苏 州 大 学 实 验 报 告

院、系	计科院	年级专业	软件工程	姓名	高歌	学号	2030416018
课程名称	数据库课程实践					同组实验者	/
指导教师	瞿剑锋	实验日期	2022-04-06	成绩			

实 验 名 称　SQL 语言（试验十）存储过程

试验十 存储过程

目的： 掌握存储过程的概念、编程及使用。

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

```
1. CREATE PROCEDURE usp_avgage
2. AS
3. SELECT Sdept, AVG(SAge) AS AvgAge
4. FROM Student
5. GROUP BY Sdept;
```

2）调试、运行该存储过程。

```
1. usp_avgage;
```

	Sdept	AvgAge
1	JSJ	21
2	SX	21

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

```
1. CREATE PROCEDURE usp_sdept
2. (@Sdept varchar(10))
3. AS
4. SELECT AVG(SAge) AS AvgAge, COUNT(*) AS Count
5. FROM Student
6. WHERE Sdept = @Sdept;
```

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.

```
1. CREATE PROCEDURE usp_updateGrade
2. (@cno varchar(10))
3. AS
4. UPDATE SC
5. SET Grade = CASE
6.   WHEN Grade > 80 THEN Grade + 2
7.   WHEN Grade > 60 THEN Grade + 1
8.   ELSE Grade - 1
9. END
10. WHERE Cno = @cno;
11. SELECT Sno, Grade
12. FROM SC
13. WHERE Cno = @cno;
14. RETURN;
```

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。

```
1. CREATE PROCEDURE usp_g1
2. (@cno varchar(10))
3. AS
4. DECLARE @avgGrade float;
5. SELECT @avgGrade = AVG(Grade)
6. FROM SC
7. WHERE Cno = @cno;
8. UPDATE SC
9. SET Grade = CASE
10.   WHEN Grade < @avgGrade - 5 THEN Grade
11.   WHEN Grade <= @avgGrade + 5 THEN Grade + 1
12.   ELSE Grade + 2
13. END
14. WHERE Cno = @cno;
15. SELECT Sno, Grade
16. FROM SC
17. WHERE Cno = @cno;
18. RETURN;
```

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
AS
DECLARE @s1_age int, @s2_age int;
DECLARE @s1_name varchar(10), @s2_name varchar(10);
SELECT @s1_age = Sage, @s1_name = Sname
FROM Student
WHERE Sno = '0001';
SELECT @s2_age = Sage, @s2_name = Sname
FROM Student
WHERE Sno = '0002';
IF @s1_age > @s2_age
   PRINT @s1_name + '学生的年龄大'
ELSE
   PRINT @s2_name + '学生的年龄大'
RETURN;
```

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
AS
DECLARE @c1_avgGrade float, @c2_avgGrade float;
DECLARE @c1_Cno varchar(10), @c2_Cno varchar(10);
SELECT @c1_avgGrade = AVG(Grade), @c1_Cno = MAX(Cno)
FROM SC
WHERE Cno = '1001';
SELECT @c2_avgGrade = AVG(Grade), @c2_Cno = MAX(Cno)
FROM SC
WHERE Cno = '1002';
IF @c1_avgGrade > @c2_avgGrade
   PRINT @c1_Cno + '课程的平均分高'
ELSE
   PRINT @c2_Cno + '课程的平均分高'
RETURN;
```

