

# 苏州大学实验报告

院、系	计算机学院	年级专业	19 计科图灵	姓名	张昊	学号	1927405160
课程名称	数据库课程实践					成绩	90
指导教师	赵朋朋	同组实验者	无	实验日期	2021 年 3 月 26 日		

实验名称

SQL 语言实验 9-11

## 试验九 安全性控制实验

### 一、实验目的

掌握 Sql-server 的授权机制.

### 二、实验内容

- 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。
  - 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。
  - 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。
  - 8) mary 只能查询 '1001' 号课程的学生成绩,请问如何授权
- 思考: 1 `sp_addlogin` , `sp_grantdbaccess` 语句的区别.
- 2 如有 200 个人需要授权, SQL-SERVER 如何简化授权机制。

### 三、实验结果

新用户 mary, 密码 1234

```
SP_ADDLOGIN 'mary', '1234'
```

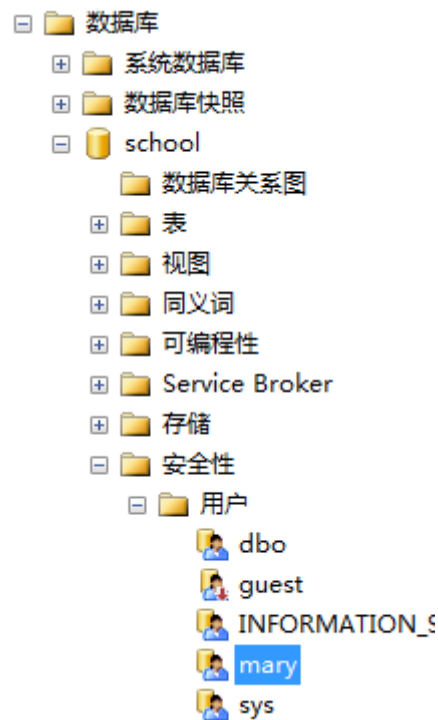
安全性

登录名

BUILTIN\Administrators  
DEV-238-01\SQLServer200  
DEV-238-01\SQLServer200  
DEV-238-01\SQLServer200  
mary  
NT AUTHORITY\SYSTEM  
sa

授予 mary 可以访问 School 数据库的权力

```
SP_GRANTDBACCESS mary
```



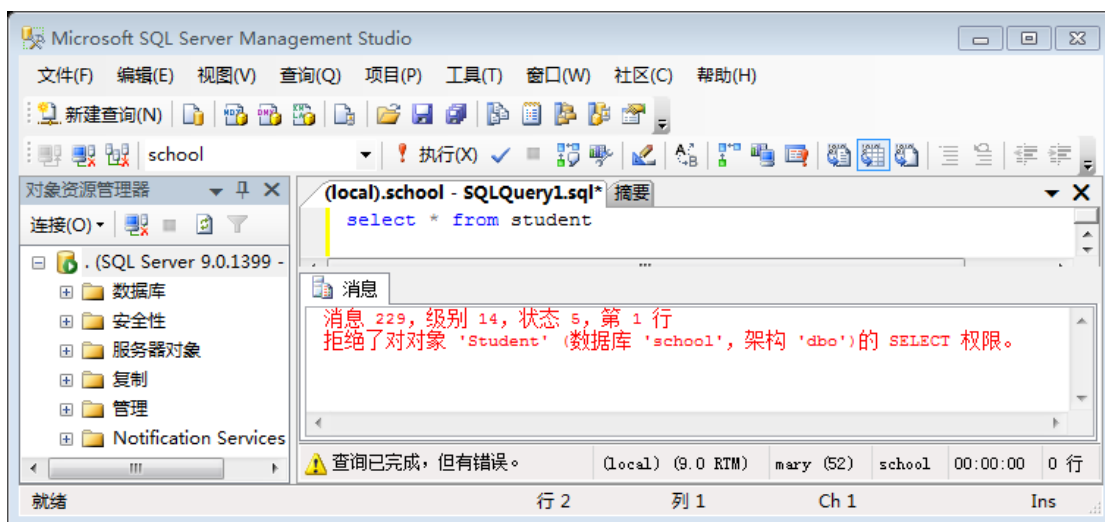
以 mary 登录 sql-server



出现登陆失败，“该用户与可信 SQL Server 连接无关联”的解决办法：

使用 Windows 身份验证重新连接数据库引擎，在左上的根目录点右键->属性->安全性，选择“服务器身份验证”为“SQL Server 和 Windows 身份验证模式(S)”，重启 SQL Server 服务。





