

長安大學

学生上机实验报告

实验课名称：《数据库原理与应用》课程上机实验

专业名称：软件工程

班级： 2021240205

学号： 2021902228

学生姓名： 严晏来

教师姓名： 李芳

___年___月___日

目录

实验一 数据库操作	5
1.1 数据库的建立和删除	5
1.1.1 建立数据库	5
1.1.2 删除数据库	5
1.1.3 建立学生一选课数据库，建立 SPJ 数据库	5
1.2 表格建立、修改、删除	5
1.2.1 建立表格：建立学生一选课数据库和 SPJ 数据库中对应的表格	5
1.2.2 修改表格	7
1.2.3 对表格进行删除	8
1.3 索引的建立与删除	8
1.3.1 索引建立	8
1.3.2 索引删除	8
1.4 总结	9
1.4.1 实验过程遇到的问题	9
1.4.2 解决方法	9
1.4.3 收获与体会	9
实验二 数据查询	10
2.1 单表查询	10
2.1.1 比较	10
2.1.2 范围	10
2.1.3 集合	11
查询既不是计算机科学系、数学系，也不是信息系的学生的姓名和 性别	11
2.1.4 聚集函数	11
2.1.5 匹配	12
2.1.6 空值	12
2.1.7 排序	13
2.1.8 分组	13

2.2 连接查询	14
2.2.1 自然	14
2.2.2 自身	14
2.2.3 外	15
2.2.4 嵌套	15
2.2.5 相关	16
2.2.6 eixsts	16
2.3 数据的插入、修改和删除	17
2.3.1 插入	17
2.3.2 修改	17
2.3.3 删除	17
2.4 视图	18
2.4.1 请为三建工程项目建立一个供应情况的视图，包括供应商代 码(SNO)、零件代码(PNO)、供应数量(QTY).	18
2.4.2 找出三建工程项目使用的各种零件代码及其数量	18
2.4.3 找出供应商 S1 的供应情况	18
2.5 总结	19
2.5.1 实验过程遇到的问题	19
2.5.2 解决方法	19
2.5.3 收获与体会	19
实验三 数据库安全性和完整性	20
3.1 授权与收回、角色	20
3.1.1 第五题	20
3.1.2 第六题	21
3.1.3 第七题	23
3.2 实体、参照完整性、用户定义完整性	23
3.3 触发器	25
实验四 数据库编程	29
4.1 过程	29

4.2 游标	35
--------------	----

实验一 数据库操作

目的：熟悉 RDBMS 基本操作，通过 RDBMS 进行各数据库对象的定义。

1.1 数据库的建立和删除

1.1.1 建立数据库

```
student_course> create database s1  
[2024-04-12 14:37:58] 17 ms 中有 1 行受到影响
```

1.1.2 删除数据库

```
student_course> drop database s1  
[2024-04-12 14:39:01] 在 24 ms 内完成
```

1.1.3 建立学生—选课数据库，建立 SPJ 数据库

```
create database student_course;
```

1.2 表格建立、修改、删除

1.2.1 建立表格：建立学生—选课数据库和 SPJ 数据库中对应的表格

```
1. use student_course;  
2. create table Student(  
3.     -> Sno int not null auto_increment primary key,  
4.     -> Sname varchar(10) not null,  
5.     -> Sex char(1) not null,  
6.     -> Sage tinyint(100) not null,  
7.     -> Sdept char(4) not null)comment = '学生表';
```

```

8.
9.  alter table Student auto_increment = 201215121;
10. insert into Student (Sname, Sex, Sage, Sdept) values ('李勇', '男', 20, 'CS');
11. insert into Student (Sname, Sex, Sage, Sdept) values ('刘晨', '女', 19, 'CS');
12. insert into Student (Sname, Sex, Sage, Sdept) values ('王敏', '女', 18, 'MA');
13. insert into Student (Sname, Sex, Sage, Sdept) values ('张立', '男', 19, 'IS');
14.
15. create table Course(
16.     ->Cno tinyint not null auto_increment primary key,
17.     ->Cname varchar(20) not null,
18.     ->Cpno tinyint,
19.     ->Ccredit tinyint not null)comment = '课程表';
20. alter table Course add foreign key(Cpno) references Course (Cno);
21.
22. insert into Course(Cname, Ccredit) values ('数据库', 4);
23. insert into Course(Cname, Ccredit) values ('数学', 2);
24. insert into Course(Cname, Ccredit) values ('信息系统', 4);
25. insert into Course(Cname, Ccredit) values ('操作系统', 3);
26. insert into Course(Cname, Ccredit) values ('数据结构', 2);
27. insert into Course(Cname, Ccredit) values ('PASCAL 语言', 4);
28. update Course set Cpno = 5 where Cno = 1;
29. update Course set Cpno = 1 where Cno = 3;
30. update Course set Cpno = 6 where Cno = 4;
31. update Course set Cpno = 2 where Cno = 6;
32.
33. create table SC(
34.     -> Sno int not null,
35.     -> Cno tinyint not null,
36.     -> Grade tinyint not null,
37.     -> primary key(Sno, Cno))comment = '学生课程表';
38. alter table SC add foreign key(Sno) references Student(Sno);
39. alter table SC add foreign key(Cno) references Course(Cno);
40.
41. insert into SC values(201215121, 1, 92);
42. insert into SC values(201215121, 2, 85);
43. insert into SC values(201215121, 3, 88);
44. insert into SC values(201215122, 2, 90);
45. insert into SC values(201215122, 3, 80);