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。

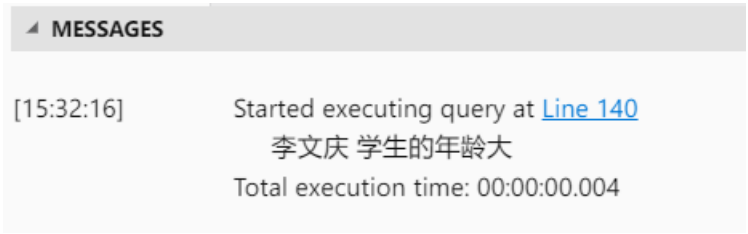
注意： XXXX 为学生的姓名

```
CREATE PROCEDURE usp_comp_age1
(@s1_Sno varchar(10), @s2_Sno varchar(10))
AS
DECLARE @s1_age int, @s2_age int;
DECLARE @s1_name varchar(10), @s2_name varchar(10);
SELECT @s1_age = Sage, @s1_name = Sname
FROM Student
WHERE Sno = @s1_Sno;
```

```
9. SELECT @s2_age = Sage, @s2_name = Sname
10. FROM Student
11. WHERE Sno = @s2_Sno;
12. IF @s1_age > @s2_age
13.   PRINT @s1_name + '学生的年龄大'
14. ELSE
15.   PRINT @s2_name + '学生的年龄大'
16. RETURN;
```

8 利用第 8 题的存储过程，判断 0002，0003 学生的年龄大小。

```
1. 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

```
1. CREATE PROCEDURE usp_comp1
2. (@c1_Cno varchar(10), @c2_Cno varchar(10))
3. AS
4. DECLARE @c1_avgGrade float, @c2_avgGrade float;
5. SELECT @c1_avgGrade = AVG(Grade)
6. FROM SC
7. WHERE Cno = @c1_Cno;
8. SELECT @c2_avgGrade = AVG(Grade)
9. FROM SC
10. WHERE Cno = @c2_Cno;
11. IF @c1_avgGrade > @c2_avgGrade
12.   PRINT @c1_Cno + '课程的平均分高'
13. ELSE
14.   PRINT @c2_Cno + '课程的平均分高'
15. RETURN;
```

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。

```
1. CREATE PROCEDURE usp_comp_age2
2. (@s1_Sno varchar(10), @s2_Sno varchar(10))
3. AS
4. DECLARE @s1_age int, @s2_age int;
5. DECLARE @s1_name varchar(10), @s2_name varchar(10);
6. DECLARE @s1_sex char(2), @s2_sex char(2);
```

```
SELECT @s1_age = Sage, @s1_name = Sname, @s1_sex = Ssex
FROM Student
WHERE Sno = @s1_Sno;
SELECT @s2_age = Sage, @s2_name = Sname, @s2_sex = Ssex
FROM Student
WHERE Sno = @s2_Sno;
IF @s1_age > @s2_age
   SELECT @s1_name AS Sname, @s1_sex AS Ssex;
ELSE
   SELECT @s2_name AS Sname, @s2_sex AS Ssex;
RETURN;
```

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
(@c1_no varchar(10), @c2_no varchar(10))
AS
DECLARE @c1_avgGrade float, @c2_avgGrade float;
SELECT @c1_avgGrade = AVG(Grade)
FROM SC
WHERE Cno = @c1_no;
SELECT @c2_avgGrade = AVG(Grade)
FROM SC
WHERE Cno = @c2_no;
IF @c1_avgGrade > @c2_avgGrade
   SELECT Cname
   FROM Course
   WHERE Cno = @c1_no;
ELSE
   SELECT Cname
   FROM Course
   WHERE Cno = @c2_no;
RETURN;
```

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 varchar(10))
AS
WHILE (SELECT Grade
       FROM SC
       WHERE Sno = @sno
        AND Cno = '1001') > 58
BEGIN
```

```
  UPDATE SC
   SET Grade = Grade - 1
   WHERE Sno = @sno
    AND Cno = '1001';
END;
RETURN;
```

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
(@dept varchar(10))
AS
DECLARE @age int;
WHILE (SELECT MAX(Sage)
       FROM Student
       WHERE Sdept = @dept) < 28
BEGIN
  UPDATE Student
   SET Sage = Sage + 1
   WHERE Sdept = @dept;
END;
RETURN;
```

