《数据库》课程实践报告

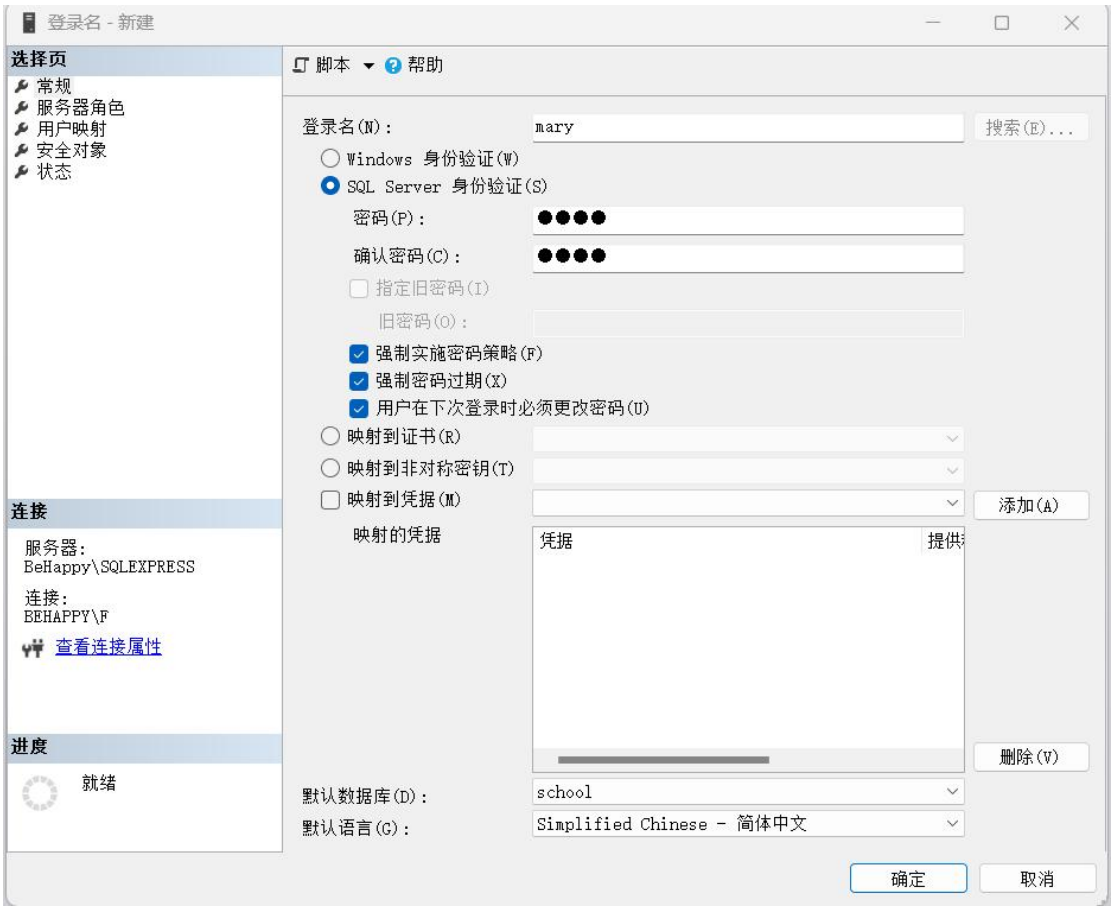
院、系	计算机学院	年级专业	21 计科	姓名	方浩楠	学号	2127405048
实验布置日期		提交日期				成绩	