```

	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	<null>	2
6	6	PASCAL语言	2	4

	Sno	Cno	Grade
1	201215121	1	92
2	201215121	2	85
3	201215121	3	88
4	201215122	2	90
5	201215122	3	80

	Sno	Sname	Sex	Sage	Sdept
1	201215121	李勇	男	20	CS
2	201215122	刘晨	女	19	CS
3	201215123	王敏	女	18	MA
4	201215124	张立	男	19	IS

1.2.2 修改表格

表中添加属性

```
ALTER TABLE student ADD S_entrance DATE;
```

	Sno	Sname	Sex	Sage	Sdept	S_entrance
1	201215121	李勇	男	20	CS	<null>
2	201215122	刘晨	女	19	CS	<null>
3	201215123	王敏	女	18	MA	<null>
4	201215124	张立	男	19	IS	<null>

更改表中的属性

```
ALTER TABLE student change Sage Sage int(10);
```

删除属性

```
ALTER TABLE student drop S_entrance;
```

	Sno	Sname	Sex	Sage	Sdept
1	201215121	李勇	男	20	CS
2	201215122	刘晨	女	19	CS
3	201215123	王敏	女	18	MA
4	201215124	张立	男	19	IS

1.2.3 对表格进行删除

限制

```
DROP TABLE student RESTRICT ;
```

级联

```
DROP TABLE student CASCADE ;
```

1.3 索引的建立与删除

1.3.1 索引建立

```
1 CREATE UNIQUE INDEX Stusno ON Student(Sno);
2 CREATE UNIQUE INDEX Coucno ON course(Cno);
3 ✓ CREATE UNIQUE INDEX SCno ON SC(Sno ASC,Cno DESC);
```

1.3.2 索引删除

```
DROP INDEX Stusno ON student;
```


1.4 总结

1.4.1 实验过程遇到的问题

在进行数据库操作的过程中，我遇到了一些挑战和问题。首先，对于表格的建立、修改和删除，我在一开始并没有清晰地理解每个步骤的含义以及操作方法，导致在进行这些操作时遇到了困难。特别是在修改表格时，对于添加、更改和删除属性的语法不是很熟悉，导致操作时出现了一些错误。

另外，索引的建立和删除也是一个较为复杂的部分。我发现自己在理解索引的作用和创建方法时存在一些模糊的地方，导致在实验中花费了较多的时间来理解和尝试。

1.4.2 解决方法

为了解决以上问题，我采取了一些措施。首先，我深入阅读了数据库操作的相关文档和教程，加强了对于每个操作步骤的理解。其次，我通过反复练习和尝试，逐渐熟悉了表格建立、修改和删除的操作方法，确保能够独立完成这些任务。对于索引的建立和删除，我也查阅了更多的资料，加深了对于索引的理解，并通过实际操作加以巩固。

1.4.3 收获与体会

通过这次实验，我不仅学会了如何在 RDBMS 中进行基本的数据库操作，还深刻体会到了细节对于操作的影响。在实验中遇到问题并解决问题的过程中，我逐渐提高了对于数据库操作的熟练度和自信心。同时，我也意识到在学习新知识和技能时，耐心和坚持是非常重要的，只有不断地实践和尝试，才能够真正掌握和运用所学知识。

实验二 数据查询

目的：掌握 SQL 查询语句的使用方法，加深对 SQL 和 SQL 语言的查询语句的理解。熟练掌握数据查询、数据排序和数据连接查询的操作方法。

2.1 单表查询

2.1.1 比较

查询所有年龄在 20 岁以下的学生姓名及年龄

```
SELECT Sname, Sage
from student
where Sage < 20;
```

	Sname	Sage
1	刘晨	19
2	王敏	18
3	张立	19

2.1.2 范围

查询年龄在 20-23 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

2.1.3 集合

查询既不是计算机科学系、数学系，也不是信息系的学生的姓名和性别

```
select Sname, Sex
from student
where Sdept not in ('CS', 'MA', 'IS');
```

0行

Sname	Sex
-------	-----

2.1.4 聚集函数

查询学生总人数

```
/*聚集函数*/
select count(*)
from student;
```

1	4
---	---

2.1.5 匹配

查询学号为 201215121 的学生的详细情况

```
select *
from student
where Sno like '201215121';
```

	Sno	Sname	Sex	Sage	Sdept
1	201215121	李勇	男	20	CS

2.1.6 空值

查询缺少成绩的学生的学号和相应的课程号

```
/*空值*/
select Sno,Cno
from SC
where Grade is null;
```

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

2.1.7 排序

查询全体学生情况，查询结果按所在系的系号升序排列，同一系中的学生按年龄降序排列。

```
select *  
from student  
order by Sage DESC;
```

	Sno	Sname	Sex	Sage	Sdept
1	201215121	李勇	男	20	CS
2	201215122	刘晨	女	19	CS
3	201215124	张立	男	19	IS
4	201215123	王敏	女	18	MA

2.1.8 分组

求各个课程号及相应的选课人数

```
/*分组*/  
select Cno, count(Sno)  
from sc  
group by Cno;
```

