

华中科技大学

数据库系统概论实验报告

姓 名：鄢湧棚
学 院：网络空间安全学院
专 业：网络空间安全
班 级：网安 1902 班
学 号：U201911741
指导教师：路松峰

分数	
教师签名	

2021 年 12 月 26 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.....	12
3 SQL 的复杂操作	13
3.1 任务要求.....	13
3.2 完成过程.....	13
3.3 任务总结.....	20
4 SQL 的高级实验	21
4.1 任务要求.....	21
4.2 完成过程.....	21
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.....	33
6 课程总结	34
附录.....	35

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

本次实验采用微软的 Microsoft SQL Server 数据库，使用轻量级可视化工具 Azure Data Studio 进行数据库操作。

使用可视化工具新建数据库：

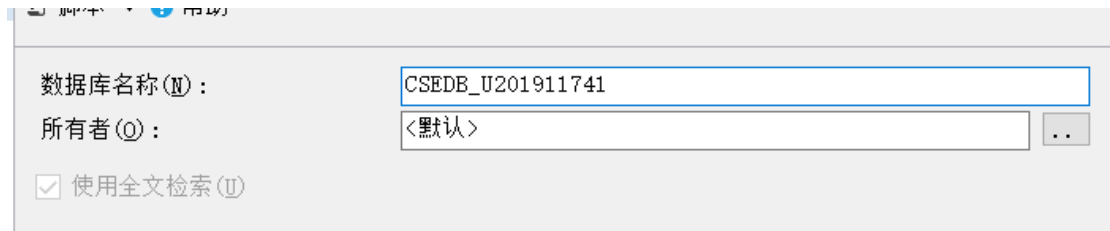


图 2.1 新建数据库

2.2.2 基本表操作

在数据库中创建三个表：

学生表：Student(Sno, Sname, Ssex, Sage, Sdept, Scholarship)

课程表：Course(Cno, Cname, Cpno, Ccredit)

学生选课表：SC(Sno, Cno, Grade)

使用 SQL 语句为：

```
Use CSEDB_U201911741;
```

```
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 Course
```

```
(
```

```

        Cno CHAR(3),
        Cname CHAR(20),
        Cpno INT,
        Ccredit INT
    )
CREATE TABLE SC(
    Sno CHAR(5),
    Cno CHAR(3),
    Grade int,
    Primary key (Sno, Cno)
);

```

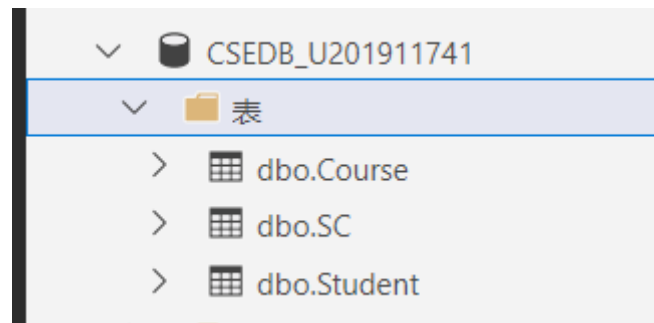


图 2.2 建立表

删除表 SC，使用 SQL 语句为：

```
DROP TABLE SC;
```

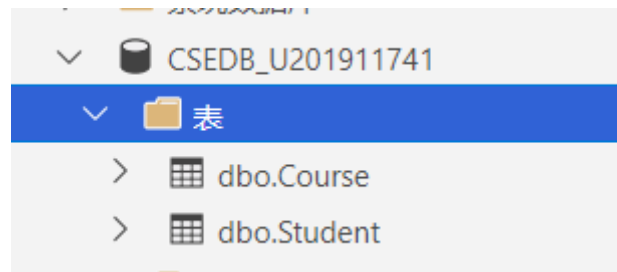


图 2.3 删除表

修改该 Student 表，添加入学时间，将 Sage 的类型改为 SMALLINT：

```
ALTER TABLE Student ADD Scome DATETIME;
```

```
ALTER TABLE Student ALTER COLUMN Sage SMALLINT;
```

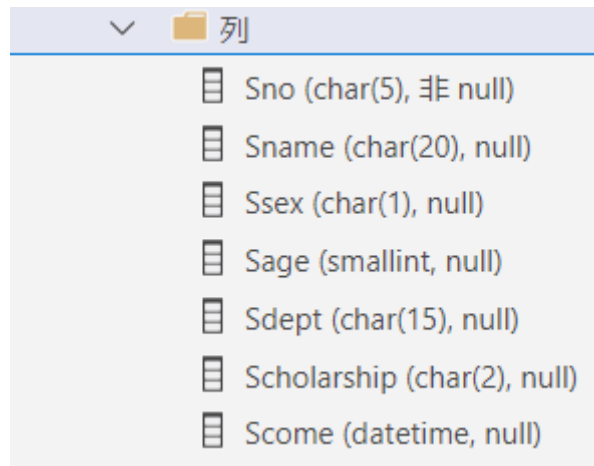


图 2.4 修改表

为三个表创建索引：

CREATE UNIQUE INDEX Stusno ON Student(Sno);

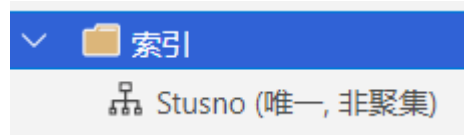


图 2.5 Student 表索引

CREATE UNIQUE INDEX Coucno ON Course(Cno);

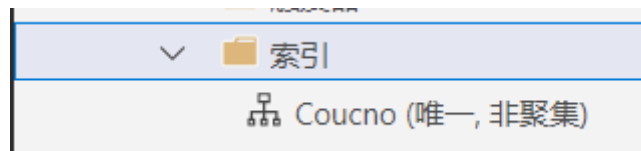


图 2.6 Course 表索引

CREATE UNIQUE INDEX SCno ON SC(Sno ASC, Cno DESC);



