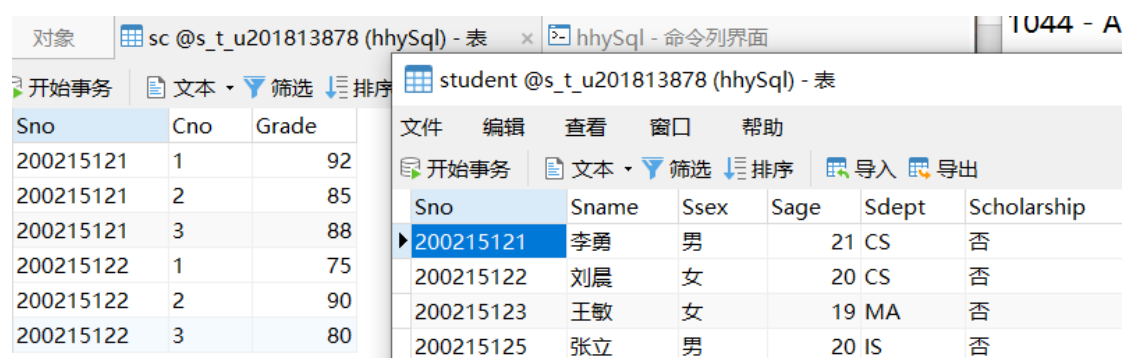
否”。

SQL 语句:

```
create trigger sc_t
after update on sc
for each row
begin
    if(new.Grade>=95) then
        update student set Scholarship='是' where student.Sno=new.Sno and
        Scholarship='否';
    else
        update student set Scholarship='否' where student.Sno=new.Sno and
        old.Grade>=95 and not exists (select * from sc where Sno=new.Sno and Grade>=95);
    end if;
end;
```

(2) 然后进行成绩修改, 并进行验证是否触发器正确执行。

如图 4.7 为原本的 student 表和 sc 表, 用于对照。



Sno	Cno	Grade
200215121	1	92
200215121	2	85
200215121	3	88
200215122	1	75
200215122	2	90
200215122	3	80

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

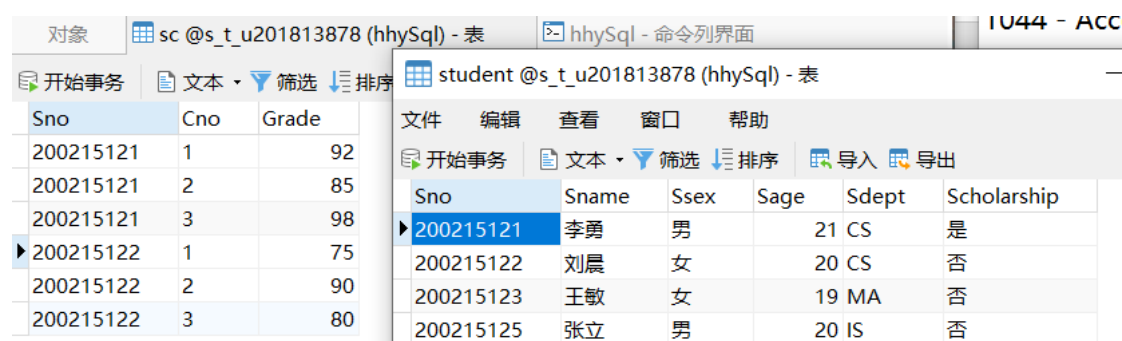
图 4.7 原本的 student 表和 sc 表

1) 首先把某个学生成绩修改为 98, 查询其奖学金。

SQL 语句: `update sc set Grade=98 where Sno='200215121' and Cno='3';`

说明: 将学号为 ‘200215121’ 的学生课程 ‘3’ 的成绩改为 98, 原为 88。

运行结果: 如图 4.8, 学号为 ‘200215121’ 的同学的奖学金 ‘Scholarship’ 属性已经由 ‘否’ 变为了 ‘是’。



Sno	Cno	Grade
200215121	1	92
200215121	2	85
200215121	3	98
200215122	1	75
200215122	2	90
200215122	3	80

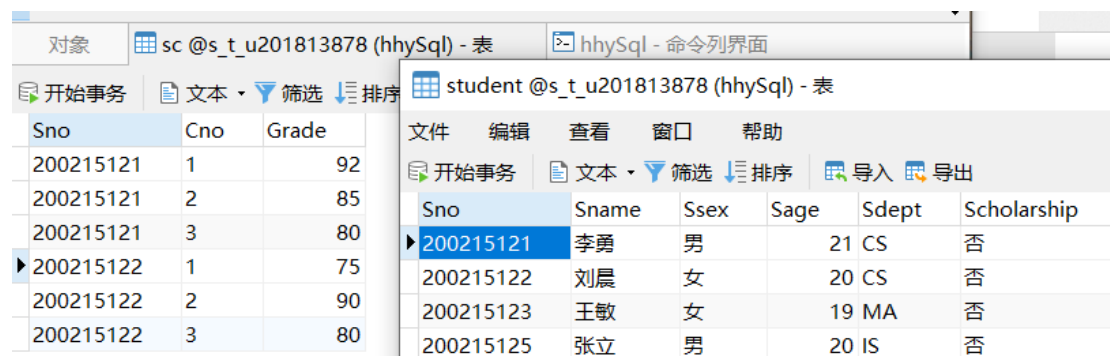
Sno	Sname	Ssex	Sage	Sdept	Scholarship
200215121	李勇	男	21	CS	是
200215122	刘晨	女	20	CS	否
200215123	王敏	女	19	MA	否
200215125	张立	男	20	IS	否