	Cno	`count(Sno)`
1	1	1
2	2	2
3	3	2

2.2 连接查询

2.2.1 自然

```
select student.sno, Sname, Sex, Sage, Sdept, Cno, Grade
from student, sc
where student.Sno=sc.Sno;
```

	sno	Sname	Sex	Sage	Sdept	Cno	Grade
1	201215121	李勇	男	20	CS	1	92
2	201215121	李勇	男	20	CS	2	85
3	201215121	李勇	男	20	CS	3	88
4	201215122	刘晨	女	19	CS	2	90
5	201215122	刘晨	女	19	CS	3	80

2.2.2 自身

```
/*自身*/
select first.cno, second.cjno
from course first, course second
where first.Cjno=second.Cno;
```

	cno	cjno
1	3	5
2	6	<null>
3	1	<null>
4	4	2

2.2.3 外

```
select student.sno, Sname, Sex, Sage, Sdept, Cno, Grade
from student left outer join sc s on student.Sno = s.Sno;
```

	sno	Sname	Sex	Sage	Sdept	Cno	Grade
1	201215121	李勇	男	20	CS	3	88
2	201215121	李勇	男	20	CS	2	85
3	201215121	李勇	男	20	CS	1	92
4	201215122	刘晨	女	19	CS	3	80
5	201215122	刘晨	女	19	CS	2	90
6	201215123	王敏	女	18	MA	<null>	<null>
7	201215124	张立	男	19	IS	<null>	<null>

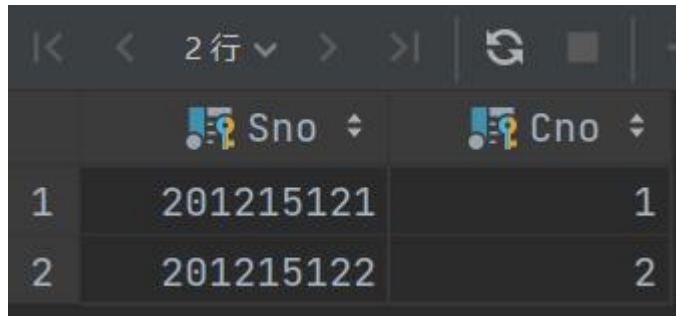
2.2.4 嵌套

```
select sno, Sname, Sdept
from student
where Sdept in
(
    select Sdept
    from student
    where Sname='刘晨'
);
```

	sno	Sname	Sdept
1	201215121	李勇	CS
2	201215122	刘晨	CS

2.2.5 相关

```
/*相关*/
select Sno,Cno
from sc x
where Grade >= (
    select avg(Grade)
    from sc y
    where y.Sno=x.Sno
);
```



The screenshot shows a database query result with two columns, Sno and Cno. The first row has Sno 201215121 and Cno 1. The second row has Sno 201215122 and Cno 2.

	Sno	Cno
1	201215121	1
2	201215122	2

2.2.6 exists

```
/*exist*/
select Sname
from student
where exists(
    select *
    from sc
    where sc.Sno=student.Sno and Cno='1'
);
```




A screenshot of a database client interface. At the top, there are navigation buttons and a dropdown menu showing '1 行' (1 row). Below this, a table is displayed with a single row. The first column is labeled 'Sname' and the value in the row is '李勇' (Li Yong).

Sname
李勇

2.3 数据的插入、修改和删除

2.3.1 插入

请将(S2, J6, P4, 200)插入供应情况关系。

```
spj> INSERT INTO SPJ (SNO, JNO, PNO, QTY) VALUES ('S2', 'J6', 'P4', 200)
[2024-04-12 15:12:24] 8 ms 中有 1 行受到影响
```

2.3.2 修改

把全部红色零件的颜色改成蓝色。

```
spj> UPDATE P
      SET COLOR = '蓝'
      WHERE COLOR = '红'
[2024-04-12 15:13:08] 8 ms 中有 3 行受到影响
```

2.3.3 删除

从供应商关系中删除供应商号是 S2 的记录，并从供应情况关系中删除相应的记录。

```
spj> DELETE FROM SPJ WHERE SNO = 'S2'
[2024-04-12 15:13:57] 8 ms 中有 7 行受到影响
```

2.4 视图

2.4.1 请为三建工程项目建立一个供应情况的视图，包括供应商代码(SNO)、零件代码(PNO)、供应数量(QTY).

```
/*请为三建工程项目建立一个供应情况的视图，包括供应商代码(SNO)、零件代码(PNO)、供应数量(QTY)*/  
CREATE VIEW VSP  
AS  
SELECT SNO,SPJ.PNO,QTY FROM SPJ,J  
WHERE SPJ.JNO=J.JNO AND J.JNAME='三建';
```

2.4.2 找出三建工程项目使用的各种零件代码及其数量

```
/*(1)找出三建工程项目使用的各种零件代码及其数量。*/  
SELECT DISTINCT PNO, QTY FROM VSP;
```

	PNO	QTY
1	P1	200
2	P3	200
3	P5	100

2.4.3 找出供应商 S1 的供应情况

```
/*(2)找出供应商S1的供应情况。*/  
SELECT DISTINCT * FROM VSP WHERE SNO='S1';
```

	SNO	PNO	QTY
1	S1	P1	200

2.5 总结

2.5.1 实验过程遇到的问题

在进行数据库查询实验的过程中，我遇到了一些问题。首先，对于复杂查询语句的理解和构建存在一定困难，特别是在涉及到连接查询和子查询的情况下。这些查询语句的语法结构和逻辑需要我花费更多的时间来理解和掌握，有时候容易在语法上出现错误。