实验十一 安全性控制实验

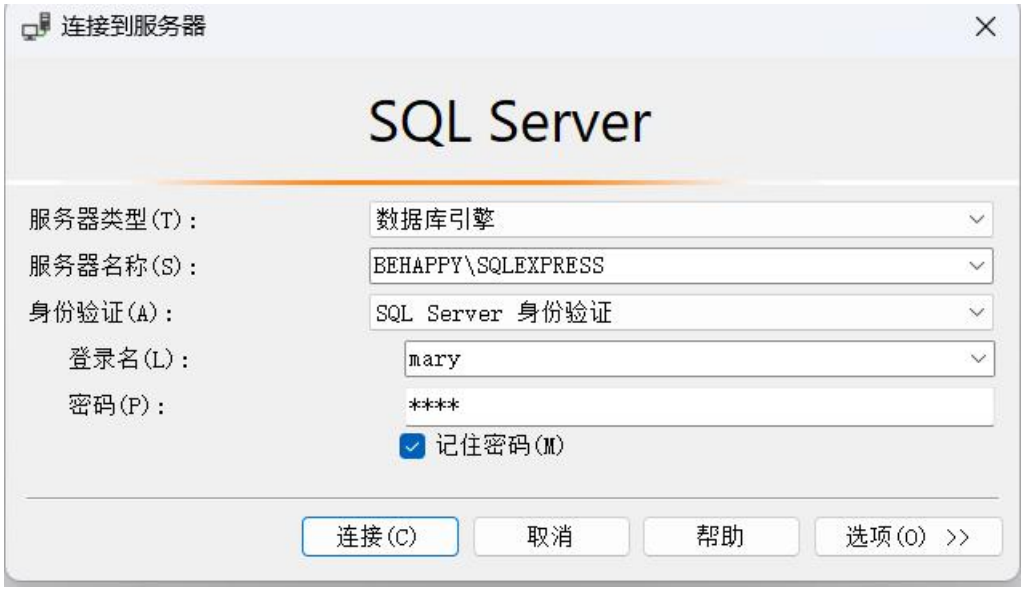
目的:掌握 Sql-server 的授权机制.

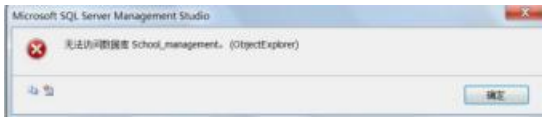
在 school 数据库中，完成以下操作：

1) 建立新用户 mary ， 密码 1234
2) 授予 mary 可以访问 School 数据库的权力



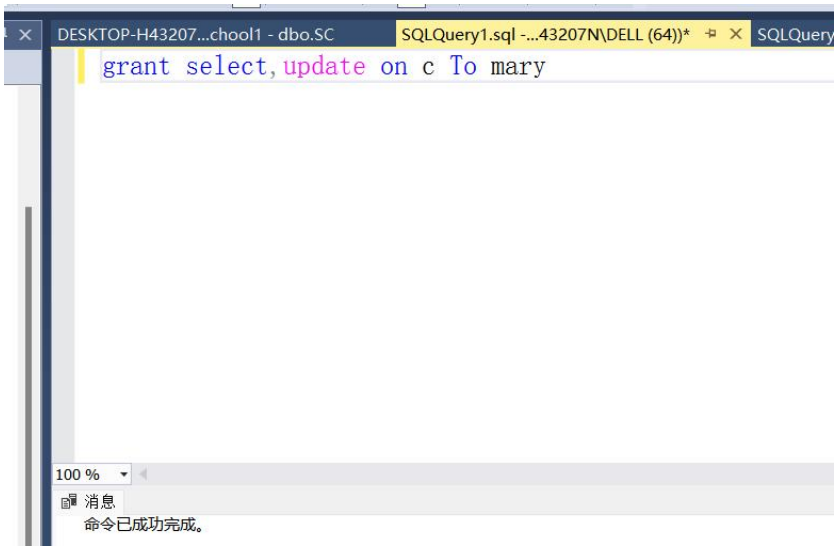
3) 以 mary 登录 sql-server ，
执行 select * from student ，记录执行结果，说明原因。





由于 mary 登录名的默认数据库 是 mary 数据库，因此当访问其他数据库时，会出现错误提示信息。

4）将 course 的查询、更改权限授予 mary



5）把查询 student 表和修改学生学号的权限授予用户 mary,且他能将此权限转授他人。

```
grant select,update(sno) on S To mary with grant option
```

```
0 %
消息
命令已成功完成。
```