图 4.8 修改成绩为 98 后的 student 表和 sc 表

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='3';

运行结果: 如图 4.9, 该学生的奖学金属性又由‘是’变为了‘否’。



Sno	Cno	Grade
200215121	1	92
200215121	2	85
200215121	3	80
200215122	1	75
200215122	2	90
200215122	3	80

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

图 4.9 修改成绩为 80 后的 student 表和 sc 表

(3) 删除刚定义的触发器

SQL 语句: drop trigger sc_t;

4.2.4 存储过程和函数的相关操作

(1) 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 getCSAvgMax ()
begin
    select sc.Sno,AVG(Grade),MAX(Grade) from sc,student
    where student.Sno=sc.Sno and Sdept='CS' group by Sno;
end;
call getCSAvgMax();
```

运行结果:



```

1 create procedure getCSAvgMax ()
2 begin
3     select sc.Sno,AVG(Grade),MAX(Grade) from sc,student
4     where student.Sno=sc.Sno and Sdept='CS' group by Sno;
5 end;
6 call getCSAvgMax();

```

Sno	AVG(Grade)	MAX(Grade)
200215121	85.6667	92
200215122	81.6667	90

图 4.10 存储过程 getCSAvgMax 的运行结果

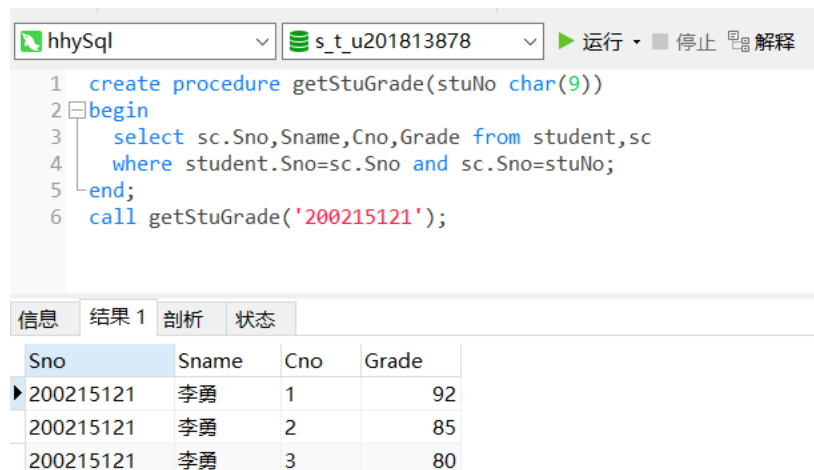
(2) 定义一个带学号为参数的查看某个学号的所有课程的成绩, 查询结果要包含

学生姓名。进行验证。

SQL 语句:

```
create procedure getStuGrade(stuNo char(9))
begin
    select sc.Sno,Sname,Cno,Grade from student,sc
    where student.Sno=sc.Sno and sc.Sno=stuNo;
end;
call getStuGrade('200215121');
```

运行结果:



The screenshot shows a MySQL IDE interface. The top toolbar includes a dropdown menu with 'hhySql', a session dropdown with 's_t_u201813878', and buttons for '运行' (Run), '停止' (Stop), and '解释' (Explain). The main editor area contains the following SQL code:

```
1 create procedure getStuGrade(stuNo char(9))
2 begin
3     select sc.Sno,Sname,Cno,Grade from student,sc
4     where student.Sno=sc.Sno and sc.Sno=stuNo;
5 end;
6 call getStuGrade('200215121');
```

Below the editor, the '结果 1' (Result 1) tab is active, displaying a table with the following data:

Sno	Sname	Cno	Grade
200215121	李勇	1	92
200215121	李勇	2	85
200215121	李勇	3	80

图 4.11 存储过程 getStuGrade 的运行结果

(3) 把上一题改成函数。再进行验证。

SQL 语句:

```
create function funcStuGrade(stuNo char(9)) returns char(100)
reads sql data
begin
    return (select group_concat(Sname,' ',Cno,' ',Grade) from student,sc
    where student.Sno=sc.Sno and sc.Sno=stuNo group by Sname);
end;
select funcStuGrade('200215121');
```

说明: 由于 MySQL 中函数的返回值只能是一个基本类型, 即 1 行 1 列的表, 因此此处使用“group_concat”函数将该名学生的信息拼接起来一同返回。使用 select 语句调用该函数。

4.2.5 完整性约束的相关操作

(1) 在 SC 表上定义一个完整性约束, 要求成绩再 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。

2) 定义约束后, 再把该学生成绩修改为 120, 然后进行查询。

SQL 语句: alter table sc add constraint graRange check(Grade between 0 and 100);

运行结果: 如图 4.12 显示了定义约束前后分别修改成绩 120 的结果, 可以看到

在添加约束后可以成功添加数据，在添加约束后，数据就拒绝被插入。

```
mysql> update sc set Grade=120 where Sno='200215122' and Cno='3';
Query OK, 1 row affected (0.01 sec)
Rows matched: 1 Changed: 1 Warnings: 0

mysql> select * from sc where Sno='200215122' and Cno='3';
+-----+-----+-----+
| Sno      | Cno | Grade |
+-----+-----+-----+
| 200215122 | 3   | 120   |
+-----+-----+-----+
1 row in set (0.02 sec)

mysql> update sc set Grade=80 where Sno='200215122' and Cno='3';
Query OK, 1 row affected (0.01 sec)
Rows matched: 1 Changed: 1 Warnings: 0

mysql> alter table sc add constraint graRange check(Grade between 0 and 100);
Query OK, 6 rows affected (0.09 sec)
Records: 6 Duplicates: 0 Warnings: 0