其次，对于数据的插入、修改和删除操作，我在操作时可能会因为不熟悉语法而出现错误，导致操作失败或者数据不一致的情况。

2.5.2 解决方法

为了解决以上问题，我采取了一些措施。首先，我加强了对于 SQL 语法和查询语句的学习，通过查阅资料和实践练习，逐步提高了自己的理解能力和操作技能。特别是针对连接查询和子查询，我多次尝试并反复练习，逐渐加深了对于这些复杂查询语句的理解。

其次，我在进行数据操作时更加谨慎，仔细阅读文档并反复确认语法，避免因疏忽导致操作错误。同时，我也通过查询错误信息和参考文档等方式，及时纠正了操作中的错误，确保了实验的顺利进行。

2.5.3 收获与体会

通过这次数据库查询实验，我不仅学会了如何使用 SQL 语言进行数据查询和操作，还提高了自己的问题解决能力和学习能力。在实验中遇到问题并解决问题的过程中，我逐渐掌握了解决问题的方法和技巧，也增强了自己的学习动力和信心。

实验三 数据库安全性和完整性

目的：分析数据库完整性需满足的条件，掌握数据库完整性的定义方式，完成相应的完整性检验。并掌握 SQL Server 中有关用户，角色及操作权限的管理方法。

3.1 授权与收回、角色

3.1.1 第五题

题目：

对下列两个关系模式:

学生(学号,姓名,年龄 ,性别 ,家庭住址 ,班级号)

班级(班级号 ,班级名 ,班主任,班长)

请用 SQL 的 GRANT 语句完成下列授权功能：

- ① 授予用户 U1 拥有对两个表的所有权限 ,并可给其他用户授权

```
student_course> GRANT ALL PRIVILEGES
                  ON TABLE Student
                  TO U1
                  WITH GRANT OPTION
[2024-05-15 13:54:29] 在 8 ms 内完成
```

- ② 授予用户 U2 对学生表具有查看权限 ,对家庭住址具有更新权限

```
student_course> GRANT SELECT,UPDATE(Se
                  ON TABLE Student
                  TO U2
[2024-05-15 13:58:22] 在 6 ms 内完成
```

- ③ 将对"班级"表查看权限授予所有用户。
④ 将对学生表的查询 ,更新权限授予角色 R1。

⑤ 将角色 R1 授予用户 U1,并且 U1 可继续授权给其他角色。

解答:

```
1. 1.
2. GRANT ALL PRIVILEGES
3. ON TABLE Student,Class
4. TO U1
5. WITH GRANT OPTION;
6.
7. 2.
8. GRANT SELECT,UPDATE('家庭住址')
9. ON TABLE Student
10. TO U2;
11.
12. 3.
13. GRANT SELECT
14. ON TABLE Class
15. TO PUBLIC;
16.
17. 4.
18. GRANT SELECT.UPDATE
19. ON TABLE Student
20. TO R1;
21.
22. 5.
23. GRANT R1 TO U1
24. WITH GRANT OPTION;
```

3.1.2 第六题

今有两个关系模式:

职工(职工号 ,姓名 ,年龄 ,职务 ,工资 ,部门号)

部门(部门号,名称,经理名,地址,电话号)

请用 SQL 的 GRANT 和 REVOKE 语句(加上视图机制)完成以下授权定义或存取控制功能:

① 用户王明对两个表有 SELECT 权限。

```
1. GRANT INSERT
2. ON TABLE EMP,DEPT
3. TO '王明';
```

② 用户李勇对两个表有 INSERT 和 DELETE 权限。

```
1. GRANT INSERT,DELETE
2. ON TABLE EMP,DEPT
3. TO '李勇';
```

③ 每个职工只对自己的记录有 SELECT 权限;

```
1. GRANT SELECT
2. ON TABLE EMP
3. WHEN USER()=NAME
4. TO ALL;
```

④ 用户刘星对职工表有 SELECT 权限 ,对工资字段具有更新权限。

```
1. GRANT SELECT,UPDATE('工资')
2. ON TABLE EMP
3. TO '刘星'
```

⑤ 用户张新具有修改这两个表的结构权限。

```
1. GRANT ALTER TABLE
2. ON TABLE EMP,DEPT
3. TO '张新'
```

⑥ 用户周平具有对两个表所有权限(读,插,改,删数据),并具有给其他用户授权的权限。

```
1. GRANT ALL PRIVILIGES ON 职工,部门
2. ON TABLE EMP,DEPT
3. TO '周平'
4. WITH GRANT OPTION;
```

⑦ 用户杨兰具有从每个部门职工中 SELECT 最高工资.最低工资 .平均工资的权限,他 不能查看每个人的工资。

首先建立一个视图。然后对这个视图定义杨兰的存取权限。

```
1. CREATE VIEW 部门工资 AS
2. SELECT DEPT.名称,MAX(工资) ,MIN(工资) ,AVG(工资)
3. FROM EMP ,DEPT
4. WHERE EMP.部门号=DEPT.部门号
5. GROUP BY EMP.部门号;
6.
```

```
7. GRANT SELECT ON TABLE 部门工资 TO '杨兰';
```