6）把对 course 表的更改权限从 mary 收回

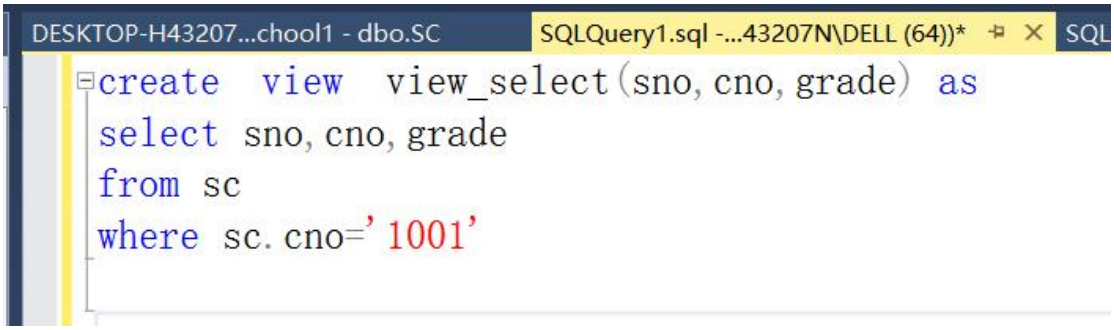
7）把第 5）小题授予 mary 的权限收回。

```
revoke select ,update(sno) on S to mary  cascade
```

```
0 %
消息
命令已成功完成。
```

8）mary 只能查询 ‘1001’ 号课程的学生成绩，请问如何授权

先建立视图 view_select



然后调用 grant

9）. 对 school 数据库分别做完整备份、差异备份和日志备份。(参考 14.2.3)
10）. 利用 9）中的三种备份，进行数据库的恢复（参考 14.2.4)

思考: 1 sp_addlogin ， sp_grantdbaccess 语句的区别.

sp_addlogin：创建新的 SQL Server 登录，该登录允许用户使用 SQL Server 身份验证连接到 SQL Server 实例。

2 如有 200 个人需要授权，SQL-SERVER 如何简化授权机制。

可以授权给角色来简化授权机制

试验十二 索引

目的：掌握索引的建立、删除的方法。
在 school 数据库中完成以下操作。
一 创建索引
1 建 student 的索引
为姓名建立索引，索引名：Ix_student_sname
为系科建立索引，索引名：Ix_student_sdept

```
create index ix_student_sname on student(sname);
create index ix_student_sdept on student(Sdept);
```

2 SC 的索引
为课程号建立索引： ix_sc_cno

```
create index ix_sc_cno on sc(cno);
```

3 Course 的索引
为课程名建立唯一性索引 ： Ix_course_cname

```
create unique index ix_course_cname on course(cname);
```

4 如何 SP_HELP 查看索引刚才建立的索引?
如何在企业管理器中查看索引?

```
exec sp_helpindex student;
exec sp_helpindex sc;
exec sp_helpindex course;
```

二 删除索引 course 表的索引 IX_course_cname

```
drop index ix_course_cname on Course;
```

三 思考：如何把索引 IX_student_sname 修改为唯一性索引?

```
drop index ix_student_sname on Student;
create unique index ix_student_sname on student(sname);
```

*四 思考建立索引的目的
1 输入下列存储过程，该程序生成大量数据供测试：

```
create procedure usp_makedata as
  declare @nCnt int , @sNo varchar(6) , @sname varchar(8)
  set @nCnt =12000  --计数器
  while @nCnt<999999
  begin
     set  @nCnt = @nCnt + 1
     set @sNo = convert(varchar(6) ,@nCnt)
```

```
        set @sName = '张'+@sno
        insert into student (sno,sname,ssex,sage) values ( @sno,@sname,'男',20)
      end
    return
```

2 exec usp_makedata --生成测试数据

3 输入下述测试程序:

```
create procedure usp_test as
  declare @nCount int ,@data int
  set @nCount=0
   while @nCount<100
   begin
    select @data=count(*) from student where sname <'张 3800' or sname>'张 8800'
    set @nCount =@nCount + 1
   end
```

4 测试

1) 建立姓名的索引, 查看运行时间(8 秒).

```
create index ix_student_sname on student(sname)  --建立索引
exec usp_test
```

2) 删除姓名索引, 查看运行时间 (2 分 11 秒), 比较与 1) 的时间长短。

```
 drop index student.ix_student_sname    --删除索引
exec usp_test
```

试验十三 存储过程

目的：掌握存储过程的概念、编程及使用

在 school 数据库中创建以下存储过程。

1 编写一个存储过程 usp_avgage ， 向客户端返回每个系科的学生平均年龄。

系科	平均年龄
JSJ	21
SX	20
◦◦◦	

1）编写存储过程的代码

2）调试、运行该存储过程。

```
create procedure usp_avgage
as
begin
    select Sdept,avg(Sage) as '平均年龄'
    from Student
    group by Sdept
end;
```

