数据库实验六-存储

> 邓语苏 2021级
> 计科 2024/06/04

实验目的

过本次实验，要求掌握变量定义，流程控制，存储过程，存储函数，游标等内容。

实验内容

1. 变量的声明和使用，掌握@@ERROR、@@ROWCOUNT、@@IDENTITY等全局变量的使用。
2. 使用BEGIN...END、IF...ELSE...、 WHILE...CONTINUE...BREAK...、CASE等流程控制语句。
3. 使用存储过程。
4. 使用系统函数和用户自定义函数。
5. 使用游标处理数据。

实验步骤

1. 用T-SQL语言完成1+2+3.....+100，并使用@@ERROE判断是否执行成功，如果成功则输出值，否则打印执行失败。

SQL

```
declare @sum int;
declare @error int;
declare @i int;

set @sum=0;
set @i=1;
while @i<=100
begin
set @sum=@sum+@i;
set @i=@i+1;
end

set @error=@@error;

if @error=0
begin
    print '计算成功，结果为:'+cast(@sum as varchar);--将int类型转化为
char类型
end
else
begin
    print '执行失败';
end
```

消息

```
计算成功，结果为:5050

完成时间: 2024-06-03T00:18:11.9509604+08:00
```

2. 更新STUDENTS表中sid为80000759500的学生的email为 ddff@sina.com，并通过@@ROWCOUNT判断是否有数据被更新，如果没有则打印警告。
    - `@@ROWCOUNT`返回上一个 `UPDATE` 语句影响的行数。如果 为0，表示没有记录被更新。

```sql
-- 更新 STUDENTS 表中的 email
UPDATE STUDENTS
SET email = 'ddff@sina.com'
WHERE sid = '80000759500';

-- 检查更新的行数
IF @@ROWCOUNT = 0
BEGIN
    PRINT '警告: 没有记录被更新';
END
ELSE
BEGIN
    PRINT '更新成功';
END;
```

3. 使用`IF…ELSE…`语句，查询STUDENTS表中学号为800007595的学生，如果学生存在，则输出学生的各科成绩，否则打印查无此人。

```sql
use School;
go

if exists(select 1 from students where sid='800007595')
begin
    select *
    from CHOICES
    where sid='800007595';
end
else
begin
    print '查无此人';
end
```

结果 消息

	no	sid	tid	cid	score
1	513921122	800007595	248213502	10035	93
2	527427359	800007595	224059084	10044	65

4. 计算1+2+3……+100中使用的是WHILE语句

```SQL
declare @sum int;
declare @error int;
declare @i int;

set @sum=0;
set @i=1;
while @i<=100
begin
set @sum=@sum+@i;
set @i=@i+1;
end

set @error=@@error;

if @error=0
begin
    print '计算成功，结果为:'+cast(@sum as varchar);--将int类型转化为
char类型
end
else
begin
    print '执行失败';
end
```

5. 使用CASE语句，查询学号为800007595所选择的课程号为10042的成绩，如果为80分或以上，打印优秀，如果在60—80分之间则打印及格，否则打印不及格。

SQL

```sql
DECLARE @Score INT;

-- 查询学号为800007595，课程号为10042的成绩
SELECT @Score = score
FROM CHOICES
WHERE sid = '800007595' AND cid = '10042';

-- 使用 CASE 语句根据成绩范围输出结果
DECLARE @Result VARCHAR(20);
SET @Result =
    CASE
        WHEN @Score >= 80 THEN '优秀'
        WHEN @Score >= 60 AND @Score < 80 THEN '及格'
        ELSE '不及格'
    END;

-- 打印结果
PRINT '成绩评定: ' + @Result;
```

	no	sid	tid	cid	score
1	11133	800007595	NULL	10042	90

消息

成绩评定: 优秀

6. 创建一个带输入和输出参数的存储过程，查询学生选修课程成绩，将分数低于60分的成绩改为60分，高于80分的成绩改为80分。输入参数为学生的学号，输出参数为提示信息，如果不学生不存在则参数值为查无此人，更改失败则为更改失败，更改成功则为更改成功。（提示：可使用事务机制）