mysql> update sc set Grade=120 where Sno='200215122' and Cno='3';
3819 - Check constraint 'graRange' is violated.
mysql> select * from sc where Sno='200215122' and Cno='3';
+-----+-----+-----+
| Sno      | Cno | Grade |
+-----+-----+-----+
| 200215122 | 3   | 80    |
+-----+-----+-----+
1 row in set (0.02 sec)
```

图 4.12 添加约束的运行过程

4.3 任务总结

该部分实验实践了对数据库的一些高级操作，包括：视图、授权、触发器、存储过程和函数、完整性约束等。

该部分遇到的最大问题是对于以上这些高级操作，MySQL 的 SQL 语句的语法与书本上给出的语法基本上都有所不同，比如触发器语句中没有“referencing”部分，而是直接对“new”和“old”使用；已经删除触发器时不需要在语句后用“on”表明所属数据表等。实验中该部分主要通过上网查阅 MySQL 的相关语法资料进行解决书写的。

此外，在用户授权那一部分，在 SQL 语句中用户名不仅仅是创建时的“U1”“U2”，通过上网查阅资料确定，其后面还需要使用“@”符合表明用户的主机，如“U1@localhost”。

5 数据库设计

5.1 任务要求

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。
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 开发概述

实验中选择 python 作为此次学生管理系统的开发语言。使用 pymysql 库进行数据库的相关操作。其中，部分操作数据库的相关函数如下：

(1) 连接数据库：connect(**kwargs)

调用该函数可连接数据库，并返回数据库的一个对象 Connection。其中，**kwargs 可以传入一个字典，字典中包含待连接数据库的相关配置，包括，主机名(host)、用户名(user)、密码(password)、编码格式(charset)等等。

(2) 获取数据库游标：Connection.cursor()

调用该函数可以返回数据库对象 Connection 的一个游标 Cursor，利用该游标可以对数据库进行相关操作。

(3) 执行 SQL 语句：Cursor.execute(sql)

调用该函数可以执行 SQL 语句，其中 sql 为待执行的 SQL 语句字符串。