无法查询到数据，因为 mary 没有查询 student 表的权限。

将 course 的查询、更改权限授予 mary

```
GRANT SELECT, UPDATE ON course TO mary
```

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 Student TO mary WITH GRANT OPTION
```

把对 course 表的更改权限从 mary 收回

```
REVOKE UPDATE ON course FROM mary
```

把第 5) 小题授予 mary 的权限收回

```
REVOKE SELECT, UPDATE(Sno) ON Student FROM mary CASCADE
```

mary 只能查询 '1001' 号课程的学生成绩

```
CREATE VIEW v_mary_select(Sno, Cno, Grade) AS
SELECT Sno, Cno, Grade FROM SC WHERE Cno = 1001
```

```
GRANT SELECT ON v_mary_select TO mary
```

#### 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。

sp\_grantdbaccess: 将数据库用户添加到当前数据库。

简单来说，sp\_addlogin 增加用户，sp\_grantdbaccess 使用户成为数据库合法用户

#### 思考 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。

#### 四、实验总结

通过实验，我掌握了 Sql-server 的授权机制。

## 试验十 存储过程

### 一、实验目的

掌握存储过程的概念、编程及使用

### 二、实验内容

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。

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

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。

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。

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

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

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

8 利用第 8 题的存储过程, 判断 0002, 0003 学生的年龄大小。

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

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。

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。

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。

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。

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

.....

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
----	----	---------	---------	---------	-----

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
----	----	---------	---------	---------	-----

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）

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。

思考：何时需要存储过程？

### 三、实验结果

存储过程 usp\_avgage:

```
CREATE PROCEDURE usp_avgage AS
    SELECT Sdept, AVG(Sage) FROM student GROUP BY Sdept;
RETURN;
```

usp\_avgage

Sdept	(无列名)
1 JSJ	21
2 SX	21

存储过程 usp\_sdept:

```
CREATE PROCEDURE usp_sdept(@dept CHAR(10)) AS
    SELECT AVG(Sage) AS AVG_SAGE, COUNT(*) AS DEPT_COUNT
    FROM student
    WHERE Sdept = @dept;
RETURN;
```

EXEC usp\_sdept 'JSJ'

AVG_SAGE	DEPT_COUNT
1 21	8

存储过程 usp\_updateGrade:

```
CREATE PROCEDURE usp_updateGrade(@Cno CHAR(4)) AS
    UPDATE SC SET Grade = Grade + 2
    WHERE Cno = @Cno AND Grade > 80;
    UPDATE SC SET Grade = Grade + 1
```

```

WHERE Cno = @Cno AND Grade > 60;
UPDATE SC SET Grade = Grade - 1
WHERE Cno = @Cno AND Grade <= 60;
SELECT Sno, Grade
FROM SC
WHERE Cno = @Cno;
RETURN;

```

存储过程 usp\_g1:

```

CREATE PROCEDURE usp_g1(@Cno CHAR(4)) AS
DECLARE @AVG_GRADE INT;
SELECT @AVG_GRADE = AVG(Grade)
FROM SC
WHERE Cno = @Cno;
UPDATE SC
SET Grade = CASE
    WHEN Grade >= @AVG_GRADE - 5 AND Grade <= @AVG_GRADE + 5 THEN Grade+1
    WHEN Grade > @AVG_GRADE + 5 THEN Grade + 2
    ELSE Grade
END;
RETURN;

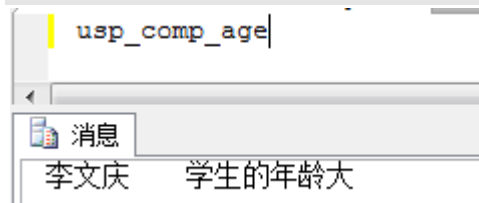
```

存储过程 usp\_comp\_age:

```

CREATE PROCEDURE usp_comp_age AS
DECLARE @age1 INT, @age2 INT;
DECLARE @name1 CHAR(10), @name2 CHAR(10);
SELECT
    @age1 = Sage,
    @name1 = Sname
FROM student
WHERE Sno = '0001';
SELECT
    @age2 = Sage,
    @name2 = Sname
FROM student
WHERE Sno = '0002';
IF ( @age1 > @age2 )
    PRINT @name1 + '学生的年龄大';
ELSE
    PRINT @name2 + '学生的年龄大';
RETURN;

```



存储过程 usp\_comp:

```

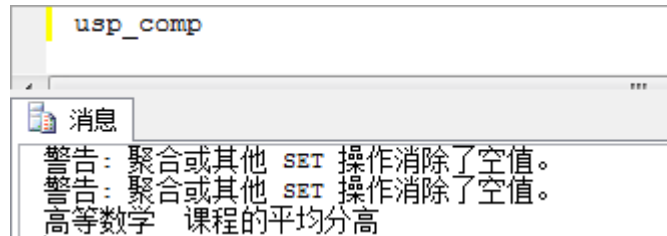
CREATE PROCEDURE usp_comp AS

```

```

DECLARE @AVG_GRADE1 INT, @AVG_GRADE2 INT;
DECLARE @Cname1 CHAR(10), @Cname2 CHAR(10);
SELECT @Cname1 = Cname FROM course WHERE Cno = '1001';
SELECT @Cname2 = Cname FROM course WHERE Cno = '1002';
SELECT @AVG_GRADE1 = AVG(Grade)
FROM SC
WHERE Cno = '1001';
SELECT @AVG_GRADE2 = AVG(Grade)
FROM SC
WHERE Cno = '1002';
IF ( @AVG_GRADE1 > @AVG_GRADE2 )
    PRINT @Cname1 + '课程的平均分高';
ELSE
    PRINT @Cname2 + '课程的平均分高';
RETURN;

```



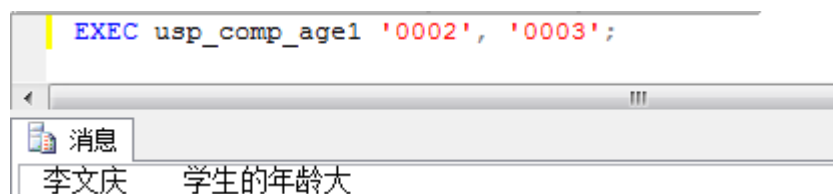
存储过程 usp\_comp\_age1:

```

CREATE PROCEDURE usp_comp_age1( @sno1 CHAR(4), @sno2 CHAR(4) ) AS
DECLARE @age1 INT, @age2 INT;
DECLARE @name1 CHAR(10), @name2 CHAR(10);
SELECT
    @age1 = Sage,
    @name1 = Sname
FROM student
WHERE Sno = @sno1;
SELECT
    @age2 = Sage,
    @name2 = Sname
FROM student
WHERE Sno = @sno2;
IF ( @age1 > @age2 )
    PRINT @name1 + '学生的年龄大';
ELSE
    PRINT @name2 + '学生的年龄大';
RETURN;

```

判断 0002, 0003 学生的年龄大小:



存储过程 usp\_comp1:

```
CREATE PROCEDURE usp_comp1( @cno1 CHAR(4), @cno2 CHAR(4) ) AS
    DECLARE @AVG_GRADE1 INT, @AVG_GRADE2 INT;
    DECLARE @Cname1 CHAR(10), @Cname2 CHAR(10);
    SELECT @Cname1 = Cname FROM course WHERE Cno = @cno1;
    SELECT @Cname2 = Cname FROM course WHERE Cno = @cno2;
    SELECT @AVG_GRADE1 = AVG(Grade)
    FROM SC
    WHERE Cno = @cno1;
    SELECT @AVG_GRADE2 = AVG(Grade)
    FROM SC
    WHERE Cno = @cno2;
    IF ( @AVG_GRADE1 > @AVG_GRADE2 )
        PRINT @Cname1 + '课程的平均分高';
    ELSE
        PRINT @Cname2 + '课程的平均分高';
RETURN;
```

```
EXEC usp_comp1 '1001', '1002'
```

消息

警告: 聚合或其他 SET 操作消除了空值。  
警告: 聚合或其他 SET 操作消除了空值。  
高等数学 课程的平均分高

存储过程 usp\_comp\_age2:

```
CREATE PROCEDURE usp_comp_age2( @sno1 CHAR(4), @sno2 CHAR(4) ) AS
    DECLARE @age1 INT, @age2 INT;
    DECLARE @name1 CHAR(10), @name2 CHAR(10);
    DECLARE @return_no CHAR(4);
    SELECT @age1 = Sage
    FROM student
    WHERE Sno = @sno1;
    SELECT @age2 = Sage
    FROM student
    WHERE Sno = @sno2;
    IF ( @age1 > @age2 )
        SET @return_no = @sno1;
    ELSE
        SET @return_no = @sno2;
    SELECT Sname, Ssex
    FROM student
    WHERE Sno = @return_no;
RETURN;
```



EXEC usp\_comp\_age2 '0001', '0002'

结果		消息
Sname	Ssex	
1	李文庆	男

存储过程 usp\_comp2:

```
CREATE PROCEDURE usp_comp2( @cno1 CHAR(4), @cno2 CHAR(4) ) AS
    DECLARE @AVG_GRADE1 INT, @AVG_GRADE2 INT;
    DECLARE @return_no CHAR(4);
    SELECT @AVG_GRADE1 = AVG(Grade)
    FROM SC
    WHERE Cno = @cno1;
    SELECT @AVG_GRADE2 = AVG(Grade)
    FROM SC
    WHERE Cno = @cno2;
    IF ( @AVG_GRADE1 > @AVG_GRADE2 )
        SET @return_no = @cno1;
    ELSE
        SET @return_no = @cno2;
    SELECT Cname
    FROM course
    WHERE Cno = @return_no;
RETURN;
```

EXEC usp\_comp2 '1001', '1002'

结果		消息
Cname		
1	高等数学	

存储过程 usp\_t1:

```
CREATE PROCEDURE usp_t1(@sno CHAR(4)) AS
    DECLARE @grade INT;
    SELECT @grade = Grade FROM SC WHERE Cno = '1001' AND Sno = @sno;
    WHILE (@grade > 58)
    BEGIN
        UPDATE SC SET Grade = Grade - 1 WHERE Cno = '1001' AND Sno = @sno;
        SET @grade = @grade - 1;
    END
RETURN;
```

存储过程 usp\_t2:

```
CREATE PROCEDURE usp_t2(@dept CHAR(10)) AS
    DECLARE @stu_count INT;
    SELECT @stu_count = COUNT(*)
    FROM student
    WHERE Sdept = @dept AND Sage >= 28;
    WHILE (@stu_count <= 0)
    BEGIN
```

```

UPDATE student
SET Sage = Sage + 1
WHERE Sdept = @dept;
SELECT @stu_count = COUNT(*)
FROM student
WHERE Sdept = @dept AND Sage >= 28;

END
RETURN;

```

存储过程 usp\_disp:

```

CREATE PROCEDURE usp_disp(@cno CHAR(4)) AS
SELECT
    Sno AS '学号',
    Cno AS '课程号',
    Grade AS '成绩',
    CASE
        WHEN Grade >= 90 THEN '优'
        WHEN Grade >= 80 AND Grade < 90 THEN '良'
        WHEN Grade >= 70 AND Grade < 80 THEN '中'
        WHEN Grade >= 60 AND Grade < 70 THEN '及格'
        ELSE '不及格'
    END AS '等第'
FROM SC
WHERE Cno = @cno;
RETURN;

```

EXEC usp\_disp '1001'

	学号	课程...	成绩	等第
1	0001	1001	92	优
2	0002	1001	91	优
3	0003	1001	77	中
4	0004	1001	97	优
5	0009	1001	93	优
6	0081	1001	90	优
7	0091	1001	93	优
8	0092	1001	98	优
9	8001	1001	91	优
10	8002	1001	96	优
11	8003	1001	91	优
12	8004	1001	92	优
13	8006	1001	NULL	不及格
14	8007	1001	99	优
15	8008	1001	91	优

输出学生成绩:

```

CREATE PROCEDURE usp_student_course(@sno CHAR(4)) AS
DECLARE @grade1001 INT, @grade1002 INT, @grade1003 INT;
SELECT @grade1001 = Grade FROM SC WHERE Cno = '1001' AND Sno = @sno;
SELECT @grade1002 = Grade FROM SC WHERE Cno = '1002' AND Sno = @sno;
SELECT @grade1003 = Grade FROM SC WHERE Cno = '1003' AND Sno = @sno;

```

```

SELECT
    Sno AS '学号',
    Sname AS '姓名',
    @grade1001 AS '1001课程',
    @grade1002 AS '1002课程',
    @grade1003 AS '1003课程',
    (@grade1001 + @grade1002 + @grade1003) / 3.0 AS '平均分'
FROM Student
WHERE Sno = @sno;
RETURN;

```

```
EXEC usp_student_course '0001'
```

结果		消息				
学号	姓名	1001课程	1002课程	1003课程	平均分	
1	0001	周志林	92	92	76	86.666666

输出某一系科学生成绩：（使用游标变量结果集，表变量保存结果，利用上面定义的存储过程）