- 创建存储过程

SQL

```sql
CREATE PROCEDURE Updatestudentscore @sid           CHAR(10),
                                    @ResultMessage NVARCHAR(50)
output
AS
  BEGIN
      BEGIN TRANSACTION

      DECLARE @Count INT

      SELECT @Count = Count(*)
      FROM   CHOICES
      WHERE  sid = @sid

      --若查找不到
      IF @Count = 0
        BEGIN
            SET @ResultMessage='查无此人'

            ROLLBACK TRANSACTION--回滚状态

            RETURN
        END

      --更新成绩
      UPDATE CHOICES
      SET    score = CASE
                       WHEN score < 60 THEN 60
                       WHEN score > 80 THEN 80
                       ELSE score
                     END
      WHERE  sid = @sid

      -- 检查更新是否成功
      IF @@ROWCOUNT = 0
        BEGIN
            -- 更新失败
            SET @ResultMessage = '更改失败'

            ROLLBACK TRANSACTION
        END
      ELSE
        BEGIN
```

```
            -- 更新成功
            SET @ResultMessage = '更改成功'

            COMMIT TRANSACTION
        END
    END
```

- 执行存储过程

```sql
DECLARE @ResultMessage NVARCHAR(50);

-- 执行存储过程
EXEC Updatestudentscore
  @sid = '12345      ',
  @ResultMessage = @ResultMessage OUTPUT;

-- 输出结果信息
SELECT @ResultMessage AS ResultMessage;
```

- 执行前

2	999999	12345	12345	12345	99

- 执行后

结果 消息

	ResultMessage
1	更改成功

2	999999	12345	12345	12345	80

7. 查询学号为800007595的学生的email转换成大写输出，并查询其选修课程名的前三个字符。提示：使用UPPER（）函数和SUBSTRING（）函数。

```sql
select UPPER(email) as UppercaseEmail,SUBSTRING(cname,1,3) as
CourseNamePrefix
from STUDENTS as S,COURSES as C,CHOICES as CH
where S.sid = '800007595' and CH.sid=S.sid and CH.cid=C.cid
```

结果 消息

	UppercaseEmail	CourseNamePrefix
1	CR8G@ZRVGT.EDU	c

8.

* 创建标量值函数，要求根据输入的学生学号参数，返回学生的选课的平均成绩。

SQL

```sql
CREATE FUNCTION dbo.GetAverageScore
(
    @sid CHAR(10)
)
RETURNS FLOAT
AS
BEGIN
    DECLARE @AverageScore FLOAT;

    SELECT @AverageScore = AVG(score)
    FROM CHOICES
    WHERE sid = @Sid;

    RETURN @AverageScore;
END;
```

SQL

```sql
--执行函数
SELECT dbo.GetAverageScore('800007595') AS AverageScore;
```

结果 消息

	AverageScore
1	82

- 创建内联表值函数，要求根据学生真实姓名显示其所有选修课程名和成绩。

SQL

```sql
CREATE FUNCTION dbo.GetStudentCoursesAndScores
(
    @StudentName NVARCHAR(50)
)
RETURNS TABLE
AS
RETURN
(
    SELECT
        C.cname,
        Ch.Score
    FROM
        Students S
    JOIN
        Choices Ch ON S.sid = Ch.sid
    JOIN
        Courses C ON Ch.cid = C.cid
    WHERE
        S.sname = @StudentName
);
```

SQL

```sql
-- 执行函数
SELECT * FROM dbo.GetStudentCoursesAndScores('xcikvufj
');
```

```
        ![[Pasted image 20240603233908.png]]
```

- 创建多语句内联表值函数，要求根据课程名称查询所有选修些课程的学生姓名和分数。

SQL

```sql
CREATE FUNCTION dbo.GetStudentsByCourse
(
    @CourseName NVARCHAR(100)
)
RETURNS @tb_scores TABLE
(
    StudentName NVARCHAR(50),
    Score INT
)
AS
BEGIN
    INSERT INTO @tb_scores
    SELECT
        S.sname,
        Ch.Score
    FROM
        Courses C
    JOIN
        Choices Ch ON C.cid = Ch.cid
    JOIN
        Students S ON Ch.sid = S.sid
    WHERE
        C.cname = @CourseName;

    RETURN;
END;
```

查看当前CHOICES表中选课最多的课程名称。

SQL

```sql
SELECT C.cname, COUNT(CH.sid) AS EnrollmentCount
FROM COURSES AS C
JOIN CHOICES AS CH ON C.cid = CH.cid
GROUP BY C.cname;
```

结果 消息

	cname	EnrollmentCount
1	algorithm	5992
2	architectonics	6042
3	c	1
4	c++	6031
5	compiling principle	5955
6	computer graphics	5975
7	computer interface	5955
8	computer network	5866
9	data structure	5985
10	design pattern	6090
11	real-time system	6039
12	software engineering	6027
13	tcp/ip protocol	5990