3.1.3 第七题

针对第 6 题中 1-7 的每一种情况，撤销各用户所授予的权限。

```
1. 1.REVOKE SELECT ON TABLE EMP,DEPT FROM '王明';
2.
3. 2.REVOKE INSERT,DELETE ON TABLE EMP,DEPT FROM '李勇';
4.
5. 3.REVOKE SELECT ON TABLE EMP WHEN USER()= NAME FROM ALL;
6.
7. 4.REVOKE SELECT,UPDATE ON TABLE EMP FROM '刘星';
8.
9. 5.REVOKE ALTER TABLE ON TABLE EMP,DEPT FROM '张新';
10.
11. 6.REVOKE ALL PRIVILEGES ON TABLE EMP,DEPT FROM '周平';
12.
13. 7.REVOKE SELECT ON TABIE 部门工资 FROM '杨兰';
14. DROP VIEW 部门工资;
```

3.2 实体、参照完整性、用户定义完整性

3.2.1 第六题

假设有下面两个关系模式：

职工（职工号，姓名，年龄，职务，工资，部门号），其中职工号为主码；

部门（部门号，名称，经理姓名，电话），其中部门号为主码。

用 SQL 语言定义这两个关系模式，要求在模式中完成以下完整性约束条件的定义：

- （1）定义每个模式的主码
- （2）定义参照完整性约束
- （3）定义职工年龄不超过 65 岁

```
1. -- 定义职工关系模式
2. CREATE TABLE 职工 (
3.     职工号 INT PRIMARY KEY,
4.     姓名 VARCHAR(50),
```

```

5.      年龄 INT,
6.      职务 VARCHAR(50),
7.      工资 DECIMAL(10, 2),
8.      部门号 INT,
9.      CONSTRAINT FK_部门号 FOREIGN KEY (部门号) REFERENCES 部门(部门号),
10.     CONSTRAINT CK_年龄 CHECK (年龄 <= 65)
11. );
12. -- 定义部门关系模式
13. CREATE TABLE 部门 (
14.     部门号 INT PRIMARY KEY,
15.     名称 VARCHAR(50),
16.     经理姓名 VARCHAR(50),
17.     电话 VARCHAR(20)
18. );

```

解释：

(1) 主码定义：

职工关系模式的主码为职工号（PRIMARY KEY）。

部门关系模式的主码为部门号（PRIMARY KEY）。

(2) 参照完整性约束：

在职工关系模式中，部门号列设有外键约束（FOREIGN KEY），参照了部门关系模式中的部门号。

(3) 年龄约束：

在职工关系模式中，通过 CHECK 约束条件（CK_年龄），限制了年龄不超过 65 岁。

3.2.2 题目自设

（1）建立考生信息表，要求考号需在 000000-999999 之间，姓名不能为空值，年龄需大于 18，报考学校不能多余 3 个。

```

1.  CREATE TABLE 考生信息 (
2.      考号 INT PRIMARY KEY,
3.      姓名 VARCHAR(50) NOT NULL,
4.      年龄 INT CHECK (年龄 > 18),
5.      报考学校 1 INT,
6.      报考学校 2 INT,
7.      报考学校 3 INT,
8.      FOREIGN KEY (报考学校 1, 报考学校 2, 报考学校 3) REFERENCES 报考学校(学
校编号)

```



```
9. );
```

(2) 建立报考学校信息表，要求每个学校招生名额不得多于 100 个。

```
1. CREATE TABLE 报考学校 (  
2.     学校编号 INT PRIMARY KEY,  
3.     学校名称 VARCHAR(100),  
4.     招生名额 INT CHECK (招生名额 <= 100)  
5. );
```

(3) 删除考生信息表中对考生年龄的限制（即无论多少岁都能报考）。

```
1. ALTER TABLE 考生信息  
2. DROP CONSTRAINT 考生信息_年龄_check;
```

(4) 修改考生信息表的约束条件，要求报考学校不能多于 2 个。

```
1. ALTER TABLE 考生信息  
2. DROP CONSTRAINT 考生信息_报考学校_3_check;  
3. ALTER TABLE 考生信息  
4. ADD CHECK (  
5.     (报考学校 1 IS NULL AND 报考学校 2 IS NULL) OR  
6.     (报考学校 1 IS NOT NULL AND 报考学校 2 IS NULL AND 报考学校  
7.     3 IS NULL) OR  
8.     (报考学校 1 IS NOT NULL AND 报考学校 2 IS NOT NULL AND 报考学校  
9.     3 IS NULL)  
10. );
```

3.3 触发器

3.3.1 例 5.21

当对 SC 表 Grade 属性进行修改时，将此操作记录到另一个表 SC_U (Sno,Cno,Oldgrade,Newgrade)。

```
1. USE student_course;  
2. CREATE TABLE SC_U (  
3.     Sno CHAR(9),  
4.     Cno INT,  
5.     Oldgrade INT,  
6.     Newgrade INT  
7. );  
8. CREATE TRIGGER update_grade_trigger  
9. AFTER UPDATE ON SC  
10. FOR EACH ROW  
11. BEGIN  
12.     IF OLD.Grade != NEW.Grade THEN
```

```

13.      INSERT INTO SC_U (Sno, Cno, Oldgrade, Newgrade)
14.      VALUES (OLD.Sno, OLD.Cno, OLD.Grade, NEW.Grade);
15.  END IF;
16. END;