```

CREATE PROCEDURE usp_dept_course(@dept CHAR(10)) AS
    DECLARE @result TABLE (
        Sno CHAR(4),
        Sname CHAR(10),
        grade1001 INT,
        grade1002 INT,
        grade1003 INT,
        AVG_grade DECIMAL(5, 1)
    );
    DECLARE @no CHAR(4);
    -- 声明一个游标用来遍历查询到的结果
    DECLARE C_STUDENT_NO CURSOR FOR
    SELECT Sno FROM Student WHERE Sdept = @dept;
    OPEN C_STUDENT_NO;
    FETCH next FROM C_STUDENT_NO INTO @no;
    -- 使用游标遍历集合
    WHILE @@FETCH_STATUS = 0
    BEGIN
        INSERT INTO @result EXEC usp_student_course @no;
        FETCH next FROM C_STUDENT_NO INTO @no;
    END
    CLOSE C_STUDENT_NO;
    DEALLOCATE C_STUDENT_NO;
    SELECT
        Sno AS '学号',
        Sname AS '姓名',
        grade1001 AS '1001课程',
        grade1002 AS '1002课程',
        grade1003 AS '1003课程',
        AVG_grade AS '平均分'

```

```
FROM @result;
RETURN;
```

```
EXEC usp_dept_course 'JSJ'
```

	学号	姓名	1001课程	1002课程	1003课程	平均分
1	0002	李文庆	91	90	67	82.7
2	0004	杨秀红	97	78	65	80.0
3	0078	王振	NULL	NULL	NULL	NULL
4	8003	钱凯	91	NULL	92	NULL
5	8005	张英	NULL	NULL	NULL	NULL
6	8006	赵章	NULL	92	20	NULL
7	8007	钱利	99	95	92	95.3
8	8008	王铁	91	99	92	94.0

统计男女生 1001, 1002, 1003 各自的选修人数: (使用临时视图和表变量组织结果)

```
CREATE PROCEDURE usp_course_statistics AS
    DECLARE @result TABLE (
        sex CHAR(6),
        cnt1001 INT,
        cnt1002 INT,
        cnt1003 INT,
        total INT
    );
    WITH statistics_result(Sex, Cno, cnt)
    AS (SELECT Ssex, Cno, COUNT(Student.Sno) AS cnt
        FROM Student, SC
        WHERE Student.Sno = SC.Sno
        GROUP BY Ssex, Cno
        HAVING Cno IN ('1001', '1002', '1003'))
    INSERT INTO @result
    SELECT
        r.sex,
        r.cnt,
        s.cnt,
        t.cnt,
        r.cnt + s.cnt + t.cnt
    FROM
        statistics_result AS r,
        statistics_result AS s,
        statistics_result AS t
    WHERE r.sex = s.sex AND s.sex = t.sex AND r.cno = '1001' AND
        s.cno = '1002' AND t.cno = '1003';
    INSERT INTO @result
    SELECT '合计', SUM(cnt1001), SUM(cnt1002), SUM(cnt1003), SUM(total)
    FROM @result;
    SELECT
```

```
sex AS '性别',
cnt1001 AS '1001人数',
cnt1002 AS '1002人数',
cnt1003 AS '1003人数',
total AS '小计'
FROM @result;
RETURN;
```

	性别	1001人数	1002人数	1003人数	小计
1	男	10	7	9	26
2	女	5	5	5	15
3	合计	15	12	14	41

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_time (@date DATETIME OUTPUT) AS
SET @date = GETDATE();
RETURN;
```

```
DECLARE @now DATETIME
EXEC usp_time @now OUTPUT
SELECT @now
```

(无列名)
1 2021-04-09 11:06:42.840

#### 思考：何时需要存储过程

当一个事务涉及到多个 SQL 语句时或者涉及到对多个表的操作时就要考虑用存储过程；当在一个事务的完成需要很复杂的逻辑时、比较复杂的统计和汇总时就要考虑使用存储过程。