执行函数

```SQL
SELECT * FROM dbo.GetStudentsByCourse('algorithm
');
```

结果 消息

	StudentName	Score
1	tkbzqduq	65
2	yahvv	82
3	tongd	95
4	tulnfr	72
5	jkrdzdh	68
6	iaxeybfd	68
7	erdpcn	84
8	ctvxn	60
9	ieiojxooe	78
10	hdcliwkp	83
11	rsbrnhzbo	66
12	qukon	95
13	jrmogw	70
14	mmdds	66
15	covlmj	66

分析存储过程和存储函数的异同点：

- 相同点
  1. 都可以重复被调用
  2. 都可以包含事务控制逻辑
- 不同点

1. 返回值不同。
   存储过程可以不返回值；存储函数必须有明确的返回值
2. 用途不同
   存储过程一般作为一个集成的sql语句使用。存储函数主要用于计算并返回值，可以在查询语句中使用
3. 数据修改
   存储过程可以包含数据修改操作。存储函数一般不对数据进行修改，而是对其进行计算，将结果返回。

9.

- 定义一个游标，将学号为800007595的学生的选修课程名和成绩逐行打印出来。
  输出可知学号为800007595的学生的选修课程仅有一门，无法展现游标的作用。因此先找到CHOICES表中选课门数最多的学生

```SQL
SELECT TOP 1 Ch.sid,COUNT(*) as COURSECOUNT
FROM Courses AS C
JOIN Choices AS Ch ON C.cid = Ch.cid
GROUP BY Ch.sid
ORDER BY COUNT(*) DESC;
```

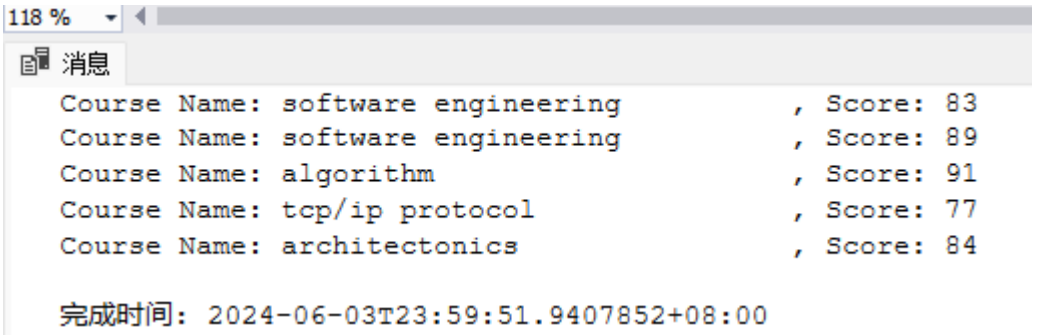
结果 消息

	sid	COURSECOUNT
1	859761946	5

打印其选课名和成绩

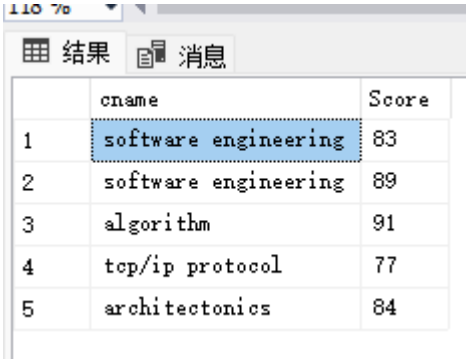
SQL

```sql
DECLARE @CourseName NVARCHAR(100), @Score INT;

-- 定义游标
DECLARE course_cursor CURSOR FOR
SELECT C.cname, Ch.Score
FROM Courses AS C
JOIN Choices AS Ch ON C.cid = Ch.cid
WHERE Ch.sid = '822863200 ';

-- 打开游标
OPEN course_cursor;

-- 获取第一行数据
FETCH NEXT FROM course_cursor INTO @CourseName, @Score;

-- 遍历游标中的每一行
WHILE @@FETCH_STATUS = 0
BEGIN
    -- 打印课程名和成绩
    PRINT 'Course Name: ' + @CourseName + ', Score: ' + CAST(@Score
AS NVARCHAR(10));

    -- 获取下一行数据
    FETCH NEXT FROM course_cursor INTO @CourseName, @Score;
END;

-- 关闭游标
CLOSE course_cursor;

-- 释放游标资源
DEALLOCATE course_cursor;
```



验证结果

```SQL
SELECT C.cname, Ch.Score
FROM Courses AS C
JOIN Choices AS Ch ON C.cid = Ch.cid
WHERE Ch.sid = '859761946 ';
```