图 2.7 SC 表索引

删除索引使用如下 SQL 语句：

DROP INDEX Stusno on Student;

2.2.3 删除数据库

删除数据库使用如下 SQL 语句：

DROP DATABASE CSEDB_U201911741;

2.2.4 创建示例数据库 S_T_学号

使用 SQL 语句: CREATE DATABASE S_T_U201911741;

2.2.5 在数据库 S_T_学号中创建 3 个表并添加数据

表 Student 的主码为 Sno, 属性列 Sname 取唯一值

```
create table Student
```

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

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

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

```
Sage SMALLINT,
```

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

```
Scholarship char(2)
```

```
);
```

表 Course 的主码为 Cno, 属性列 Cpno(先修课)为外码, 被参照表为 Course, 被参照列是 Cno

```
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)
```

```
);
```

插入数据:

```
use S_T_U201911741;
```

```
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);

```

更新数据:

```

update Course set Cpno = '5' where Cno = '1';
update Course set Cpno = '1' where Cno = '3';
update Course set Cpno = '6' where Cno = '4';
update Course set Cpno = '7' where Cno = '5';
update Course set Cpno = '6' where Cno = '7';
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
1	200215121	李勇	男	20	CS	否
2	200215122	刘晨	女	19	CS	否
3	200215123	王敏	女	18	MA	否
4	200215125	张立	男	19	IS	否

图 2.8 Student 表

	Sno	Cno	Grade
1	200215121	1	92
2	200215121	2	85
3	200215121	3	88
4	200215122	2	90
5	200215122	3	80

图 2.9 SC 表

	Cno	Cname	Cpno	Ccredit
1	1	数据库 ...	5	4
2	2	数学 ...	NULL	2
3	3	信息系统 ...	1	4
4	4	操作系统 ...	6	3
5	5	数据结构 ...	7	4
6	6	数据处理 ...	NULL	2
7	7	PASCAL 语言 ...	6	4

图 2.10 Course 表

2.2.6 对 Student, Course, SC 表进行查询

查询全体学生的详细记录。

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

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

图 2.11 查询结果

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

```
SELECT Student.Sno, student.Sname
```

```
FROM Student, SC
```

```
WHERE Student.Sno = SC.Sno AND SC.Cno= '2 ' AND SC.Grade > 90;
```

Sno	Sname
-----	-------

图 2.12 查询结果

查询信息系（IS）、数学系（MA）和计算机科学系（CS）学生的姓名和性别。

```
SELECT Sname,Ssex
```

```
FROM Student
```

```
WHERE Sdept IN ( 'IS','MA','CS' );
```

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

图 2.13 查询结果

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。

```
SELECT Sname,Sdept,Sage
FROM Student
WHERE Sage BETWEEN 20 AND 23;
```

	Sname	Sdept	Sage
1	李勇	CS	20

图 2.14 查询结果

查询所有姓刘学生的姓名、学号和性别。

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

	Sname	Sno	Ssex
1	刘晨	200215122	女

图 2.15 查询结果

查询选修了 3 号课程的学生的学号及其成绩，查询结果按分数降序排列。

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

	Sno	Grade
1	200215121	88
2	200215122	80

图 2.16 查询结果

计算 1 号课程的学生平均成绩。