#### 四、实验总结

通过实验，我掌握存储过程的概念、编程及使用。了解了临时视图、表变量和游标的使用方法。

## 试验十一 触发器

### 一、实验目的

了解触发器的机制及编程设计、使用

### 二、实验内容

(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

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

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

(二) 实现下列触发器

1 不能删除年龄大于 25 岁的学生记录。

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。

3 对学生表建立一触发器，使更改后的年龄只能比原值大

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

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

(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
```

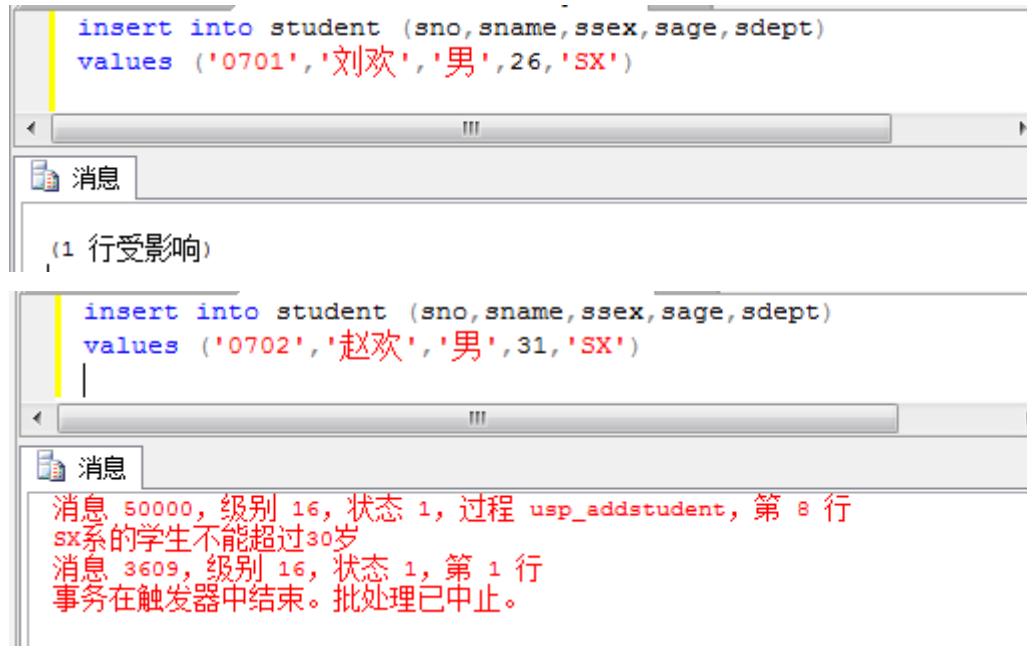
```
price    int,      --单价
qty      int,      --数量
charge   int,      --金额
primary key (billid)
```

思考: 触发器中 inserted , deleted 表的作用? 在触发器中如没有用到此两个表中的任何一个, 你认为触发器还有意义吗?

### 三、实验结果

触发器 usp\_addstudent:

```
CREATE TRIGGER usp_addstudent ON Student AFTER INSERT AS
BEGIN
    DECLARE @dept CHAR(10);
    DECLARE @age SMALLINT;
    SELECT @dept = Sdept, @age = Sage FROM INSERTED;
    IF ( @dept = 'SX' AND @age > 30 )
    BEGIN
        RAISERROR('SX系的学生不能超过30岁', 16, 1);
        ROLLBACK;
    END
END
```



不能删除年龄大于 25 岁的学生记录:

```
CREATE TRIGGER usp_delstudent ON Student FOR DELETE AS
BEGIN
    DECLARE @cnt INT;
    SELECT @cnt = COUNT(*) FROM DELETED WHERE Sage > 25;
    IF ( @cnt > 0 )
    BEGIN
        RAISERROR('不能删除年龄大于25岁的学生记录', 16, 1);
        ROLLBACK TRANSACTION;
    END
END
```

END

END

测试:

```
DELETE FROM student
WHERE Sno = '0701'
```

消息

消息 50000, 级别 16, 状态 1, 过程 usp\_delstudent, 第 7 行  
不能删除年龄大于25岁的学生记录  
消息 3609, 级别 16, 状态 1, 第 1 行  
事务在触发器中结束。批处理已中止。

建立触发器 usp\_delcourse , 使课程表中 1001, 1002, 1003 三门课不会被删除:

```
CREATE TRIGGER usp_delcourse ON course FOR DELETE AS
```

```
BEGIN
```

```
    DECLARE @cnt INT;
```

```
    SELECT @cnt = COUNT(*)
```

```
    FROM DELETED
```

```
    WHERE Cno IN ('1001', '1002', '1003');
```

```
    IF ( @cnt > 0 )
```

```
    BEGIN
```

```
        RAISERROR('不能删除1001, 1002, 1003三门课', 16, 1);
```

```
        ROLLBACK TRANSACTION;
```

```
    END
```

```
END
```

测试:

```
DELETE FROM course
WHERE Cno = '1003'
```

消息

消息 50000, 级别 16, 状态 1, 过程 usp\_delcourse, 第 9 行  
不能删除 1001, 1002, 1003 三门课  
消息 3609, 级别 16, 状态 1, 第 1 行  
事务在触发器中结束。批处理已中止。

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_updatestudent ON Student FOR UPDATE AS
```

```
BEGIN
```

```
    DECLARE @cnt INT;
```

```
    SELECT @cnt = COUNT(*)
```

```
    FROM INSERTED, DELETED
```

```
    WHERE INSERTED.Sno = DELETED.Sno AND INSERTED.Sage <= DELETED.Sage;
```

```
    IF ( @cnt > 0 )
```

```
    BEGIN
```

```
        RAISERROR('更改后的年龄只能比原值大', 16, 1);
```

```
        ROLLBACK TRANSACTION;
```



END

END

测试:

```
SELECT * FROM Student WHERE Sno = '0001'
```

结果 消息

	Sno	Sname	Ssex	Sage	Sdept
1	0001	周志林	男	20	SX

```
UPDATE Student SET Sage = 25 WHERE Sno = '0001'
```

消息

(1 行受影响)

```
UPDATE Student SET Sage = 18 WHERE Sno = '0001'
```

消息

消息 50000, 级别 16, 状态 1, 过程 usp\_updatestudent, 第 9 行  
更改后的年龄只能比原值大  
消息 3609, 级别 16, 状态 1, 第 1 行  
事务在触发器中结束。批处理已中止。

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_updateSC ON SC FOR INSERT AS
```

```
BEGIN
```

```
    DECLARE @cnt INT;
```

```
    SELECT @cnt = COUNT(*)
```

```
    FROM INSERTED, Student
```

```
    WHERE INSERTED.Sno = Student.Sno AND
```

```
          Student.Sdept = 'JSJ' AND
```

```
          INSERTED.Cno = '1004';
```

```
    IF ( @cnt > 0 )
```

```
    BEGIN
```

```
        RAISERROR(' 'JSJ' 系的学生不可选择 '1004' 号课程', 16, 1);
```

```
        ROLLBACK TRANSACTION;
```

```
    END
```

```
END
```

测试:

SQL Query: `SELECT * FROM Student WHERE Sdept = 'JSJ'`  
`SELECT * FROM sc`

Results:

	Sno	Sname	Ssex	Sage	Sdept
1	0002	李文庆	男	23	JSJ
2	0004	杨秀红	女	21	JSJ
3	0078	王振	男	21	JSJ
4	8003	钱凯	男	22	JSJ
5	8005	张英	女	21	JSJ
6	8006	赵章	女	22	JSJ
7	8007	钱利	男	23	JSJ
8	8008	王铁	男	21	JSJ

SQL Query: `INSERT INTO SC VALUES ('0002', '1004', 90)`

Message:

消息 50000, 级别 16, 状态 1, 过程 usp\_updateSC, 第 9 行  
 'JSJ'系的学生不可选择 '1004' 号课程  
 消息 3609, 级别 16, 状态 1, 第 1 行  
 事务在触发器中结束。批处理已中止。

SQL Query: `SELECT * FROM SC`

Results:

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

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:

```
CREATE TRIGGER usp_delcourse ON course
INSTEAD OF DELETE AS
BEGIN
    DECLARE @cnt INT;
    SELECT @cnt = COUNT(*)
    FROM DELETED, SC
    WHERE DELETED.Cno = SC.Cno;
    IF ( @cnt > 3 ) BEGIN
        RAISERROR('选修人数大于3, 不能删除', 16, 1);
        ROLLBACK TRANSACTION;
    END
    ELSE BEGIN
        DELETE FROM SC
        WHERE SC.Cno IN (SELECT Cno FROM DELETED);
        DELETE FROM course
```

```
WHERE course.Cno IN (SELECT Cno FROM DELETED);
END
END
```

测试:

```
SELECT * FROM SC
SELECT * FROM course
```

	Sno	Cno	Grade
1	0001	1001	NULL
2	0001	1002	NULL
3	0002	1001	NULL
4	0003	1001	NULL
5	0003	1002	NULL
6	0004	1001	NULL

	Cno	Cname	Cpno	Ccredit
1	1001	高等数学		5
2	1002	离散数学	1001	3

```
DELETE FROM course WHERE Cno = '1001'
```

消息 50000, 级别 16, 状态 1, 过程 usp\_delcourse, 第 9 行  
选修人数大于3, 不能删除  
消息 3609, 级别 16, 状态 1, 第 1 行  
事务在触发器中结束。批处理已中止。