2 编写一个存储过程 usp_sdept，传入一个系科代码，返回该系的平均年龄，人数

```
create procedure usp_sdept
    @sdept varchar(15)
as
begin
    select avg(sage),count(*)
    from Student
    where Sdept = @sdept;
end;
```

3 编写存储过程 usp_updateGrade ， 传入参数为课程号,处理逻辑:

对传入的这门课,进行如下处理:

如某学生该门课成绩>80 ， 则加 2 分

如某学生该门课成绩>60 ， 则加 1 分

如某学生该门课成绩<=60 ,则减 1 分

并且返回此门课的每个学生的最新成绩： 学号 成绩.

```
create procedure usp_updateGrade
    @cno char(4)
as
    begin
        update sc
        set Grade = case
            when grade>80 then Grade + 2
            when grade>60 then Grade + 1
            else Grade - 1
        end
        where cno = @cno;

        select sno,grade
        from sc
        where cno = @cno;
    end;
```

4 编写存储过程 usp_g1 ，传入参数课程号，对该门课程进行如下处理，低于平均分 5 分不加分，低于平均分 0-5 分的加 1 分，高于平均分 0-5 加 1 分，高于平均分 5 分的加 2 分。

```
create procedure usp_g1
    @cno char(4)
as
    begin
        declare @avg_grade decimal(12,1)
        select @avg_grade = avg(grade)
        from sc
        where cno = @cno;

        update sc
        set grade = case
            when Grade < @avg_grade - 5 then grade
            when Grade < @avg_grade then Grade + 1
            when Grade < @avg_grade then Grade + 1
            when Grade < @avg_grade then Grade + 2
        end
        where cno = @cno
    end;
```

5 编写存储过程 usp_comp_age，比较 0001，0002 学生的年龄的高低，输出： XXXX 学生的年龄大

注意： XXXX 为学生的姓名

```
create procedure usp_comp_age
    @sno1 char(4),
```

```
    @sno2 char(4)
as
    begin
        declare @age1 int,@age2 int
        select @age1 = Sage
        from Student
        where sno = @sno1;

        select @age2 = Sage
        from Student
        where sno = @sno2;

        if @age1 > @age2
            select sname + '学生的年龄大' as 结果
            from Student
            where sno = @sno1;
        else if @age2 > @age1
            select sname + '学生的年龄大' as 结果
            from Student
            where sno = @sno2;
        else
            select '两个学生的年龄相同' as '结果';
    end;
```

6 编写存储过程 usp_comp，比较 1001，1002 课程的平均分的高低，输出： XXXX 课程的平均分高

```
CREATE PROCEDURE usp_comp
    @Cno1 char(4),
    @Cno2 char(4)
AS
BEGIN
    DECLARE @AvgGrade1 decimal(12, 1), @AvgGrade2 decimal(12, 1);
    SELECT @AvgGrade1 = AVG(Grade)
    FROM SC
    WHERE Cno = @Cno1;

    SELECT @AvgGrade2 = AVG(Grade)
    FROM SC
    WHERE Cno = @Cno2;

    IF @AvgGrade1 > @AvgGrade2
        SELECT Cname + '课程的平均分高' AS '结果'
        FROM Course
        WHERE Cno = @Cno1;
```

```
    ELSE IF @AvgGrade1 < @AvgGrade2
        SELECT Cname + '课程的平均分高' AS '结果'
        FROM Course
        WHERE Cno = @Cno2;
    ELSE
        SELECT '两门课程的平均分相同' AS '结果';
END;
```

7 编写存储过程 usp_comp_age1，比较两个学生的年龄的高低，两个学生的学号有参数输入，最后输出： XXXX 学生的年龄大。

注意： XXXX 为学生的姓名

```
create procedure usp_comp_age1
    @sno1 char(4),
    @sno2 char(4)
as
    begin
        declare @age1 int,@age2 int
        select @age1 = Sage
        from Student
        where sno = @sno1;

        select @age2 = Sage
        from Student
        where sno = @sno2;

        if @age1 > @age2
            select sname + '学生的年龄大' as 结果
            from Student
            where sno = @sno1;
        else if @age2 > @age1
            select sname + '学生的年龄大' as 结果
            from Student
            where sno = @sno2;
        else
            select '两个学生的年龄相同' as '结果';
    end;
```