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 varchar(10))
AS
SELECT Sno AS 学号, Cno AS 课程号, Grade AS 成绩, CASE Grade / 10
  WHEN 9 THEN '优'
  WHEN 8 THEN '良'
  WHEN 7 THEN '中'
  WHEN 6 THEN '及格'
  ELSE '不及格'
END AS 等第
FROM SC
WHERE Cno = @cno;
RETURN;
```

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_disp2
(@sno varchar(10))
AS
DECLARE @grade_1001 int;
SELECT @grade_1001 = Grade
FROM SC
WHERE Sno = @sno
 AND Cno = '1001';
DECLARE @grade_1002 int;
SELECT @grade_1002 = Grade
FROM SC
WHERE Sno = @sno
 AND Cno = '1002';
DECLARE @grade_1003 int;
SELECT @grade_1003 = Grade
FROM SC
WHERE Sno = @sno
 AND Cno = '1003';
SELECT Sno AS 学号, Sname AS 姓名,
       @grade_1001 AS C1001 课程,
       @grade_1002 AS C1002 课程,
       @grade_1003 AS C1003 课程,
       (@grade_1001 + @grade_1002 + @grade_1003) / 3.0 AS 平均分
FROM Student
WHERE Sno = @sno;
RETURN;
```

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_disp3
(@dept varchar(10))
AS
CREATE TABLE #tmp (
  学号 char(4),
  姓名 char(10),
  C1001 课程 int NULL,
  C1002 课程 int NULL,
  C1003 课程 int NULL,
   平均分 float NULL
)
```


教务处制

```
DECLARE @sno char(4), @sname char(10),
        @nC1001 int, @nC1002 int, @nC1003 int;
DECLARE sno_cursor CURSOR FOR
SELECT Sno, Sname
FROM Student
WHERE Sdept = @dept;
OPEN sno_cursor;
FETCH NEXT FROM sno_cursor INTO @sno, @sname;
WHILE @@FETCH_STATUS = 0
BEGIN
  SELECT @nC1001 = Grade
  FROM SC
  WHERE Sno = @sno
   AND Cno = '1001';
  SELECT @nC1002 = Grade
  FROM SC
  WHERE Sno = @sno
   AND Cno = '1002';
  SELECT @nC1003 = Grade
  FROM SC
  WHERE Sno = @sno
   AND Cno = '1003';
  INSERT INTO #tmp
  VALUES (@sno, @sname, @nC1001, @nC1002, @nC1003,
          (@nC1001 + @nC1002 + @nC1003) / 3.0);
  FETCH NEXT FROM sno_cursor INTO @sno, @sname;
END;
CLOSE sno_cursor;
DEALLOCATE sno_cursor;
SELECT *
FROM #tmp;
DROP TABLE #tmp;
RETURN;
```

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_disp4
AS
CREATE TABLE #tmp (
```

```
    性别 char(4),
    C1001 人数 int,
    C1002 人数 int,
    C1003 人数 int,
    小计 int
)
DECLARE @sex char(2),
        @nC1001b int, @nC1002b int, @nC1003b int,
        @nC1001g int, @nC1002g int, @nC1003g int;
SELECT @nC1001b = COUNT(*)
FROM Student, SC
WHERE Student.Sno = SC.Sno
 AND Cno = '1001'
 AND Ssex = '男';
SELECT @nC1002b = COUNT(*)
FROM Student, SC
WHERE Student.Sno = SC.Sno
 AND Cno = '1002'
 AND Ssex = '男';
SELECT @nC1003b = COUNT(*)
FROM Student, SC
WHERE Student.Sno = SC.Sno
 AND Cno = '1003'
 AND Ssex = '男';
INSERT INTO #tmp
VALUES ('男', @nC1001b, @nC1002b, @nC1003b, @nC1001b + @nC1002b + @nC1003b);
SELECT @nC1001g = COUNT(*)
FROM Student, SC
WHERE Student.Sno = SC.Sno
 AND Cno = '1001'
 AND Ssex = '女';
SELECT @nC1002g = COUNT(*)
FROM Student, SC
WHERE Student.Sno = SC.Sno
 AND Cno = '1002'
 AND Ssex = '女';
SELECT @nC1003g = COUNT(*)
FROM Student, SC
WHERE Student.Sno = SC.Sno
 AND Cno = '1003'
 AND Ssex = '女';
INSERT INTO #tmp
VALUES ('女', @nC1001g, @nC1002g, @nC1003g, @nC1001g + @nC1002g + @nC1003g);
INSERT INTO #tmp
VALUES ('合计',
        @nC1001b + @nC1001g, @nC1002b + @nC1002g, @nC1003b + @nC1003g,
```

```
        @nC1001b + @nC1001g + @nC1002b + @nC1002g + @nC1003b + @nC1003g);
SELECT *
FROM #tmp;
DROP TABLE #tmp;
RETURN;
```