```
DELETE FROM course WHERE Cno = '1002'
```

(2 行受影响)  
(1 行受影响)  
(1 行受影响)

```
SELECT * FROM SC
SELECT * FROM course
```

	Sno	Cno	Grade
1	0001	1001	NULL
2	0002	1001	NULL
3	0003	1001	NULL
4	0004	1001	NULL

	Cno	Cname	Cpno	Ccredit
1	1001	高等数学		5
2	1003	程序设计		5

对 sc 表成绩的修改自动记录修改日志:

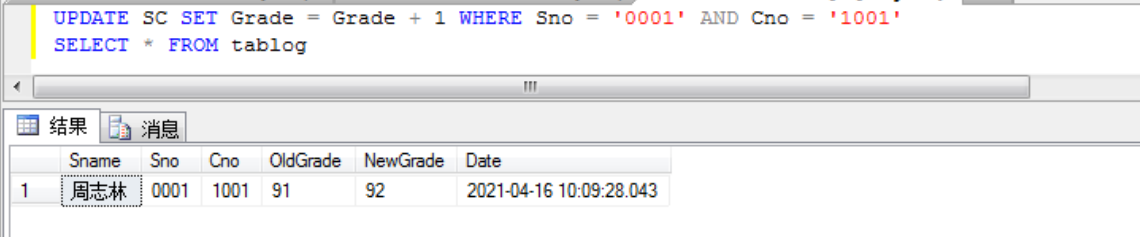
创建表:

```
CREATE TABLE tablog (  
    Sname CHAR(10),  
    Sno CHAR(6),  
    Cno CHAR(4),  
    OldGrade INT,  
    NewGrade INT,  
    Date DATETIME,  
    PRIMARY KEY (Sno, Cno, Date));
```

创建触发器:

```
CREATE TRIGGER ups_logscupdate ON SC FOR UPDATE AS  
BEGIN  
    INSERT INTO tablog  
    SELECT  
        Student.Sname,  
        Student.Sno,  
        INSERTED.Cno,  
        DELETED.Grade,  
        INSERTED.Grade,  
        GETDATE()  
    FROM  
        Student,  
        INSERTED,  
        DELETED  
    WHERE INSERTED.Sno = Student.Sno AND  
        INSERTED.Cno = DELETED.Cno AND  
        INSERTED.Sno = DELETED.Sno;  
END
```

测试:



The screenshot shows a SQL query window with the following SQL code:

```
UPDATE SC SET Grade = Grade + 1 WHERE Sno = '0001' AND Cno = '1001';  
SELECT * FROM tablog
```

Below the query window, there is a results pane with two tabs: "结果" (Results) and "消息" (Messages). The "结果" tab is active, displaying a table with the following data:

	Sname	Sno	Cno	OldGrade	NewGrade	Date
1	周志林	0001	1001	91	92	2021-04-16 10:09:28.043

触发器 utr\_money :

```
CREATE TRIGGER utr_money ON bill  
INSTEAD OF INSERT AS  
BEGIN  
    INSERT INTO bill  
    SELECT  
        billID,  
        date,  
        product,  
        price,
```

```

    qty,
    price * qty
FROM INSERTED;
END

INSERT INTO bill (billID, date, product, price, qty)
VALUES ('1001', GETDATE(), '0001', 28, 10);
SELECT * FROM bill;

```

结果

消息

	billID	date	product	price	qty	charge
1	1001	2021-04-16 10:18:20.817	0001	28	10	280

#### 思考: 触发器中 inserted , deleted 表的作用?

inserted 表和 deleted 表用于存放对表中数据行的修改信息, 他们是触发器执行时自动创建的。

- ✓ inserted 表: 用来存储 INSERT 和 UPDATE 语句所影响的行的副本。可以从 inserted 表检查插入的数据是否满足需求, 如不满足则撤消操作。
- ✓ deleted 表: 用来存储 DELETE 和 UPDATE 语句所影响行的副本。可以从 deleted 表中检查删除的数据行是否能删除。

#### 在触发器中如没有用到此两个表中的任何一个, 你认为触发器还有意义吗?

一些触发器那可是仅更删改的执行时间, 没有具体统计更删改的内容, 这样的触发器还是有存在的意义的。

#### 四、实验总结

通过实验, 我了解了触发器的机制, 以及编程设计、使用触发器的方法。此外对于 inserted , deleted 表的作用有了一定的认识。