(4) 获取数据库执行后的数据：Cursor.fetchall()

调用该函数可以获取使用 Cursor.execute(sql)执行查询语句后得到的全部数据，函数的返回值可根据需求设置为字典或列表。

(5) 提交操作：Connection.commit()

一般在执行增删改等 SQL 语句成功执行后使用该函数，将事务进行提交。

(6) 回滚操作：Connection.rollback()

一般在增删改等 SQL 语句执行失败后调用该函数，将事务进行回滚。

5.2.2 功能实现效果

(1) 新生入学信息增加

此处，添加一名学生的信息：学号：200215130，姓名：张三，性别：男，年

龄：21，系别：IS。添加效果如图 5.1。

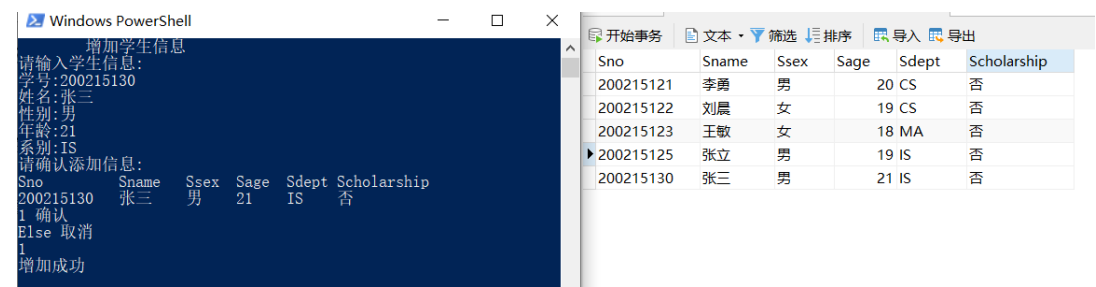


图 5.1 新生信息添加效果

(2) 学生信息修改

此处，修改学生张三的年龄为 18，系别为 MA。添加效果如图 5.2。

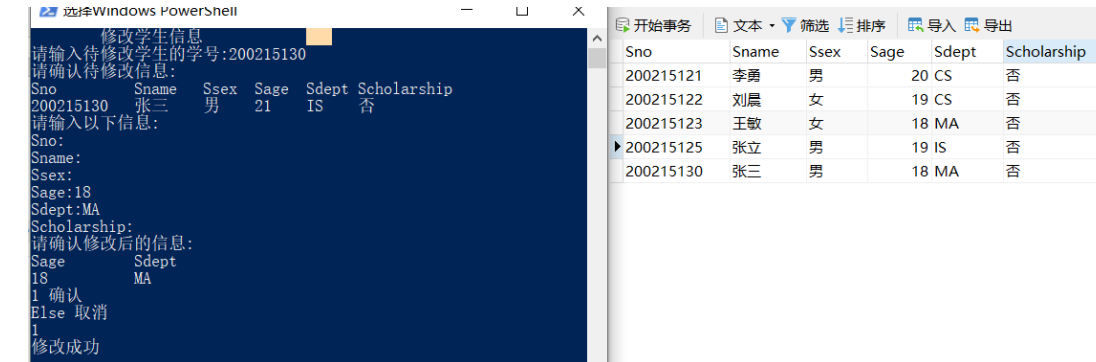


图 5.2 学生信息修改效果

(3) 增加课程信息

此处添加一门课程信息：课程号：9，课程名：C 语言，先修课号：2，学分：3。添加效果如图 5.3。

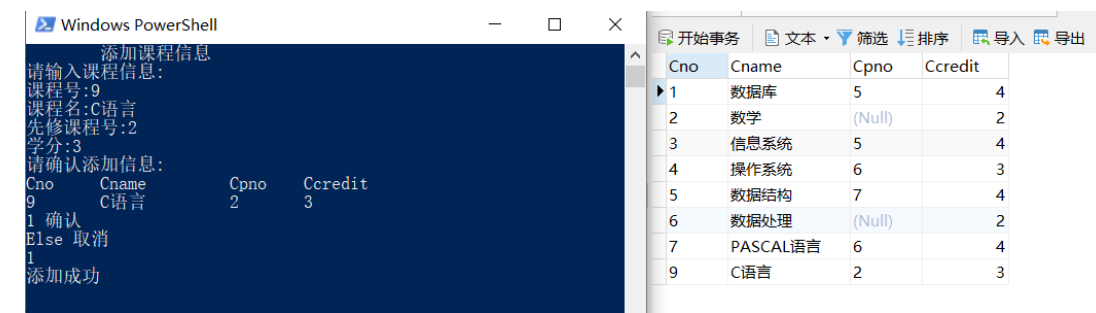


图 5.3 增加课程信息效果

(4) 修改课程信息

此处修改课程 9 的课程名为：C 语言设计，学分为 4。修改效果如图 5.4。

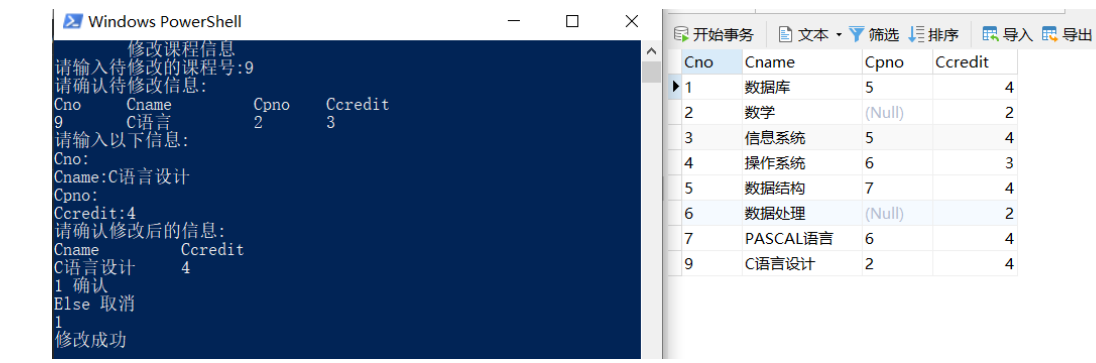


图 5.4 修改课程信息效果

(5) 删除没有选课的課程信息

删除效果如图 5.5.

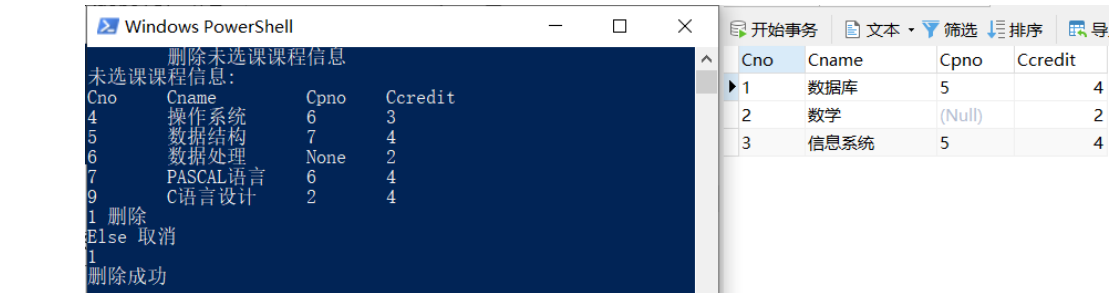


图 5.5 删除未选课的課程信息效果

(6) 录入学生成绩

此处录入一条学生成绩：学号：200215130，课程号：1，分数 91。录入效果如图 5.6.

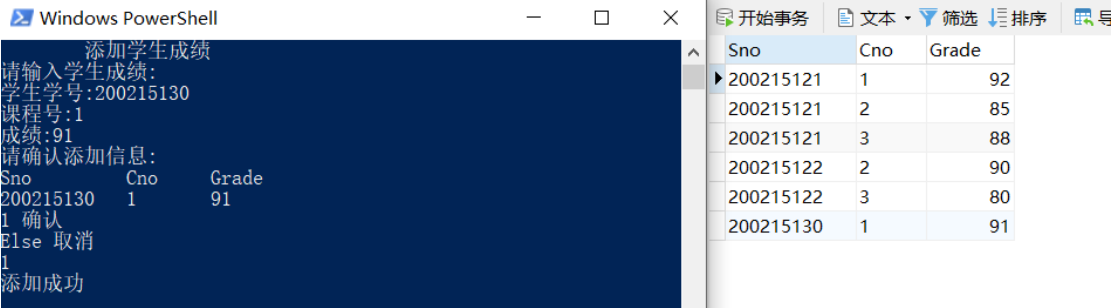


图 5.6 录入学生成绩效果

(7) 修改学生成绩

此处修改成绩信息：学号：200215130，课程号：1，分数 91 的课程号改为 3，分数改为 87。修改学生成绩效果如图 5.7。

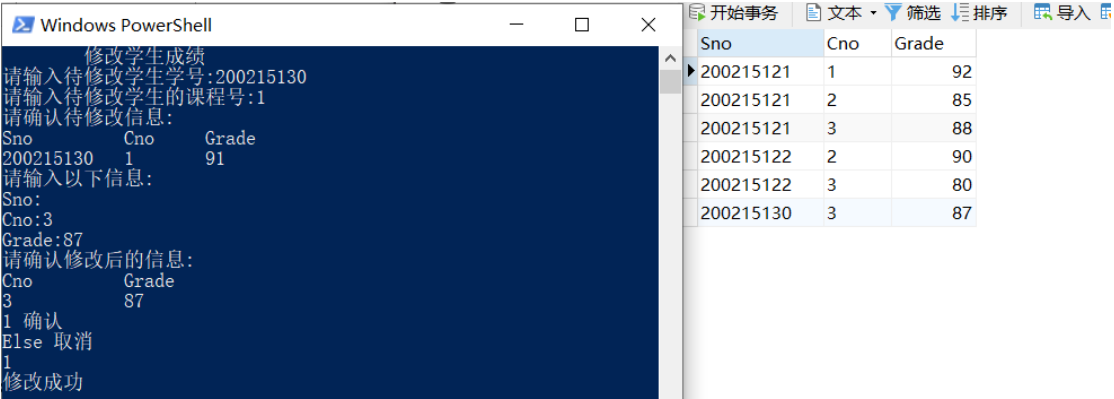


图 5.7 修改学生成绩

(8) 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

成绩统计效果如图 5.8。