```
SELECT AVG(Grade)
FROM SC
```

WHERE Cno= '1';

(无列名称)	
1	92

图 2.17 查询结果

查询选修了 3 门以上课程的学生学号。

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

Sno

图 2.18 查询结果

2.2.7 扩展练习

查询全体学生的学号、姓名和年龄；

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

	Sno	Sname	Sage
1	200215121	李勇	20
2	200215122	刘晨	19
3	200215123	王敏	18
4	200215125	张立	19

图 2.19 查询结果

查询所有计算机系学生的详细记录；

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

	Sno	Sname	Ssex	Sage	Sdept	Scholarship
1	200215121	李勇	男	20	CS	否
2	200215122	刘晨	女	19	CS	否

图 2.20 查询结果

找出考试成绩为优秀（90 分及以上）或不及格的学生的学号、课程号及成绩；

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

	Sno	Cno	Grade
1	200215121	1	92
2	200215122	2	90

图 2.20 查询结果

查询年龄不在 19~20 岁之间的学生姓名、性别和年龄；

SELECT Sname, Ssex, Sage FROM Student WHERE Sage NOT BETWEEN 19 and 20;

	Sname	Ssex	Sage
1	王敏	女	18

图 2.20 查询结果

查询数学系（MA）、信息系（IS)的学生的姓名和所在系；

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

	Sname	Sdept
1	王敏	MA
2	张立	IS

图 2.20 查询结果

查询名称中包含“数据”的所有课程的课程号、课程名及其学分；

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

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

图 2.20 查询结果

找出所有没有选修课成绩的学生学号和课程号；

SELECT DISTINCT Student.Sno, Cno FROM SC, Student WHERE (Student.Sno = SC.Sno AND Grade IS NULL) OR Student.Sno NOT IN (SELECT Sno From SC);

	Sno	Cno
1	200215123	1
2	200215123	2
3	200215123	3
4	200215125	1
5	200215125	2
6	200215125	3

图 2.20 查询结果

查询学生 200215121 选修课的最高分、最低分以及平均成绩；

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

	MAX	MIN	AVG
1	92	85	88

图 2.20 查询结果

查询选修了 2 号课程的学生的学号及其成绩，查询结果按成绩升序排列；

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

	Sno	Grade
1	200215121	85
2	200215122	90

图 2.20 查询结果

查询每个系名及其学生的平均年龄。

SELECT Sdept, AVG(Sage) FROM Student Group BY Sdept;

	Sdept	AVG
1	CS	19
2	IS	19
3	MA	18

图 2.20 查询结果

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？

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

	AVG	Sdept
1	19	CS
2	19	IS
3	18	MA

图 2.20 查询结果

2.3 任务总结

简单的 sql 语句书写，没碰见什么问题，但是发现 sql server 对大小写不敏感，如对 student 表，定义的时候使用的是 Student，然而在写 sql 语句的时候使用 student 也可以。

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
- (7) 记录实验结果，认真完成实验报告

3.2 完成过程

3.2.1 基本练习

查询每个学生及其选修课的情况。

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

	Sno	Sname	Ssex	Sage	Sdept	Scholarship	Sno	Cno	Grade
1	200215121	李勇	男	26	CS	否	200215121	1	92
2	200215121	李勇	男	26	CS	否	200215121	2	85
3	200215121	李勇	男	26	CS	否	200215121	3	88
4	200215122	刘晨	女	25	CS	否	200215122	2	90
5	200215122	刘晨	女	25	CS	否	200215122	3	80

图 3.1 查询结果

查询每个学生及其选修课的情况（去掉重复列）

SELECT Student.Sno,Sname,Ssex,Sage,Cno,Grade FROM Student,SC WHERE Student.Sno = SC.Sno;

	Sno	Sname	Ssex	Sage	Cno	Grade
1	200215121	李勇	男	26	1	92
2	200215121	李勇	男	26	2	85
3	200215121	李勇	男	26	3	88
4	200215122	刘晨	女	25	2	90
5	200215122	刘晨	女	25	3	80

图 3.2 查询结果

查询每一门课的间接先修课。

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

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

图 3.3 查询结果

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

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

	Sno	Sname	Ssex	Sage	Sdept	Cno	Grade
1	200215121	李勇	男	26	CS	1	92
2	200215121	李勇	男	26	CS	2	85
3	200215121	李勇	男	26	CS	3	88
4	200215122	刘晨	女	25	CS	2	90
5	200215122	刘晨	女	25	CS	3	80
6	200215123	王敏	女	18	MA	NULL	NULL
7	200215125	张立	男	21	IS	NULL	NULL

图 3.4 查询结果

查询选修了 2 号课程而且成绩在 90 以上的所有学生的学号和姓名。

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

	Sno	Sname
1	200215122	刘晨

图 3.5 查询结果

查询每个学生的学号、姓名、选修的课程名及成绩。

SELECT Student.Sno, Sname, Cname, Grade FROM Student, SC, Course WHERE Student.Sno = SC.Sno AND SC.Cno = Course.Cno;

	Sno	Sname	Cname	Grade
1	200215121	李勇	数据库	92
2	200215121	李勇	数学	85
3	200215121	李勇	信息系统	88
4	200215122	刘晨	数学	90
5	200215122	刘晨	信息系统	80

图 3.6 查询结果

查询与“刘晨”在同一个系学习的学生的学号、姓名和所在系。

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

	Sno	Sname	Sdept
1	200215121	李勇	CS
2	200215122	刘晨	CS

图 3.7 查询结果

查询选修了课程名为“信息系统”的学生号和姓名。

```
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'
    )
);
```

	Sno	Sname
1	200215121	李勇
2	200215122	刘晨

图 3.8 查询结果

找出每个学生超过他所选修课程平均成绩的课程号。

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

	Sno	Cno
1	200215121	1
2	200215121	3
3	200215122	2

图 3.9 查询结果

查询其他系中比计算机系某个学生年龄小的学生的姓名和年龄。

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

	Sname	Sage
1	王敏	18
2	张立	21

图 3.10 查询结果

查询所有选修了 1 号课程的学生姓名。

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

	Sname
1	李勇

图 3.11 查询结果

查询所有未选修 1 号课程的学生姓名。

```
SELECT Sname FROM Student WHERE NOT EXISTS (SELECT* FROM SC WHE
RE Sno=Student.Sno AND Cno='1');
```

	Sname
1	刘晨
2	王敏
3	张立

图 3.12 查询结果

查询计算机系的学生以及年龄不大于 19 岁的学生。

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

	Sno	Sname	Ssex	Sage	Sdept	Scholarship
1	200215121	李勇	男	26	CS	否
2	200215122	刘晨	女	25	CS	否
3	200215123	王敏	女	18	MA	否

图 3.13 查询结果

查询既选修了课程 1 又选修了课程 2 的学生（交集运算）。

```
SELECT Sno FROM SC WHERE Cno='1' INTERSECT SELECT Sno FROM SC W
HERE Cno='2';
```

	Sno
1	200215121

图 3.14 查询结果

将信息系所有学生的年龄增加 1 岁。

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

删除学号为 95019 的学生记录。

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

插入一条选课记录('95020', '1 ')。

```
INSERT INTO SC (Sno,Cno) VALUES('95020','1');*/
```

3.2.2 扩展练习

查询每门课程及其被选情况（输出所有课程中每门课的课程号、课程名称、选修该课程的学生学号及成绩--如果没有学生选择该课，则相应的学生学号及成绩为空值）。

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

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

图 3.15 查询结果

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）

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

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

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

	Sno	Sname	Sage
1	200215125	张立	21

图 3.16 查询结果

查询选修了 3 号课程而且成绩为良好（80~89 分）的所有学生的学号和姓名。
 SELECT Student.Sno, Sname FROM Student, SC WHERE Student.Sno=SC.Sno AND SC.Cno='3' AND SC.Grade BETWEEN 80 AND 90;

	Sno	Sname
1	200215121	李勇
2	200215122	刘晨

图 3.17 查询结果

查询学生 200215122 选修的课程号、课程名
 SELECT SC.Cno,Cname FROM SC,Course WHERE SC.Cno=Course.Cno AND SC.Sno='200215122';

	Cno	Cname
1	2	数学
2	3	信息系统

图 3.18 查询结果

思考：如何查询学生 200215122 选修的课程号、课程名及成绩？
 SELECT SC.Cno, Cname, Grade FROM SC,Course WHERE SC.Sno='200215122' AND SC.Cno=Course.Cno;

	Cno	Cname	Grade
1	2	数学	90
2	3	信息系统	80

图 3.19 查询结果

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

Sno	Cno
-----	-----

图 3.20 查询结果

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');

	Sno	Sname	Sage
1	200215123	王敏	18

图 3.21 查询结果

查询所有选修了 2 号课程的学生姓名及所在系。