```

结果:

```

student_course> UPDATE sc
                  SET Grade = 95
                  WHERE Sno = '201215121' AND Cno = 1
[2024-05-14 20:19:15] 22 ms 中有 1 行受到影响
student_course> SELECT * FROM SC_U
[2024-05-14 20:20:53] 在 198 ms (execution: 7 ms, fetching: 191 ms) 内检索到从 1 开始的 1 行

```

	Sno	Cno	Oldgrade	Newgrade
1	201215121	1	92	95

3.3.2 例 5.22

将每次对表 `student` 的插入操作所增加的学生个数记录到表 `Student-InsertLog` 中。

```

1.  CREATE TABLE `Student-InsertLog` (
2.      InsertTime TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
3.      InsertedStudentsCount INT
4.  );
5.
6.  drop trigger student_insert_trigger;
7.  DELIMITER //
8.
9.  CREATE TRIGGER student_insert_trigger
10. AFTER INSERT ON student
11. FOR EACH ROW
12. BEGIN
13.     -- 假设每次插入只增加一个学生
14.     SET @inserted_count = 1;
15.
16.     -- 插入新的记录到 Student-InsertLog 表中, 假设 Student-InsertLog 有一个名为 InsertTime 的字段来记录时间
17.     INSERT INTO `Student-InsertLog` (InsertedStudentsCount, InsertTime)
18.     VALUES (@inserted_count, NOW());
19. END;
20. DELIMITER ;

```

结果:

```
student_course> INSERT INTO student (Sno, Sname, Sex, Sage, Sdept)
VALUES ('201215127', '赵一', '男', 20, 'CS')
[2024-05-15 09:57:20] 11 ms 中有 1 行受到影响
student_course> SELECT * FROM `Student-InsertLog`
[2024-05-15 09:57:20] 在 32 ms (execution: 18 ms, fetching: 14 ms) 内检索到从 1 开始的 3 行
```

	InsertTime	InsertedStudentsCount
1	2024-05-15 09:57:20	1

3.3.3 例 5.23

定义一个 BEFORE 行级触发器，为 course 表定义完整性规则“学分 Ccredit 不得高于 5 分，如果高于 5 分，自动改为 5 分”

```
1. CREATE TRIGGER check_credit_trigger
2. BEFORE INSERT ON course
3. FOR EACH ROW
4. BEGIN
5.     IF NEW.Ccredit > 5 THEN
6.         SET NEW.Ccredit = 5;
7.     END IF;
8. END;
```

结果:

```
student_course> INSERT INTO course (Cno, Cname, Cpno, Ccredit)
VALUES (7, '高级数据库', NULL, 6)
[2024-05-14 20:32:31] 12 ms 中有 1 行受到影响
student_course> SELECT * FROM course WHERE Cno = 7
[2024-05-14 20:32:34] 在 60 ms (execution: 4 ms, fetching: 56 ms) 内检索到从 1 开始的 1 行
```

	Cno	Cname	Cpno	Ccredit
1	7	高级数据库	<null>	5

3.5 总结

3.5.1 实验过程遇到的问题

书上的都是 SQL server 语法，而本人使用的 MySQL，语法不一致，需要根据 MySQL 语法调整。

一开始不会设置用户。

3.5.2 解决方法

查资料，上网搜索，用 MySQL 语法解决问题。

模拟用户，并授予他权限。

3.5.3 收获与体会

实验四 数据库编程

目的：掌握用户存储过程及触发器的创建、执行、修改及删除方式，理解触发器与数据库定义约束的不同。

4.1 过程

4.1.1 例 8.5

给定学生学号，计算学生的平均绩点。

```
1.  DELIMITER //
2.
3.  CREATE PROCEDURE compGPA(
4.      IN inSno CHAR(10),    -- 输入参数：学生学号
5.      OUT outGPA FLOAT      -- 输出参数：平均学分绩点
6.  )
7.  BEGIN
8.      DECLARE courseGPA FLOAT; -- 课程绩点
9.      DECLARE totalGPA FLOAT;  -- 总绩点
10.     DECLARE totalCredit INT;  -- 总学分
11.     DECLARE grade INT;        -- 学生成绩
12.     DECLARE credit INT;       -- 课程学分
13.
14.     DECLARE done INT DEFAULT 0; -- 用于循环的标志变量
15.     DECLARE mycursor CURSOR FOR
16.         SELECT Course.Ccredit, SC.grade
17.         FROM SC
18.         INNER JOIN Course ON SC.Cno = Course.Cno
19.         WHERE SC.Sno = inSno; -- 查询指定学生的课程学分和成绩
20.
21.     DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1; -- 当游标到达末尾时设置
done 标志
22.
23.     SET totalGPA = 0; -- 初始化总绩点
24.     SET totalCredit = 0; -- 初始化总学分
25.
26.     OPEN mycursor; -- 打开游标
27.
28.     GPA_loop: LOOP
```

```

29.      FETCH mycursor INTO credit, grade; -- 从游标中获取数据
30.      IF done THEN
31.          LEAVE GPA_loop;    -- 如果游标到达末尾，跳出循环
32.      END IF;
33.
34.      IF grade BETWEEN 90 AND 100 THEN
35.          SET courseGPA := 4.0;
36.      ELSEIF grade BETWEEN 80 AND 89 THEN
37.          SET courseGPA := 3.0;
38.      ELSEIF grade BETWEEN 70 AND 72 THEN
39.          SET courseGPA := 2.0;
40.      ELSEIF grade BETWEEN 60 AND 69 THEN
41.          SET courseGPA := 1.0;
42.      ELSE
43.          SET courseGPA := 0;
44.      END IF;
45.
46.      SET totalGPA := totalGPA + courseGPA * credit; -- 计算总绩点
47.      SET totalCredit := totalCredit + credit;      -- 计算总学分
48.  END LOOP;
49.
50.  CLOSE mycursor; -- 关闭游标
51.
52.  IF totalCredit > 0 THEN
53.      SET outGPA := totalGPA / totalCredit; -- 计算平均学分绩点
54.  ELSE
55.      SET outGPA := 0; -- 如果总学分为0，则平均绩点为0
56.  END IF;
57. END //
58.
59. DELIMITER ;