```

选择Windows PowerShell
学生成绩统计
平均成绩:
Sdept  AVG(Grade)
CS      87.0000
MA      87.0000
最好成绩:
Sdept  MAX(Grade)
CS      92
MA      87
最差成绩:
Sdept  MIN(Grade)
CS      80
MA      87
优秀率:
Sdept  优秀率
CS      40.0%
不及格人数:
Sdept  不及格人数
按任意键返回

```

图 5.8 学生成绩统计效果

- (9) 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
成绩排名效果如图 5.9.

```

Windows PowerShell
学生成绩排名
CS:
Sno    Cno    Grade
200215121 1      92
200215122 2      90
200215121 3      88
200215121 2      85
200215122 3      80
MA:
Sno    Cno    Grade
200215130 3      87
按任意键返回

```

图 5.9 学生成绩排名效果

- (10) 输入学号，显示该学生的基本信息和选课信息
此处查询学号为 200215121 的基本信息和选课信息，效果如图 5.10.

```

Windows PowerShell
学生信息查询
请输入学号:200215121
基本信息:
Sno    Sname  Ssex  Sage  Sdept  Scholarship
200215121 李勇   男    20    CS     否
选课信息:
Cno    Grade
1      92
2      85
3      88
按任意键返回

```

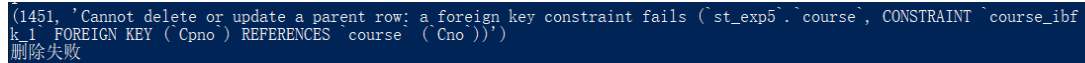
图 5.10 查询学生基本信息和选课信息效果

5.3 任务总结

该部分实验主要是学习实践了如何设计编程应用程序来控制操作数据库，在

学习了相关数据库操作的有关函数接口后，便可以顺利的对数据库进行操作。此处选择了 python 作为开发语言，其 pymysql 库提供了很方便使用的数据库接口，操作数据库比较简单，也使得该学生管理系统的开发难度大幅度降低。

实验中遇到了一个问题，即在实现删除未选课的课程信息时会报错，不能显示，如图 5.11.



```
(1451, 'Cannot delete or update a parent row: a foreign key constraint fails (`st_exp5`.`course`, CONSTRAINT `course_ibfk_1` FOREIGN KEY (`Cpno`) REFERENCES `course` (`Cno`))')
删除失败
```

图 5.11 删除未选课课程时报错信息

经过对报错信息的分析可知，建立数据库时，由于 sc 表的课程有一个约束要求其课程号必须出现在表 course 中，而数据库会为表 course 添加一条约束，因此在直接删除表 course 中未选课的课程时会被拒绝。此处在数据库中使用 SQL 语句：“alter table course drop constraint course_ibfk_1;”将对应的约束条件删除，即可正常执行删除操作。

6 课程总结

1. 学习掌握了 MySQL 数据库及其可视化客户端管理工具 Navicat 的使用。
2. 巩固实践了使用 SQL 语言对数据库的基本操作，包括创建数据库、创建数据表、增删改表中数据等。
3. 巩固实践了使用 SQL 语言对数据库中数据进行查询，包括简单的单表查询、等值连接查询、自身连接查询、嵌套查询、以及相关子查询、带有 exist 谓词的子查询、带有 all 或 any 谓词的子查询等复杂的查询语句。感受到 SQL 语句的复杂和技巧性，对于一些看似复杂的数据查询都可以转换成逻辑清晰的 SQL 语句。其中，个人感觉相关子查询和带有 exist 谓词的子查询两种嵌套查询比较复杂，对其掌握的不够熟练，同时 SQL 的查询语句语法规则复杂多样，还需要课下进一步实践学习和巩固。
4. 巩固实践了数据库中使用 SQL 语言完成的一些高级操作，包括创建表的视图、触发器的相关操作、存储过程和函数的创建和使用，对用户进行权限管理以及用户完整性约束等。在实验中，我感受到该部分中的一些 SQL 语句与所选的数据库有一定耦合性，与课本中描述的语法规则有些许差异，需要查阅相关资料进行调整。该部分中，对触发器和函数的使用还比较生疏，需要课下继续练习实践。
5. 学习了数据库的设计和开发技巧，编程实践了学生管理系统的开发。在这个过程中学习了使用 python 语言如何连接、操作数据库以及执行相关的 SQL 语句。但由于实验的时间有限，在管理系统的开发上还有一些设计的不足，比如输入合法性的判断、函数去耦合封装、错误处理等方面还有完善的空间，在课下可以进一步进行优化。

附录

学生管理系统源代码

main.py