```
SELECT Sname,Sdept FROM Student,SC WHERE SC.Cno='2' AND Student.Sno=S
C.Sno;*/
```

	Sname	Sdept
1	李勇	CS
2	刘晨	CS

图 3.22 查询结果

使用 update 语句把成绩为良的学生的年龄增加 2 岁，并查询出来。

```
/*UPDATE Student SET Sage=Sage+2 WHERE Sno IN(SELECT Sno FROM SC W
HERE Grade BETWEEN 80 AND 90);
```

```
SELECT Student.Sno,Sage,Grade FROM Student,SC WHERE Student.Sno=SC.Sno
AND Grade BETWEEN 80 AND 90;
```

	Sno	Sage	Grade
1	200215121	24	85
2	200215121	24	88
3	200215122	23	90
4	200215122	23	80

图 3.23 查询结果

使用 insert 语句增加两门课程：C 语言和人工智能，并查询出来

```
INSERT INTO Course(Cno,Cname) VALUES('8','C 语言'),('9','人工智能');
```

```
SELECT * FROM Course WHERE Cname='C 语言' OR Cname='人工智能';
```

	Cno	Cname	Cpno	Ccredit
1	8	C语言	NULL	NULL
2	9	人工智能	NULL	NULL

图 3.24 查询结果

使用 delete 语句把人工智能课程删除，并查询出来。

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

```
SELECT Cname FROM Course;*/
```

	Cname	▼
1	数据库	...
2	数学	...
3	信息系统	...
4	操作系统	...
5	数据结构	...
6	数据处理	...
7	PASCAL 语言	...

图 3.25 查询结果

3.3 任务总结

对于嵌套查询的子查询，要在逻辑上先将所要查询目标的问题剥离开，一步一步的分析，这样才不会出错。

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
- (6) 写出实验报告

4.2 完成过程

- (1) 创建 CS 系的视图 CS_View

```
CREATE VIEW CS_VIEW AS (SELECT * FROM Student WHERE Sdept='CS');  
SELECT * FROM CS_VIEW;
```

	Sno	Sname	Ssex	Sage	Sdept	Scholarship
1	200215121	李勇	男	24	CS	否
2	200215122	刘晨	女	23	CS	否

图 4.1 查询结果

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

```
SELECT SC.Sno,CS_VIEW.Sname FROM SC, CS_VIEW WHERE SC.Cno='1' AND  
SC.Sno=CS_VIEW.Sno;
```

	Sno	Sname
1	200215121	李勇

图 4.2 查询结果

- (3) 创建 IS 系成绩大于 80 的学生的视图 IS_View

```
CREATE VIEW IS_VIEW
```

```
AS
```

```
(SELECT Student.Sno,Sname, Ssex, Sage, Sdept, Scholarship, Cno, Grade FROM Student, SC WHERE SC.Grade>80 AND SC.Sno=Student.Sno AND Sdept='IS');
```

- (4) 在视图 IS_View 查询 IS 系成绩大于 80 的学生

Sno	Sname	Ssex	Sage	Sdept	Scholarship	Cno	Grade

图 4.3 查询结果

为空视图

(5)删除视图 IS_View
DROP VIEW IS_VIEW;

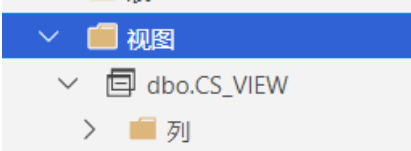


图 4.4 删除结果

(6) 利用可视化窗口创建 2 个不同的用户 U1 和 U2,利用系统管理员给 U1 授予 Student 表的 查询和更新的权限,给 U2 对 SC 表授予插入的权限。然后用 U1 登录,分别 1) 查询学生表 的信息;2) 把所有学生的年龄增加 1 岁,然后查询;3) 删除 IS 系的学生;4) 查询 CS 系 的选课信息。用 U2 登录,分别 1) 在 SC 表中插入 1 条记录 (‘200215122’ , ‘1’ , 75);2) 查询 SC 表的信息,3) 查询视图 CS_View 的信息。

创建 U1:

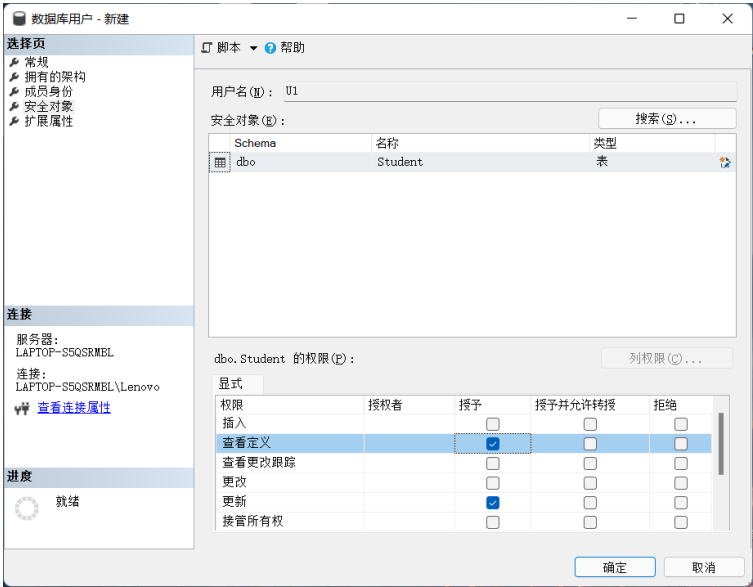


图 4.5 U1 用户权限定义

更新后查询：

	Sno	Sname	Ssex	Sage	Sdept	Scholarship
1	200215121	李勇	男	25	CS	否
2	200215122	刘晨	女	24	CS	否
3	200215123	王敏	女	19	MA	否
4	200215125	张立	男	22	IS	否

图 4.9 查询结果

登录 U2:

Connection Details

Connection type

Microsoft SQL Server

Server

LAPTOP-S5QSRMBL

Authentication type

SQL Login

User name

U2

Password

.....

☒ Remember password

Database

<Default>

服务器组

<Default>

图 4.10 用户登录

插入记录：

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

ges

F9:34:54

Started executing query at Line 1
(1 行受到影响)
Total execution time: 00:00:00.007

图 4.11 执行结果

查询 CS_VIEW 的结果：（提示没有权限）