```

结果:

```

student_course> SET @student_id = '201215121'
[2024-05-15 00:09:07] 在 3 ms 内完成
student_course> CALL compGPA(@student_id, @outGPA)
[2024-05-15 00:09:07] 在 2 ms 内完成
student_course> SELECT @outGPA
[2024-05-15 00:09:08] 在 83 ms (execution: 5 ms, fetching: 78 ms) 内检索到从 1 开始的 1 行

```

@outGPA	
1	3.4000000953674316

4.1.2P270 第 2 题

对“学生选课”数据库编写存储过程，完成下述功能：

①统计“离散数学”课程的成绩分布情况，即按照各分数段统计人数。

```
1.  create table Lisan(  
2.      Score_lisan char(20),  
3.      Count_lisan int );  
4.  
5.  
6.  insert into Course values('8','离散数学',NULL,4)  
7.  
8.  insert into SC values('201215121','8',45);  
9.  insert into SC values('201215122','8',65);  
10. insert into SC values('201215123','8',78);  
11. insert into SC values('201215125','8',82);  
12.  
13. select * from Course ;  
14. select * from SC where Cno='8';  
15.  
16. -- 创建存储过程来计算成绩分布  
17. DELIMITER //  
18. CREATE PROCEDURE Grade_add()  
19. BEGIN  
20.     DECLARE less60 INT DEFAULT 0;  
21.     DECLARE sixty_to_seventy INT DEFAULT 0;  
22.     DECLARE seventy_to_eighty INT DEFAULT 0;  
23.     DECLARE eighty_to_ninety INT DEFAULT 0;  
24.     DECLARE ninety_to_hundred INT DEFAULT 0;  
25.  
26.     SELECT COUNT(*) INTO less60  
27.     FROM SC  
28.     WHERE sc.Grade <= 60 AND Cno = 8;  
29.  
30.     SELECT COUNT(*) INTO sixty_to_seventy  
31.     FROM SC  
32.     WHERE sc.Grade > 60 AND sc.Grade <= 70 AND Cno = 8;  
33.  
34.     SELECT COUNT(*) INTO seventy_to_eighty  
35.     FROM SC  
36.     WHERE sc.Grade > 70 AND sc.Grade <= 80 AND Cno = 8;  
37.  
38.     SELECT COUNT(*) INTO eighty_to_ninety
```

```

39.     FROM SC
40.     WHERE sc.Grade > 80 AND sc.Grade <= 90 AND Cno = 8;
41.
42.     SELECT COUNT(*) INTO ninety_to_hundred
43.     FROM SC
44.     WHERE sc.Grade > 90 AND sc.Grade <= 100 AND Cno = 8;
45.
46.     -- 插入到Lisan 表中
47.     INSERT INTO Lisan VALUES('<=60', less60);
48.     INSERT INTO Lisan VALUES('60~70', sixty_to_seventy);
49.     INSERT INTO Lisan VALUES('70~80', seventy_to_eighty);
50.     INSERT INTO Lisan VALUES('80~90', eighty_to_ninety);
51.     INSERT INTO Lisan VALUES('90~100', ninety_to_hundred);
52. END //
53. DELIMITER ;

```

结果:

```

student_course> CALL Grade_add()
[2024-05-15 09:17:58] 16 ms 中有 1 行受到影响
student_course> SELECT * FROM Lisan
[2024-05-15 09:17:58] 在 57 ms (execution: 9 ms, fetching: 48 ms) 内检索到从 1 开始的 5 行

```

	Score_lisan	Count_lisan
1	<=60	1
2	60~70	1
3	70~80	1
4	80~90	1
5	90~100	0

②统计任意一门课程的平均成绩。

```

1.     DROP TABLE IF EXISTS AveSC;
2.
3.     CREATE TABLE AveSC(
4.         Cno tinyint(4),    -- 课程号
5.         CNAME CHAR(40),   -- 课程名
6.         AvgScore FLOAT,   -- 平均分
7.         FOREIGN KEY(Cno) REFERENCES Course(Cno)
8.     );
9.
10.    -- 插入初始数据, 假设所有课程的平均分初始为0
11.    INSERT INTO AveSC (Cno, CNAME, AvgScore)
12.    SELECT Cno, Cname, 0