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 strftime
(@format varchar(255))
AS
-- get current year
DECLARE @year char(4);
SET @year = YEAR(GETDATE());
-- get current month
DECLARE @month char(2);
SET @month = RIGHT('0' + CAST(MONTH(GETDATE()) AS varchar), 2);
-- get current day
DECLARE @day char(2);
SET @day = RIGHT('0' + CAST(DAY(GETDATE()) AS varchar), 2);
-- get current hour
DECLARE @hour char(2);
SET @hour = RIGHT('0' + CAST(DATENAME(HH, GETDATE()) AS varchar), 2);
-- get current hour (12)
DECLARE @hour12 char(2);
IF CAST(@hour AS int) > 12
  SET @hour12 = RIGHT('0' + CAST((CAST(@hour AS int) - 12) AS varchar), 2)
ELSE
  SET @hour12 = @hour
-- get current minute
DECLARE @minute char(2);
SET @minute = RIGHT('0' + CAST(DATENAME(MI, GETDATE()) AS varchar), 2);
-- get current second
DECLARE @second char(2);
SET @second = RIGHT('0' + CAST(DATENAME(SS, GETDATE()) AS varchar), 2);
-- get current weekday
DECLARE @weekday char(6);
SET @weekday = DATENAME(WEEKDAY, GETDATE())
-- get current time format
SET @format = REPLACE(@format, '%Y', @year);
SET @format = REPLACE(@format, '%y', RIGHT(@year, 2));
SET @format = REPLACE(@format, '%m', @month);
SET @format = REPLACE(@format, '%d', @day);
SET @format = REPLACE(@format, '%H', @hour);
SET @format = REPLACE(@format, '%I', @hour12);
SET @format = REPLACE(@format, '%M', @minute);
```

```
39. SET @format = REPLACE(@format, '%S', @second);
40. SET @format = REPLACE(@format, '%w', @weekday);
41. SELECT @format as ftime;
42. RETURN;
```

示例：

```
1. strftime '%Y-%m-%d %H:%M:%S';
```

	ftime
1	2022-04-06 16:04:48

思考：何时需要存储过程？

存储过程作用是简化操作，将处理封装至一个易用的单元中。当处理步骤比较复杂时，应当使用存储过程。同时由于一致的封装，存储过程的使用减少了出错的可能性。另外，存储过程在创建时通常会进行编译（不同于视图），因此性能更好。然而，受限于不同 DBMS 之间存储过程语法的巨大差异，存储过程代码的移植性很差，因此大量使用存储过程会对 DBMS 之间的迁移造成巨大阻碍（事实上，目前政府机关主要还在使用 Oracle 而仍未迁移至 MySQL 就是受到了这方面的影响,这当然有一部分原因是 Oracle 更强大，但大多数能够迁移至 MySQL 而仍在使用 Oracle 的中小型项目显然是受到了 DBMS 之间语法不同的影响）。

我个人倾向于在服务器后端处理复杂的操作而非在数据库端直接使用存储过程，这一方面是为了更好的可移植性，另一方面也是为了更好的可读性，毕竟 SQL 语句要进行复杂的操作所需的代码量是相当冗余的，尽管 DBMS 都会对存储过程进行优化，但受限于 SQL 语法本身的问题不见得速度比服务器后端处理更快。即使网络请求速度遇到了瓶颈，也应当优先考虑优化后端与中间件，优化架构，例如增加 Redis 作为缓存，而非优先考虑建立存储过程。