8 利用第 8 题的存储过程，判断 0002，0003 学生的年龄大小。

```
exec usp_comp_age1 '0002','0003';
```

9 编写存储过程 usp_comp1，传入两参数，课程号 1，课程号 2；比较这两门课的平均分的高低，输出： XXXX 课程的平均分高

```
CREATE PROCEDURE usp_comp1
    @Cno1 char(4),
```

```
    @Cno2 char(4)
AS
BEGIN
    DECLARE @AvgGrade1 decimal(12, 1), @AvgGrade2 decimal(12, 1);
    SELECT @AvgGrade1 = AVG(Grade)
    FROM SC
    WHERE Cno = @Cno1;

    SELECT @AvgGrade2 = AVG(Grade)
    FROM SC
    WHERE Cno = @Cno2;

    IF @AvgGrade1 > @AvgGrade2
        SELECT Cname + '课程的平均分高' AS '结果'
        FROM Course
        WHERE Cno = @Cno1;
    ELSE IF @AvgGrade1 < @AvgGrade2
        SELECT Cname + '课程的平均分高' AS '结果'
        FROM Course
        WHERE Cno = @Cno2;
    ELSE
        SELECT '两门课程的平均分相同' AS '结果';
END;
```

10 编写存储过程 usp_comp_age2 ，比较两个学生的年龄的高低，两个学生的学号有参数输入，最后把年龄大的学生的姓名、性别返回客户端。

```
 create procedure usp_comp_age2
    @sno1 char(4),
    @sno2 char(4)
as
    begin
        declare @age1 int,@age2 int
        select @age1 = Sage
        from Student
        where sno = @sno1;

        select @age2 = Sage
        from Student
        where sno = @sno2;

        if @age1 > @age2
            select sname,Ssex
            from Student
            where sno = @sno1;
```

```
        else if @age2 > @age1
            select sname,Ssex
            from Student
            where sno = @sno2;
        else
            select '两个学生的年龄相同' as '结果';
    end;
```

11 编写存储过程 usp_comp2，传入两参数，课程号 1，课程号 2；比较这两门课的平均分的高低，最后把平均分高的课程的课程名返回客户端。

```
CREATE PROCEDURE usp_comp2
    @Cno1 char(4),
    @Cno2 char(4)
AS
BEGIN
    DECLARE @AvgGrade1 decimal(12, 1), @AvgGrade2 decimal(12, 1);
    SELECT @AvgGrade1 = AVG(Grade)
    FROM SC
    WHERE Cno = @Cno1;

    SELECT @AvgGrade2 = AVG(Grade)
    FROM SC
    WHERE Cno = @Cno2;

    IF @AvgGrade1 > @AvgGrade2
        SELECT Cname
        FROM Course
        WHERE Cno = @Cno1;
    ELSE IF @AvgGrade1 < @AvgGrade2
        SELECT Cname
        FROM Course
        WHERE Cno = @Cno2;
    ELSE
        SELECT '两门课程的平均分相同' AS '结果';
END;
```

12 编写存储过程 usp_t1，传入参数为学号，把该学号的课程 1001 的成绩减到 58 分。每次只能减 1 分，用循环完成。

```
create procedure usp_t1
    @sno char(4)
as
    begin
        declare @grade decimal(12,1);
        select grade
```

```
        from sc
        where Sno = @sno and cno = '1001';

        while @grade > 58
        begin
            set @grade = @grade - 1;
            update SC
            set Grade = @grade
            where sno = @sno and cno = '1001';
        end
    end;
```

13 编写存储过程 usp_t2，传入参数为系科，把该系科的学生每次加一岁，只要该系科有一个人的年龄达到 28 岁，即停止循环。每次只能减加 1 岁分，用循环完成。

```
create procedure usp_t2
    @sdept varchar(15)
as
    begin
        declare @age int;
        set @age = (
            select max(Sage)
            from Student
            where Sdept = @sdept
            );

        while @age < 28
        begin
            update Student
            set sage = Sage + 1
            where Sdept = @sdept;

            set @age = (
            select max(Sage)
            from Student
            where Sdept = @sdept
            );
        end
    end;
```