```
SELECT * FROM CS_VIEW;
```

ges

```
午9:36:23      Started executing query at Line 1
Msg 229, Level 14, State 5, Line 29
拒绝了对象 'CS_VIEW' (数据库 'S_T_U201911741', 架构 'dbo')的 SELECT 权限。
Total execution time: 00:00:00.001
```

图 4.12 查询结果

(7) 用系统管理员登录，收回 U1 的所有权限

```
revoke select,update on student from U1;
```

ges

```
F9:41:37      Started executing query at Line 1
命令已成功完成。
Total execution time: 00:00:00.011
```

图 4.13 执行结果

(8) 用 U1 登录，查询学生表的信息

```
SELECT * FROM Student;
```

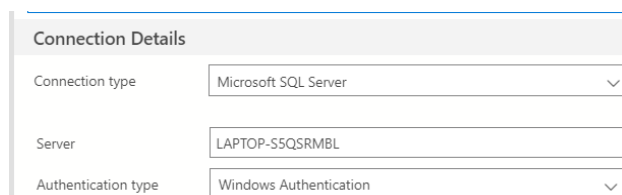
ges

```
F9:42:26      Started executing query at Line 1
Msg 229, Level 14, State 5, Line 33
拒绝了对象 'Student' (数据库 'S_T_U201911741', 架构 'dbo')的 SELECT 权限。
Total execution time: 00:00:00
```

图 4.14 执行结果

提示没有权限

(9) 用系统管理员登录



Connection Details	
Connection type	Microsoft SQL Server
Server	LAPTOP-S5QSRMBL
Authentication type	Windows Authentication

图 4.15 用户登录

以本机账户登录

(10) 对 SC 表建立一个更新触发器，当更新了 SC 表的成绩时，如果更新后的成绩大于等于 95，则检查该成绩的学生是否有奖学金，如果奖学金是“否”，则修改为“是”。如果修改后的成绩小于 95，则检查该学生的其他成绩是不是有大于 95 的，如果都没有，且修改前的成绩是大于 95 时，则把其奖学金修改为“否”。然后进行成绩修改，并进行验证是否触发器正确执行。1) 首先把某个学生成绩修改为 98，查询其奖学金。2) 再把刚才的成绩修改为 80，再查询其奖学金。

创建触发器：

GO

CREATE TRIGGER SC_T

ON SC FOR UPDATE

AS

DECLARE @Grade smallint;

SELECT @Grade = Grade FROM INSERTED;

BEGIN

IF(@Grade>=95)

BEGIN

UPDATE Student SET Scholarship = '是' WHERE Sno in (SELECT Sno FROM inserted);

SELECT * FROM Student;

SELECT * FROM inserted;

END

ELSE

BEGIN

UPDATE Student SET Scholarship = '否' WHERE NOT EXISTS
(SELECT * FROM SC WHERE Sno in (SELECT Sno FROM inserted) AND
Grade>=95) AND Sno in (SELECT Sno FROM inserted);

END

END;

修改，查询：

```
UPDATE SC SET Grade=95 WHERE Sno = '200215121' AND Cno = '1';
```

```
SELECT Sno, Sname, Scholarship FROM Student;
```

	Sno	Sname	Ssex	Sage	Sdept	Scholarship
1	200215121	李勇	男	25	CS	是
2	200215122	刘晨	女	24	CS	否
3	200215123	王敏	女	19	MA	否
4	200215125	张立	男	22	IS	否

	Sno	Cno	Grade
1	200215121	1	95

图 4.16 查询结果

(11) 删除刚定义的触发器

```
DROP TRIGGER SC_T;
```

(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。

定义存储过程：

```
GO
```

```
create procedure getCSAvgMax
```

```
as
```

```
begin
```

```
select sc.Sno,AVG(Grade) AVG,MAX(Grade) MAX from SC,Student where Student.  
Sno=SC.Sno and Sdept='CS' group by SC.Sno;
```