```



```

13. FROM Course;
14.
15. SELECT * FROM AveSC;
16.
17. -- 创建存储过程来计算并更新课程的平均分
18. DELIMITER //
19. CREATE PROCEDURE AvgCourse()
20. BEGIN
21.     DECLARE done INT DEFAULT FALSE;
22.     DECLARE cno_var CHAR(4);
23.     DECLARE avg_score FLOAT;
24.     DECLARE cur CURSOR FOR
25.         SELECT Cno FROM Course;
26.     DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;
27.
28.     OPEN cur;
29.
30.     read_loop: LOOP
31.         FETCH cur INTO cno_var;
32.         IF done THEN
33.             LEAVE read_loop;
34.         END IF;
35.
36.         -- 计算每门课程的平均分
37.         SELECT AVG(Grade) INTO avg_score
38.         FROM SC
39.         WHERE Cno = cno_var;
40.
41.         -- 更新AveSC 表中的平均分
42.         UPDATE AveSC
43.         SET AvgScore = avg_score
44.         WHERE Cno = cno_var;
45.     END LOOP;
46.
47.     CLOSE cur;
48. END //
49. DELIMITER ;

```

```

student_course> CALL AvgCourse()
[2024-05-15 09:25:26] 在 32 ms 内完成
student_course> SELECT * FROM AveSC
[2024-05-15 09:25:26] 在 33 ms (execution: 0 ms, fetching: 33 ms) 内检索到从 1 开始的 8 行

```

	Cno	CNAME	AvgScore
1	1	数据库	95
2	2	数学	87.5
3	3	信息系统	84
4	4	操作系统	<null>
5	5	数据结构	<null>
6	6	PASCAL语言	<null>
7	7	高级数据库	<null>
8	8	离散数学	67.5

③将学生选课成绩从百分制改为等级制(即 A、B、C、D、E)。

```

1.  -- 在SC表中添加Score_level列
2.  ALTER TABLE SC ADD Score_level CHAR(4);
3.
4.  -- 删除存储过程（如果存在）
5.  DROP PROCEDURE IF EXISTS Createlevel;
6.
7.  -- 创建存储过程
8.  DELIMITER //
9.  CREATE PROCEDURE Createlevel()
10. BEGIN
11.    -- 更新Score_level的值
12.    UPDATE SC SET Score_level = 'E' WHERE Grade < 60;
13.    UPDATE SC SET Score_level = 'D' WHERE Grade >= 60 AND Grade < 70;
14.    UPDATE SC SET Score_level = 'C' WHERE Grade >= 70 AND Grade < 80;
15.    UPDATE SC SET Score_level = 'B' WHERE Grade >= 80 AND Grade < 90;
16.    UPDATE SC SET Score_level = 'A' WHERE Grade >= 90;
17. END //
18. DELIMITER ;

```

结果:

```

student_course> CALL Createlevel()
[2024-05-15 09:29:04] 16 ms 中有 2 行受到影响
student_course> SELECT * FROM SC
[2024-05-15 09:29:04] 在 63 ms (execution: 0 ms, fetching: 63 ms) 内检索到从 1 开始的 9 行

```

	Sno	Cno	Grade	Score_level
1	201215121	1	95	A
2	201215121	2	85	B
3	201215121	3	88	B
4	201215121	8	45	E
5	201215122	2	90	A
6	201215122	3	80	B
7	201215122	8	65	D
8	201215123	8	78	C
9	201215125	8	82	B

4.2 游标

建立一个过程，用游标获取指定的记录，并将取出的记录输出。

4.2.1 例 8.4

```

1. DROP PROCEDURE IF EXISTS GetStudentCourses;
2. DELIMITER //
3.
4. CREATE PROCEDURE GetStudentCourses(IN student_sno CHAR(10))
5. BEGIN
6.     DECLARE done INT DEFAULT FALSE;
7.     DECLARE CnoOfStudent CHAR(10);
8.     DECLARE GradeOfStudent INT;
9.     DECLARE cur CURSOR FOR
10.         SELECT Cno, Grade FROM SC WHERE Sno = student_sno;
11.     DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;
12.
13.     OPEN cur;
14.
15.     read_loop: LOOP
16.         FETCH cur INTO CnoOfStudent, GradeOfStudent;
17.         IF done THEN
18.             LEAVE read_loop;
19.         END IF;
20.
21.         -- 在MySQL 中，你不能直接“打印”或“通知”结果，但你可以通过 SELECT 语句
返回结果

```

```

22.          -- 或者在应用程序中捕获结果并显示
23.          SELECT CONCAT('Sno: ', student_sno, ', Cno: ', CnoOfStudent, ',
Grade: ', GradeOfStudent);
24.          END LOOP;
25.
26.          CLOSE cur;
27.      END //
28.
29. DELIMITER ;

```

结果:

```

student_course> CALL GetStudentCourses('201215121')
[2024-05-15 09:49:04] 在 630 ms (execution: 41 ms, fetching: 589 ms) 内检索到从 1 开始的 1 行

  CONCAT('Sno: ', student_sno, ', Cno: ', CnoOfStudent, ', Grade: ', GradeOfStudent)
1 Sno: 201215121, Cno: 1, Grade: 95

  CONCAT('Sno: ', student_sno, ', Cno: ', CnoOfStudent, ', Grade: ', GradeOfStudent)
1 Sno: 201215121, Cno: 2, Grade: 85

  CONCAT('Sno: ', student_sno, ', Cno: ', CnoOfStudent, ', Grade: ', GradeOfStudent)
1 Sno: 201215121, Cno: 3, Grade: 88

  CONCAT('Sno: ', student_sno, ', Cno: ', CnoOfStudent, ', Grade: ', GradeOfStudent)
1 Sno: 201215121, Cno: 7, Grade: 45

```