15 编写存储过程 usp_disp ， 传入参数为课程号,处理逻辑： 返回每个学生的成绩等级。成绩>=90 为优， 成绩>=80 为良,成绩>=70 为中,成绩>=60 为及格 ,成绩<=60 为不及格。返回结果如下:

学号 课程号 成绩 等第

```
    0001    1001      91     优
    0001    1002      78     中
        ..............
```

```
create procedure usp_disp
    @cno char(4)
as
    begin
        select sno,cno,grade,
            case
                when grade >= 90 then '优'
                when grade >= 80 then '良'
                when grade >= 70 then '中'
                when grade >= 60 then '及格'
                else '不及格'
            end as '等第'
        from SC
        where cno = @cno
    end;
```

16 编写一个存储过程，传入参数为学号，执行后，把该学号的学生按如下格式输出成绩：（注意：只有一行）

学号　姓名　1001 课程　1002 课程　1003 课程　平均分

```
create procedure usp_display_grade
    @sno char(6)
as
begin
    select s.sno,s.sname,
        (select sc.grade from sc where cno = '1001') as '1001 课程',
        (select sc.grade from sc where cno = '1002') as '1002 课程',
        (select sc.grade from sc where cno = '1003') as '1003 课程',
        avg(grade)
    from Student as s join sc on s.sno = sc.Sno
    where SC.Sno = @sno
    group by s.sno,s.sname
end;
```

17 编写一个存储过程，传入参数为 系科，执行后，把该系科的学生按如下格式输出学生成绩：

学号　姓名　1001 课程　1002 课程　1003 课程　平均分

```
create procedure usp_display_grade2
    @sdept varchar(15)
as
```

```
begin
    select s.sno,s.sname,
        (select sc.grade from sc where cno = '1001') as '1001 课程',
        (select sc.grade from sc where cno = '1002') as '1002 课程',
        (select sc.grade from sc where cno = '1003') as '1003 课程',
        avg(grade)
    from Student as s join sc on s.sno = sc.Sno
    where Sdept = @sdept
    group by s.sno,s.sname
end;
```

18 编写存储过程，统计男女生 1001，1002，1003 各自的选修人数，输出格式如下：

性别	1001 人数	1002 人数	1003 人数	小计
男	3	5	2	10
女	2	4	1	7
合计	5	9	3	17

（数据为示意数据）

```
create procedure usp_count_gender_courses
as
begin
    select '男' as '性别',
    sum(case when s.ssex = '男' and sc.cno = '1001' then 1 else 0 end) as
'1001 人数',
    sum(case when s.ssex = '男' and sc.cno = '1002' then 1 else 0 end) as
'1002 人数',
    sum(case when s.ssex = '男' and sc.cno = '1001' then 1 else 0 end) as
'1003 人数',
    sum(case when s.ssex = '男' then 1 else 0 end) as '小计'
    from Student as s join sc on s.sno = sc.sno
    union all
    select '女' as '性别',
    sum(case when s.ssex = '女' and sc.cno = '1001' then 1 else 0 end) as
'1001 人数',
    sum(case when s.ssex = '女' and sc.cno = '1002' then 1 else 0 end) as
'1002 人数',
    sum(case when s.ssex = '女' and sc.cno = '1001' then 1 else 0 end) as
'1003 人数',
    sum(case when s.ssex = '女' then 1 else 0 end) as '小计'
    from Student as s join sc on s.sno = sc.sno
    union all
    select '合计' as '性别',
    sum(case when sc.cno = '1001' then 1 else 0 end) as '1001 人数',
```

```
    sum(case when sc.cno = '1002' then 1 else 0 end) as '1002人数',
    sum(case when sc.cno = '1003' then 1 else 0 end) as '1003人数',
    count(*) as '小计'
    from SC
end
```

19 编写一个存储过程，利用存储过程的参数返回数据库服务器上的日期时间。

```
create procedure usp_getDateTime
    @currentDateTime datetime output
as
begin
    set @currentDateTime = getDate();
end
```