```
import time
from utils import *
from config import *

mydb = MyDB(db_config)

def show_interface():
    go_on = True
    while go_on:
        op = '7'
        while op < '0' or op > '6':
            clear_cmd()
            print("\t 学生管理系统")
            print('1  学生信息维护')
            print('2  课程信息维护')
            print('3  学生成绩维护')
            print('4  学生成绩统计')
            print('5  学生成绩排名')
            print('6  学生信息查询')
            print('0  退出')
            op = input(op_tip)
        if op == '1':
            show_mt_stu()
        elif op == '2':
            show_mt_course()
        elif op == '3':
            show_mt_sc()
        elif op == '4':
            show_sc_grade()
        elif op == '5':
            show_sc_rank()
        elif op == '6':
            select_stu()
        else:
```

```
go_on = False
```

```
def select_stu():
    clear_cmd()
    print("\t 学生信息查询")
    sno = inputing('请输入学号:')
    condition = f'Sno="{sno}"'
    if not sno:
        print('学号为空,查询失败')
        time.sleep(sleep_time)
        return
    ret = mydb.select_sql('student', '*', condition)
    if not ret[1]:
        print('查询学生信息不存在')
        time.sleep(sleep_time)
        return
    print('基本信息:')
    print_list(stu_title, stu_format)
    print_sql(ret[1][0], stu_format)
    col = ['Cno', 'Grade']
    col_format = [8, 8]
    c = ""
    for s in col:
        c += s + ','
    ret = mydb.select_sql('sc', c.rstrip(','), condition)
    print('选课信息:')
    if ret[1]:
        print_list(col, col_format)
        for info in ret[1]:
            print_sql(info, course_format)
    else:
        print('无选课信息')
    op = input("\n 按任意键返回\n")
```

```
def show_sc_rank():
    clear_cmd()
    print("\t 学生成绩排名")
```



```

sdepts = mydb.select_sql('student', 'distinct Sdept')
# print(sdepts)
for sdept in sdepts[1]:
    ret = mydb.select_sql('sc,student', 'sc.Sno,Cno,Grade',
                          'Sdept="{' + sdept + '}" and sc.Sno=student.Sno order by
Grade desc'
                          .format(sdept['Sdept']))
    if ret[1]:
        print(sdept['Sdept'] + ':')
        print_list(sc_title, sc_format)
        for info in ret[1]:
            print_sql(info, sc_format)
op = input('\n 按任意键返回\n')

def show_sc_grade():
    clear_cmd()
    print('\t 学生成绩统计')
    titles = ['平均成绩', '最好成绩', '最差成绩']
    sdept_str = 'Sdept'
    grade_format = [8, 12]
    table_name = 'student,sc'
    where_str = 'student.Sno=sc.Sno group by Sdept'
    grade_str = ['AVG(Grade)', 'MAX(Grade)', 'MIN(Grade)']
    for i in range(len(grade_str)):
        col_str = sdept_str + ',' + grade_str[i]
        ret = mydb.select_sql(table_name, col_str, where_str)
        print(titles[i] + ':')
        if ret[0] == 1 and ret[1]:
            grade_title = [sdept_str, grade_str[i]]
            print_list(grade_title, grade_format)
            for info in ret[1]:
                print_sql(info, grade_format)
    title = '优秀率'
    cnt_str = 'COUNT(*)'
    print(title + ':')
    col_str = sdept_str + ',' + cnt_str
    total = mydb.select_sql(table_name, col_str, where_str)
    # print(total)
    if total[1]:

```

```

        good = mydb.select_sql(table_name, col_str,
                                'student.Sno=sc.Sno and Grade>=90 group by
Sdept')
        # print(good)
        t = [sdept_str, title]
        print_list(t, grade_format)
        for a, b in zip(good[1], total[1]):
            if b[cnt_str] != 0:
                print_list([a[sdept_str], str(a[cnt_str] / b[cnt_str] * 100) + '%'],
                            grade_format)

        title = '不及格人数'
        print(title + ':')
        bad = mydb.select_sql(table_name, col_str,
                                'student.Sno=sc.Sno and Grade<60 group by Sdept')

        # print(bad)
        t = [sdept_str, title]
        print_list(t, grade_format)
        for info in bad[1]:
            print_sql(info, grade_format)
        op = input('\n 按任意键返回\n')

```

```

def show_mt_sc():
    go_on = True
    while go_on:
        op = '6'
        while op < '0' or op > '2':
            clear_cmd()
            print("\t 学生成绩信息维护")
            print('1 添加学生成绩')
            print('2 修改学生成绩')
            print('0 返回')
            op = input(op_tip)
        if op == '1':
            add_sc()
        elif op == '2':
            update_sc()
        else:
            go_on = False

```

```

def add_sc():
    clear_cmd()
    print("\t 添加学生成绩")
    print('请输入学生成绩:')
    sno = inputing('学生学号:')
    cno = inputing('课程号:')
    grade = inputing('成绩:')
    sc = [sno, cno, grade]
    print('请确认添加信息:')
    print_list(sc_title, sc_format)
    print_list(sc, sc_format)
    op = input('1 确认\nElse 取消\n')
    if op == '1':
        if mydb.insert_sql('sc', sc) == 1:
            print('添加成功')
        else:
            print('添加失败')
        time.sleep(sleep_time)

def update_sc():
    clear_cmd()
    print("\t 修改学生成绩")
    sno = input('请输入待修改学生学号:')
    # sno = '20001010'
    cno = input('请输入待修改学生的课程号:')
    condition = f'Sno="{sno}"and Cno="{cno}"'
    ret = mydb.select_sql('sc', '*', condition)
    if ret[0] == 0:
        print('待修改学生成绩查询失败')
        time.sleep(sleep_time)
        return
    elif ret[1]:
        print('请确认待修改信息:')
        print_list(sc_title, sc_format)
        print_sql(ret[1][0], sc_format)

```