```
end;
```

使用 EXEC getCSAvgMax; 执行，

执行结果：

	Sno	AVG	MAX
1	200215121	89	95
2	200215122	81	90

图 4.17 查询结果

(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。

定义存储过程:

GO

CREATE PROCEDURE getCourseGrade

@Sno char(9)

AS

BEGIN

SELECT Sname, Cno, Grade FROM Student, SC WHERE Student.Sno=@Sno AND
SC.Sno=@Sno;

end;

验证: EXEC getCourseGrade @Sno='200215121';

	Sname	Cno	Grade
1	李勇	1	95
2	李勇	2	85
3	李勇	3	88

图 4.18 查询结果

(14) 把上一题改成函数。再进行验证。

定义表值函数:

GO

CREATE FUNCTION getGrade(@Sno char(9))

RETURNS @test table(sno char(9), name char(20), grade smallint)

AS

BEGIN

INSERT @test SELECT Sname, Cno, Grade FROM Student, SC WHERE
Student.Sno=@Sno AND SC.Sno=@Sno;

RETURN

end;

调用语句如下:

SELECT * from [dbo].getGrade('200215121');

	sno	name	grade
1	李勇	1	95
2	李勇	2	85
3	李勇	3	88

图 4.19 查询结果

(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。

修改查询：

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

SELECT * FROM SC WHERE Sno='200215121' AND Cno=1;

	Sno	Cno	Grade
1	200215121	1	120

图 4.20 查询结果

定义完整性约束：

ALTER TABLE SC ADD CONSTRAINT gradeRange CHECK(Grade BETWEEN 0 AND 100);

但是表中已有数据，会产生错误，故用可视化界面进行添加

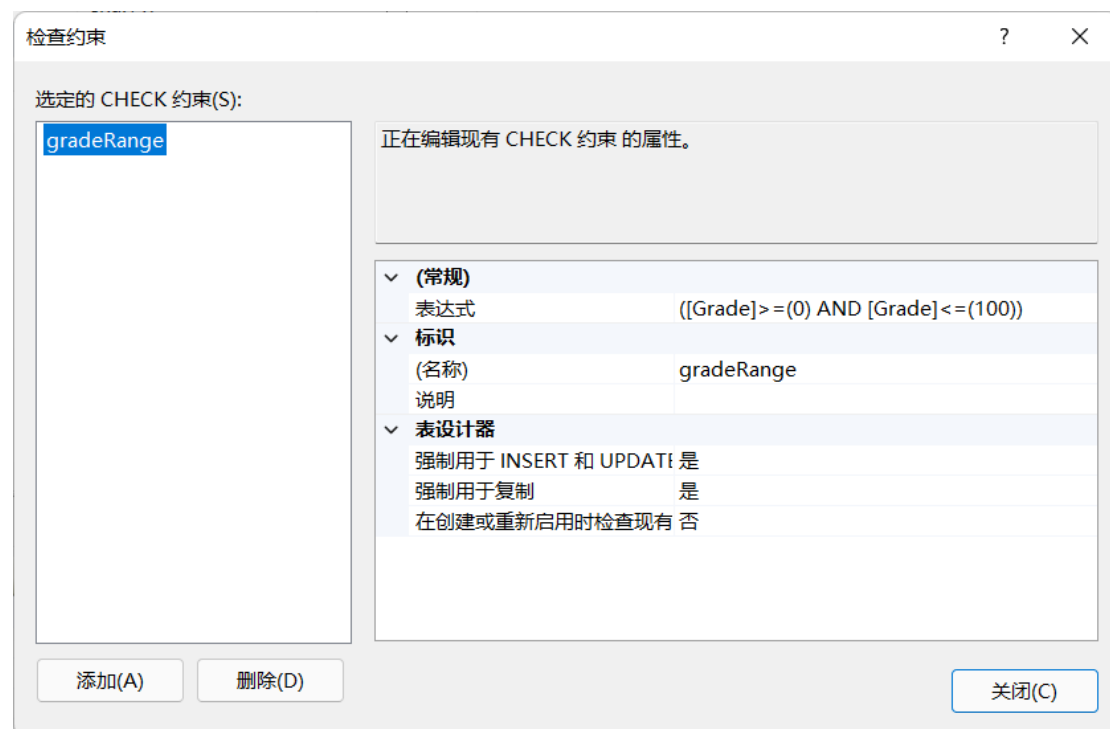


图 4.21 check 条件添加

再次修改时，会提示与完整性约束冲突：

```
Started executing query at Line 1
Msg 547, Level 16, State 0, Line 106
UPDATE 语句与 CHECK 约束"gradeRange"冲突。该冲突发生于数据库"S_T_U201911741"，表"dbo.SC"，column 'Grade'。
语句已终止。
(1 行受到影响)
```

图 4.22 执行结果

4.3 任务总结

sql server 的触发器，存储过程和函数的书写格式和 mysql 的语句格式相差很多，不能使用课堂上教的和书上写的那种格式写，还好微软的官方文档写的比较详细，经过查询，学习还是能写出来的。

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。

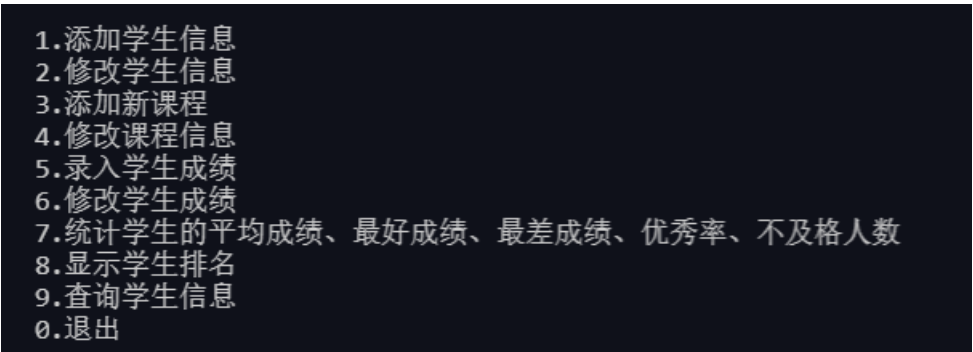
5.2 完成过程

5.2.1 系统概述

使用 python 的 pymssql 库，进行系统的编写。

大致的思路是将每个功能模块化，再通过菜单调用。

功能菜单如下图所示：



```
1.添加学生信息
2.修改学生信息
3.添加新课程
4.修改课程信息
5.录入学生成绩
6.修改学生成绩
7.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
8.显示学生排名
9.查询学生信息
0.退出
```

图 5.1 菜单

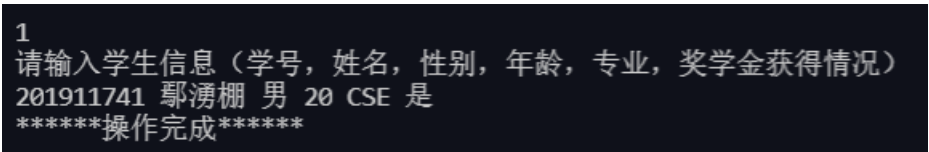
使用系统默认配置，连接到本机数据库，代码如下：

```
conDB = pymssql.connect(db_config['host'], db_config['user'], db_config['password'],
db_config['db'], db_config['charset'])
```

其中 db_config 是定义好的元组，为本机数据库的相关信息。

5.2.2 操作演示

添加学生信息：