试验十四 触发器

目的：了解触发器的机制及编程设计、使用

在 school 数据库中完成以下操作：

一 建立学生表的触发器 usp_addstudent，当增加学生时，SX 系的学生不能超过 30 岁。

1 写出触发器

```
create trigger usp_addstudent on Student
after insert
    as
    begin
        if exists(select 1 from inserted where Sdept = 'SX' and sage >=
30)
        begin
            raiserror('数学系的学生不能超过三十岁',11,1)
            rollback
        end
    end;
```

2 执行下列语句块:

```
begin tran
  insert into student (sno,sname,ssex,sage,sdept)
          values (‘0701’,’刘欢’,’男’,26,’SX’)
  if @@error=0
    commit
  else
    rollback
end
```

观察该学生是否加入到 student

能加入到 student

3 执行下列语句块：

```
begin tran
    insert into student (sno,sname,ssex,sage,sdept) values (‘0702’,’赵欢’,’男’,31,’SX’)
    if @@error=0
      commit
    else
      rollback
  end
```

观察该学生是否加入到 student
不能加入到 student
二 实现下列触发器
1 不能删除年龄大于 25 岁的学生记录。

```
create trigger usp_nodelstudent on Student
for delete
    as
    begin
        if exists(select 1 from inserted where sage>25)
        begin
            raiserror('不能删除年龄大于大于 25 岁的学生记录',11,2)
            rollback
        end
    end;
```

2 建立触发器 usp_delcourse ， 使课程表中 1001, 1002, 1003 三门课不会被删除。
注意如何调试。

```
create trigger usp_delcourse on Course
for delete
    as
    begin
        if exists(select 1 from deleted where cno in ('1001','1002','1003'))
        begin
            raiserror('课程表中 1001，1002，1003 三门课不会被删除',11,3)
            rollback
        end
    end;
```

3 对学生表建立一触发器，使更改后的年龄只能比原值大

```
create trigger usp_updateage on Student
after update
    as
    begin
        if exists(
            select 1
            from inserted as i join deleted as d on i.Sno = d.Sno
            where i.Sage < d.Sage
        )
        begin
            raiserror('更改后的年龄只能比原值大',11,4);
            rollback
        end
    end;
```

4 对 sc 表建立触发器，使‘JSJ’系的学生不可选择 ‘1004’号课程

```
create trigger usp_sccheck on sc
after insert
    as
    begin
        if exists(
            select 1
            from inserted
            where Sdept = 'JSJ' and cno = '1004'
        )
        begin
            raiserror(''‘JSJ’系的学生不可选择 ‘1004’号课程',11,5)
            rollback
        end
    end;
```

5 对表 course 建触发器，实现级联删除的功能，但某课选修人数大于 3 则不能删除。
(先删除 sc 表对 course 的外码)

*三 建立一个触发器，使对 sc 表成绩的修改自动记录修改日志。
日志文件表(tablog)记录如下:
用户名 学号 课程号 原成绩 修改后成绩 更改日期

```
create table tablog (
    用户名 varchar(50),
    学号 varchar(50),
    课程号 varchar(50),
    原成绩 int,
    修改后成绩 int,
    更改日期 datetime
);

create trigger utr_sclog on sc
after update
    as
    begin
        insert into tablog
        select suser_name(),d.sno,d.cno,d.grade,i.grade,getdate()
        from inserted as i join deleted as d on i.sno = d.sno and i.cno
= d.cno
    end;
```

四 在 School 数据库中建立一个试验用的发票表 bill，然后为发票 bill 建立触发器 utr_money ，实现当输入单价和数量后，自动填写金额，即发票金额不输入，由单价、数量相乘后自动填写到金额中。

```
Create table bill(
     billID char(8),  --发票编号
     date datetime,    --开票日期
     product char(10), --产品编号
     price   int ,     --单价
     qty     int ,     --数量
     charge  int ,     --金额
     primary key (billid) )
```

```sql
create table bill (
    billid char(8),  --发票编号
    date datetime,    --开票日期
    product char(10), --产品编号
    price int,        --单价
    qty int,          --数量
    charge int,       --金额
    primary key (billid)
);

create trigger utr_money on bill
after insert ,update
    as
    begin
        update bill
        set charge = bill.price * bill.qty
        from bill join inserted as i on bill.billid = i.billid
    end;
```