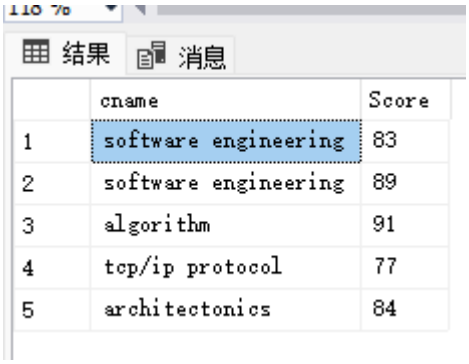
	cname	Score
1	software engineering	83
2	software engineering	89
3	algorithm	91
4	tcp/ip protocol	77
5	architectonics	84

- 定义一个游标，将学号为800007595的学生的第二门选修课程成绩（成绩降序排列）改为75分。

SQL

```sql
DECLARE @CourseID INT, @Score INT;
DECLARE @Counter INT = 0;

-- 定义游标
DECLARE second_course_cursor CURSOR FOR
SELECT cid, Score
FROM Choices
WHERE sid = '859761946 '
ORDER BY Score DESC;

-- 打开游标
OPEN second_course_cursor;

-- 获取第一行数据
FETCH NEXT FROM second_course_cursor INTO @CourseID, @Score;

-- 遍历游标中的每一行
WHILE @@FETCH_STATUS = 0
BEGIN
    SET @Counter = @Counter + 1;

    -- 如果是第二行数据
    IF @Counter = 2
    BEGIN
        -- 更新成绩为75分
        UPDATE Choices
        SET Score = 75
        WHERE sid = '859761946 ' AND cid = @CourseID;
        BREAK;
    END;

    -- 获取下一行数据
    FETCH NEXT FROM second_course_cursor INTO @CourseID, @Score;
END;

-- 关闭游标
CLOSE second_course_cursor;

-- 释放游标资源
DEALLOCATE second_course_cursor;
```

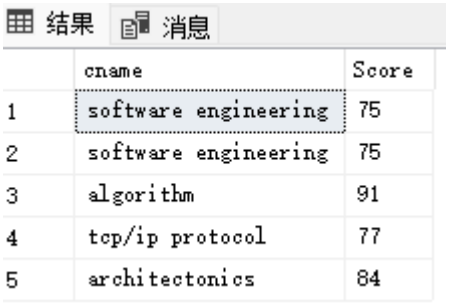
同样使用选课门数最多的学生来测试
验证结果
执行前



结果 消息

	cname	Score
1	software engineering	83
2	software engineering	89
3	algorithm	91
4	tcp/ip protocol	77
5	architectonics	84

执行后



结果 消息

	cname	Score
1	software engineering	75
2	software engineering	75
3	algorithm	91
4	tcp/ip protocol	77
5	architectonics	84

- 创建一个没有唯一索引的表，定义一个游标，删除其中一条记录，查看是否允许删除。
  创建无唯一索引的表

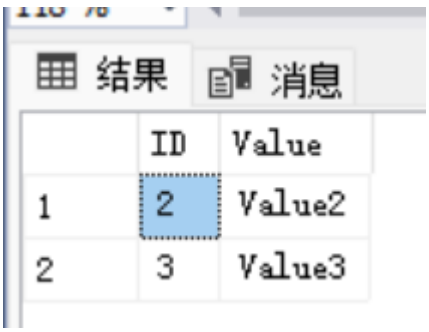
```SQL
CREATE TABLE TestTable (
    ID INT,
    Value NVARCHAR(50)
);

-- 插入一些数据
INSERT INTO TestTable (ID, Value) VALUES (1, 'Value1');
INSERT INTO TestTable (ID, Value) VALUES (2, 'Value2');
INSERT INTO TestTable (ID, Value) VALUES (3, 'Value3');
```

定义游标，删除其中一条记录

SQL

```sql
DECLARE @ID INT, @Value NVARCHAR(50);

-- 定义游标
DECLARE delete_cursor CURSOR FOR
SELECT ID, Value
FROM TestTable;

-- 打开游标
OPEN delete_cursor;

-- 获取第一行数据
FETCH NEXT FROM delete_cursor INTO @ID, @Value;

-- 删除当前游标指向的记录
DELETE FROM TestTable
WHERE CURRENT OF delete_cursor;

-- 查看是否允许删除
IF @@ROWCOUNT = 1
BEGIN
    PRINT 'Record deleted successfully';
END
ELSE
BEGIN
    PRINT 'Failed to delete record';
END;

-- 关闭游标
CLOSE delete_cursor;

-- 释放游标资源
DEALLOCATE delete_cursor;
```

验证结果



允许删除。