```
1
请输入学生信息（学号，姓名，性别，年龄，专业，奖学金获得情况）
201911741 鄢湧棚 男 20 CSE 是
*****操作完成*****
```

图 5.2 操作结果

修改学生信息：

修改学号为 201911741 的学生的年龄为 21

```
2
请输入要修改的内容（0.学号 1.姓名 2.性别 3.年龄 4.专业 5.奖学金）（先输入序号，再输入内容）
3 21
请输入要修改的学生的学号或姓名
201911741
*****操作完成*****
```

图 5.3 操作结果

添加新课程：

添加编译原理课

```
3
请输入课程信息（课程号，课程名，先修课程（无则填NULL），学分）
10 编译原理 NULL 2
*****操作完成*****
```

图 5.4 操作结果

修改课程信息：

修改编译原理的学分为 3 分

```
4
请输入修改课程的课程号或课程名
编译原理
请输入要修改的内容（0.课程 1.课程名 2.先修课程 3.学分）
3 3
*****操作完成*****
```

图 5.5 操作结果

录入学生成绩：

```
5
请输入学生学号，课程号，成绩（输入0停止输入）
201911741 10 90
0
*****操作完成*****
```

图 5.6 操作结果

修改学生成绩：

```
6
请输入要修改的学生学号，课程号和成绩
201911741 10 85
*****操作完成*****
```

图 5.7 操作结果

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

```
7
CS
max_min_avg: 95 75 85
failed: 0
failed: 0
total: 1
excellent: 0
excellent rate: 0.0

MA
max_min_avg: None None None
failed: 0
total: 1
excellent: 0
excellent rate: 0.0

*****操作完成*****
```

图 5.8 操作结果

查询学生信息

```
9
请输入学生学号:201911741
基础信息:
鄢湧棚          男          21          CSE          否
选课信息
10 编译原理
```

图 5.9 操作结果

5.3 任务总结

熟悉了 python 对数据库的操作，不同数据库有对应的不同的库，这样编写的应用程序对于不熟悉数据库的用户来说很友好，同时也能保证数据库中的信息不被泄露。

6 课程总结

1. 学习掌握了 SQL server 数据库及其可视化客户端管理工具 Azure Data Studio 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。

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。

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。

附录

实验四源代码：

Config.py:

```
from os import execl
import pymysql
```

```
db_config = {
    'host': 'LAPTOP-S5QSRMBL',
    'user': 'root',
    'password': 'root',
    'db': 'S_T_U201911741',
    'charset': 'cp936',
    #'cursorclass': pymysql.cursors.DictCursor
}
Student = {
    '11741',
    '鄢湧棚',
    '男',
    '20',
    'cse',
    '是'
}
Student_list = (
    'Sno',
    'Sname',
    'Ssex',
    'Sage',
    'Sdept',
    'Scholarship'
)
Course_list = (
    'Cno',
    'Cname',
    'Cpno',
    'Ccredit'
)
```

```
def printc(str):
```

```

print(str.encode('latin-1').decode('gbk'), end="")

def Add_student_info(cursor,conDB):
    print ("请输入学生信息（学号，姓名，性别，年龄，专业，奖学金获得情况）")
    stu_in = input()
    Student = stu_in.split()
    #print (Student)
    sql_add = "insert into student values('" + Student[0] + "','" + \
        Student[1] + "','" + Student[2] + "','" + Student[3] \
        + "','" + Student[4] + "','" + Student[5] + "');"
    #print (sql_add)
    cursor.execute(sql_add)
    conDB.commit()
    return

def Update_student_info(cursor,conDB):
    print("请输入要修改的内容(0.学号 1.姓名 2.性别 3.年龄 4.专业 5.奖学金)")
    (先输入序号，再输入内容)
    flag = input()
    item = flag.split()
    item[0] = int(item[0])
    if(item[0] != 3):
        item[1] = "" + item[1] + ""
    #print (item)
    print("请输入要修改的学生的学号或姓名")
    opt = input()
    if(opt.isdigit()):
        sno = opt
        sql_update = "update student set " + Student_list[item[0]] + " = " + item[1]
        + " where Sno = '" + sno + "';"
    else:
        sname = opt
        sql_update = "update student set " + Student_list[item[0]] + " = " + item[1]
        + " where Sname = '" + sname + "';"
    #print (sql_update)
    cursor.execute(sql_update)
    conDB.commit()
    return

```

```

def Add_new_course(cursor,conDB):
    print("请输入课程信息（课程号，课程名，先修课程（无则填 NULL），学
分）")
    course_in = input()
    course = course_in.split()
    #print(course)
    sql_add = "insert into course values('" + course[0] + "', '" + course[1] + "', " +
course[2] + "," + course[3] + ");"
    #print (sql_add)
    cursor.execute(sql_add)
    conDB.commit()
    return

def Update_course_info(cursor,conDB):
    print("请输入修改课程的课程号或课程名")
    info_in = input()
    print("请输入要修改的内容（0.课程 1.课程名 2.先修课程 3.学分）")
    flag = input()
    item = flag.split()
    item[0] = int(item[0])
    if(item[0] != 3):
        item[1] = "" + item[1] + ""
    if(info_in.isdigit()):
        cno = info_in
        sql_update = "update course set " + Course_list[item[0]] + " = " + item[1] +
" where Cno = " + cno + ";"
    else:
        cname = info_in
        sql_update = "update course set " + Course_list[item[0]] + " = " + item[1] +
" where Cname = " + cname + ";"
    #print (sql_update)
    cursor.execute(sql_update)
    conDB.commit()
    return

def Delete_not_select_course(cursor,conDB):
    sql = "delete from course where Cno not in(select distinct Cno from SC);"
    cursor.execute(sql)
    conDB.commit()
    print("Cleanup complete.")

```