```

else:
    print('待修改学生成绩不存在')
    time.sleep(sleep_time)
    return
col = []
col_format = []
sc = []
print('请输入以下信息:')
for i in range(len(sc_title)):
    info = input(sc_title[i] + ':')
    if info:
        col.append(sc_title[i])
        col_format.append(sc_format[i])
        sc.append(info)
print('请确认修改后的信息:')
print_list(col, sc_format)
print_list(sc, sc_format)
op = input('1 确认\nElse 取消\n')
if op == '1':
    if mydb.update_sql('sc', col, sc, condition) == 1:
        print('修改成功')
    else:
        print('修改失败')
    time.sleep(sleep_time)

def show_mt_course():
    go_on = True
    while go_on:
        op = '6'
        while op < '0' or op > '3':
            clear_cmd()
            print("\t 课程信息维护")
            print('1 添加课程信息')
            print('2 修改课程信息')
            print('3 删除无选课的课程信息')
            print('0 返回')
            op = input(op_tip)
        if op == '1':

```

```

        add_course()
    elif op == '2':
        update_course()
    elif op == '3':
        delete_ept_course()
    else:
        go_on = False

```

```

def add_course():
    clear_cmd()
    print('\t 添加课程信息')
    print('请输入课程信息:')
    cno = inputing('课程号:')
    cname = inputing('课程名:')
    cpno = inputing('先修课程号:')
    ccredit = inputing('学分:')
    # sno, sname, ssex, sage, sdept = '20001010', 'hhy', '男', 20, 'IS'
    course = [cno, cname, cpno, ccredit]
    print('请确认添加信息:')
    print_list(course_title, course_format)
    print_list(course, course_format)
    op = input('1 确认\nElse 取消\n')
    if op == '1':
        if mydb.insert_sql('course', course) == 1:
            print('添加成功')
        else:
            print('添加失败')
        time.sleep(sleep_time)

```

```

def update_course():
    clear_cmd()
    print('\t 修改课程信息')
    cno = input('请输入待修改的课程号:')
    condition = f'Cno="{cno}"'
    ret = mydb.select_sql('course', '*', condition)
    if ret[0] == 0:

```

```

        print('待修改课程信息查询失败')
        time.sleep(sleep_time)
        return
    elif ret[1]:
        print('请确认待修改信息:')
        print_list(course_title, course_format)
        print_sql(ret[1][0], course_format)
    else:
        print('待修改课程信息不存在')
        time.sleep(sleep_time)
        return
    col = []
    col_format = []
    course = []
    print('请输入以下信息:')
    for i in range(len(course_title)):
        info = input(course_title[i] + ':')
        if info:
            col.append(course_title[i])
            col_format.append(course_format[i])
            course.append(info)
    print('请确认修改后的信息:')
    print_list(col, col_format)
    print_list(course, col_format)
    op = input('1 确认\nElse 取消\n')
    if op == '1':
        if mydb.update_sql('course', col, course, condition) == 1:
            print('修改成功')
        else:
            print('修改失败')
        time.sleep(sleep_time)

def delete_ept_course():
    clear_cmd()
    print("\t 删除未选课课程信息")
    condition = 'Cno not in (select distinct Cno from sc )'
    ret = mydb.select_sql('course', '*', condition)
    if ret[0] == 0:

```

```

        print('查询失败')
        return
    elif ret[1]:
        print('未选课课程信息:')
        print_list(course_title, course_format)
        for info in ret[1]:
            print_sql(info, course_format)
    else:
        print('无未选课课程')
        time.sleep(sleep_time)
        return
    op = input('1 删除\nElse 取消\n')
    if op == '1':
        if mydb.delete_sql('course', condition) == 1:
            print('删除成功')
        else:
            print('删除失败')
        time.sleep(sleep_time)

def show_mt_stu():
    go_on = True
    while go_on:
        op = '6'
        while op < '0' or op > '3':
            clear_cmd()
            print('\t 维护学生信息')
            print('1 添加学生信息')
            print('2 修改学生信息')
            print('3 删除学生信息')
            print('0 返回')
            op = input(op_tip)
        if op == '1':
            add_stu()
        elif op == '2':
            update_stu()
        elif op == '3':
            delete_stu()
        else:

```

```
go_on = False
```

```
def add_stu():
    clear_cmd()
    print("\t 增加学生信息")
    print('请输入学生信息:')
    sno = inputing('学号:')
    sname = inputing('姓名:')
    ssex = inputing('性别:')
    sage = inputing('年龄:')
    sdept = inputing('系别:')
    # sno, sname, ssex, sage, sdept = '20001010', 'hhy', '男', 20, 'IS'
    stu = [sno, sname, ssex, sage, sdept, '否']
    print('请确认添加信息:')
    print_list(stu_title, stu_format)
    print_list(stu, stu_format)
    op = input('1 确认\nElse 取消\n')
    if op == '1':
        if mydb.insert_sql('student', stu) == 1:
            print('增加成功')
        else:
            print('增加失败')
        time.sleep(sleep_time)
```

```
def update_stu():
    clear_cmd()
    print("\t 修改学生信息")
    sno = input('请输入待修改学生的学号:')
    # sno = '20001010'
    condition = f'Sno="{sno}"'
    ret = mydb.select_sql('student', '*', condition)
    if ret[0] == 0:
        print('待修改学生信息查询失败')
        time.sleep(sleep_time)
        return
    elif ret[1]:
```



```

        print('请确认待修改信息:')
        print_list(stu_title, stu_format)
        print_sql(ret[1][0], stu_format)
    else:
        print('待修改学生信息不存在')
        time.sleep(sleep_time)
        return
    col = []
    col_format = []
    stu = []
    print('请输入以下信息:')
    for i in range(len(stu_title)):
        info = input(stu_title[i] + ':')
        if info:
            col.append(stu_title[i])
            col_format.append(stu_format[i])
            stu.append(info)
    print('请确认修改后的信息:')
    print_list(col, stu_format)
    print_list(stu, stu_format)
    op = input('1 确认\nElse 取消\n')
    if op == '1':
        if mydb.update_sql('student', col, stu, condition) == 1:
            print('修改成功')
        else:
            print('修改失败')
        time.sleep(sleep_time)

def delete_stu():
    clear_cmd()
    print('\t 删除学生信息')
    sno = input('请输入待修改学生的学号:')
    # sno = '20001010'
    ret = mydb.select_sql('student', '*', f'Sno="{sno}"')
    if ret[0] == 0:
        print('待修改学生信息查询失败')
        time.sleep(sleep_time)
        return

```

```

elif ret[1]:
    print('请确认待修改信息:')
    print_list(stu_title, stu_format)
    print_sql(ret[1][0], stu_format)
else:
    print('待修改学生的信息不存在!')
    return
op = input('1 删除\nElse 取消\n')
if op == '1':
    if mydb.delete_sql('student', f'Sno="{sno}"') == 1:
        print('删除成功')
    else:
        print('删除失败')
    time.sleep(sleep_time)

def main():
    try:
        show_interface()
        print('\t 已退出')
    except RuntimeError:
        print('\t 无法连接数据库')
    finally:
        mydb.db.close()

if __name__ == '__main__':
    main()
    # add_course()
    # update_course()
    # add_stu()
    # update_stu()
    # delete_ept_course()
    # add_sc()
    # update_sc()
    # show_sc_stat()
    # show_sc_rank()
    # select_stu()
    pass

```

config.py

```
import pymysql

db_config = {
    'host': 'localhost',
    'user': 'root',
    'password': '123456qwe',
    'db': 'st_exp5',
    'charset': 'utf8mb4',
    'cursorclass': pymysql.cursors.DictCursor
}

op_tip = '请选择>> '
stu_title = ['Sno', 'Sname', 'Ssex', 'Sage', 'Sdept', 'Scholarship']
stu_format = [12, 8, 6, 6, 6, 8]
course_title = ['Cno', 'Cname', 'Cpno', 'Ccredit']
course_format = [8, 14, 8, 6]
sc_title = ['Sno', 'Cno', 'Grade']
sc_format = [12, 8, 8]
sleep_time = 10
```

utils.py

```
import os
import pymysql

def clear_cmd():
    os.system('cls')

def is_Chinese(ch):
    if '\u4e00' <= ch <= '\u9fa5':
        return True
    else:
        return False

def len_str(string):
    count = 0
    for line in string:
```

```

        if is_Chinese(line):
            count = count + 2
        else:
            count = count + 1
    return count

```

```

def printing(string, size=0, e='\n'):
    l = size - len_str(string)
    if l > 0:
        string += ' ' * l
    print(string, end=e)

```

```

def print_list(str_list, format_list):
    for s, l in zip(str_list, format_list):
        s = str(s)
        printing(s, l, "")
    print("")

```

```

def print_sql(str_dict, format_list):
    for s, l in zip(str_dict.values(), format_list):
        s = str(s)
        printing(s, l, "")
    print("")

```

```

def inputing(msg, is_int=False):
    ret = input(msg)
    if not ret:
        return None
    if is_int:
        try:
            return int(ret)
        except Exception as e:
            print(e)
            return None
    return ret

```

```

class MyDB:
    def __init__(self, cfg):
        self.config = cfg
        self.db = pymysql.connect(**self.config)
        self.cursor = self.db.cursor()

    def insert_sql(self, table_name, value_list):
        s = '%s,' * len(value_list)
        sql = 'insert into ' + table_name + \
            ' values(' + s.rstrip(',') + ')'
        # print(sql)
        try:
            self.cursor.execute(sql, value_list)
            self.db.commit()
            return 1
        except Exception as e:
            print(e)
            self.db.rollback()
            return 0

    def select_sql(self, table_name, col_str, where_str=""):
        if not where_str:
            sql = 'select ' + col_str + ' from ' + table_name
        else:
            sql = 'select ' + col_str + ' from ' + table_name + \
                ' where ' + where_str
        # print(sql)
        try:
            self.cursor.execute(sql)
            data = self.cursor.fetchall()
            flag = 1
            return flag, data
        except Exception as e:
            print(e)
            return 0, None

    def update_sql(self, table_name, col_list, value_list, where_str):
        s = "
        for i in col_list:

```

```

        s += i + '=%s,'
    sql = 'update ' + table_name + ' set ' + s.rstrip(',') + \
        ' where ' + where_str
    # print(sql)
    try:
        self.cursor.execute(sql, value_list)
        self.db.commit()
        return 1
    except Exception as e:
        print(e)
        self.db.rollback()
        return 0

def delete_sql(self, table_name, where_str):
    sql = 'delete from ' + table_name + ' where ' + where_str
    try:
        self.cursor.execute(sql)
        self.db.commit()
        return 1
    except Exception as e:
        print(e)
        self.db.rollback()
        return 0

```