```

return

def Add_student_grades(cursor,conDB):
    print("请输入学生学号，课程号，成绩（输入 0 停止输入）")
    while(True):
        a = input()
        if (a == '0'):
            break
        SC = a.split()
        sql_add = "insert into SC values('" + SC[0] + "', '" + SC[1] + "', '" + SC[2] +
");"
        #print (sql_add)
        cursor.execute(sql_add)
        conDB.commit()
    return

def Update_student_grades(cursor, conDB):
    print("请输入要修改的学生学号，课程号和成绩")
    SC_in = input()
    SC = SC_in.split()
    sql_update = "update SC set grade = " + SC[2] + " where Sno = '" + SC[0] + "'
and Cno = '" + SC[1] + "';"
    #print(sql_update)
    cursor.execute(sql_update)
    conDB.commit()
    return

def Get_dept_statistics(cursor,conDB):
    sql1 = "select distinct Sdept from student;"
    cursor.execute(sql1)
    dept_get = cursor.fetchall()
    size = len(dept_get)
    for i in range (0,size):
        temp = str(dept_get[i])
        dept = temp[2:4]
        if(temp[4].isalpha()):
            dept = dept + temp[4]
        print(dept)
    #最大，最小，平均
    sql_max_min_avg = "SELECT MAX(Grade) MAX, MIN(Grade) MIN,

```



```
AVG(Grade) AVG FROM SC where Sno in (select Sno from student where Sdept =
"+dept+");"
```

```
cursor.execute(sql_max_min_avg)
```

```
max_min_avg = cursor.fetchall()
```

```
print("max_min_avg:",max_min_avg[0][0],max_min_avg[0][1],max_min_avg[0][2])
```

```
#不及格人数
```

```
sql_failed = "select count(*) from SC, Student where grade<60 and
Sc.Sno=Student.sno and Student.Sdept='"+dept+"';"
```

```
cursor.execute(sql_failed)
```

```
failed = cursor.fetchall()
```

```
print("failed:",failed[0][0])
```

```
#优秀率
```

```
sql2 = "select count(*) from student where Sdept = '"+dept+"';"
```

```
cursor.execute(sql2)
```

```
total = cursor.fetchall()
```

```
print("total:",total[0][0])
```

```
sql3 = "select count(*) from SC, Student where grade >= 80 and
SC.Sno=Student.Sno and Student.Sdept = '"+dept+"';"
```

```
cursor.execute(sql3)
```

```
excecelent = cursor.fetchall()
```

```
print("excecelent:",excecelent[0][0])
```

```
a = float(total[0][0])
```

```
b = float(excecelent[0][0])
```

```
print("excecelent rate:",b/a)
```

```
print("")
```

```
return
```

```
def Get_grade_order(cursor, conDB):
```

```
sql1 = "select distinct Sdept from student;"
```

```
cursor.execute(sql1)
```

```
dept_get = cursor.fetchall()
```

```
size = len(dept_get)
```

```
for i in range (0,size):
```

```
temp = str(dept_get[i])
```

```
dept = temp[2:4]
```

```
if(temp[4].isalpha()):
```

```
dept = dept + temp[4]
```

```
sql2 = "select Sname, Cname, grade from SC, student, Course where
SC.Sno = Student.Sno and SC.Cno = Course.Cno and Sdept = " + dept + " order by
```

```

grade desc;"
        cursor.execute(sql2)
        ret = cursor.fetchall()
        i = len(ret)
        print(dept)
        if(i != 0):
            for j in range(0,i):
                printc(ret[j][0])
                printc(ret[j][1])
                print(ret[j][2])
        else:
            print("没有学生成绩")
    return

def Get_stu_info(cursor,conDB):
    Sno = input("请输入学生学号:")
    sql1 = "select * from student where Sno = '"+Sno+"'";
    cursor.execute(sql1)
    base_info = cursor.fetchall()
    sql2 = "select SC.Cno, Cname from SC, Course where Sno = '"+Sno+"' and
SC.Cno = Course.Cno;"
    cursor.execute(sql2)
    course_info = cursor.fetchall()
    print("基础信息: ")
    printc(base_info[0][1])
    printc(base_info[0][2])
    print("\t\t",base_info[0][3],"\t\t",end="")
    printc(base_info[0][4])
    printc(base_info[0][5])
    print("")
    if(len(course_info) == 0):
        print("该学生没有选课")
    else:
        print("选课信息")
        print(course_info[0][0], end="")
        printc(course_info[0][1])
        print("\n")
    return

def Show_menu():

```

```
print("""  
功能菜单：(输入对应数字选择)  
1.添加学生信息  
2.修改学生信息  
3.添加新课程  
4.修改课程信息  
5.录入学生成绩  
6.修改学生成绩  
7.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  
8.显示学生排名  
9.查询学生信息  
0.退出  
""")
```

```

main.py:
import sys
from config import *
import pymssql
print(sys.getdefaultencoding())

conDB      =      pymssql.connect(db_config['host'],      db_config['user'],
db_config['password'], db_config['db'], db_config['charset'])

if conDB:
    print("连接成功")
    Show_menu()
    cursor = conDB.cursor()
    opt = input()
    while(opt != '0'):
        if (opt == '1'):
            Add_student_info(cursor,conDB)
        elif(opt == '2'):
            Update_student_info(cursor,conDB)
        elif(opt == '3'):
            Add_new_course(cursor,conDB)
        elif(opt == '4'):
            Update_course_info(cursor,conDB)
        elif(opt == '5'):
            Add_student_grades(cursor, conDB)
        elif(opt == '6'):
            Update_student_grades(cursor, conDB)
        elif(opt == '7'):
            Get_dept_statistics(cursor, conDB)
        elif(opt == '8'):
            Get_grade_order(cursor, conDB)
        elif(opt == '9'):
            Get_stu_info(cursor,conDB)
        elif(opt == '0'):
            break
    print("*****操作完成*****")
    Show_menu()
    opt = input()
else:
    print ("连接失败")

```

```
# 连接用完后记得关闭以释放资源  
conDB.close()
```