

## 实验六 过程语言

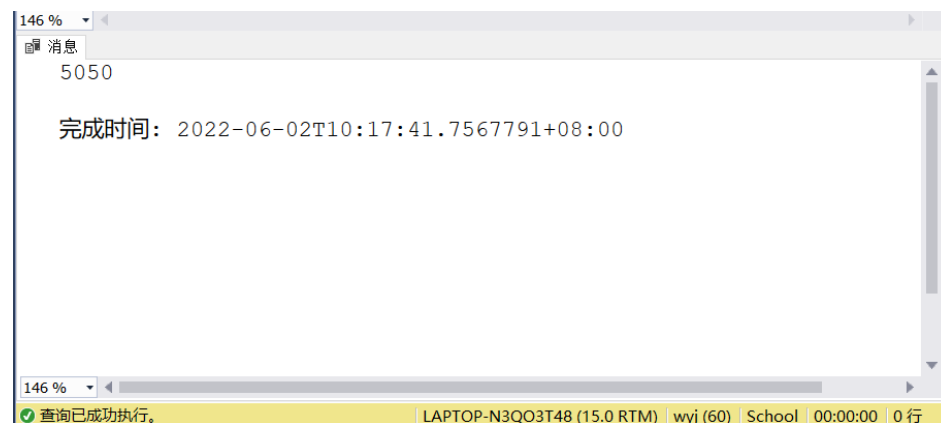
实验目的：掌握变量定义、流程控制、存储过程、存储函数、游标等内容

实验环境：WINDOWS10、SQL SERVER 2019

实验内容：

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

```
declare @sum int
declare @i int
set @sum = 0
set @i = 1
while(@i <= 100)
begin
    set @sum = @sum + @i
    set @i = @i + 1
end
if @@ERROR = 0
    print @sum
else
    print '执行失败'
```



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。

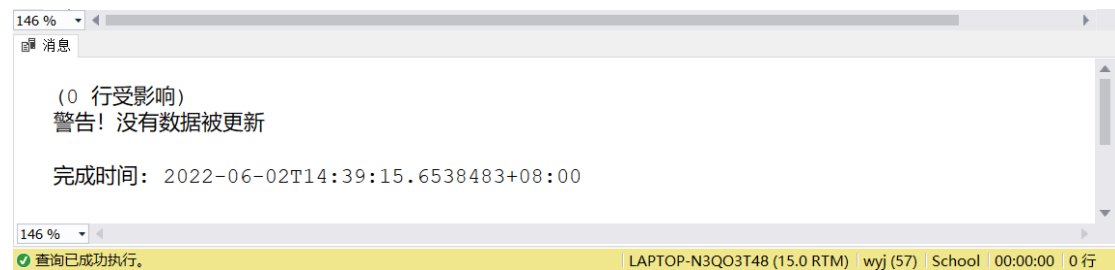
提示：

```
IF @@ROWCOUNT = 0
```

```
PRINT '警告！没有数据被更新'
```

```
update STUDENTS set email = 'ddff@sina.com' where sid = '80000759500'
```

```
if @@ROWCOUNT = 0
    print '警告！没有数据被更新'
```

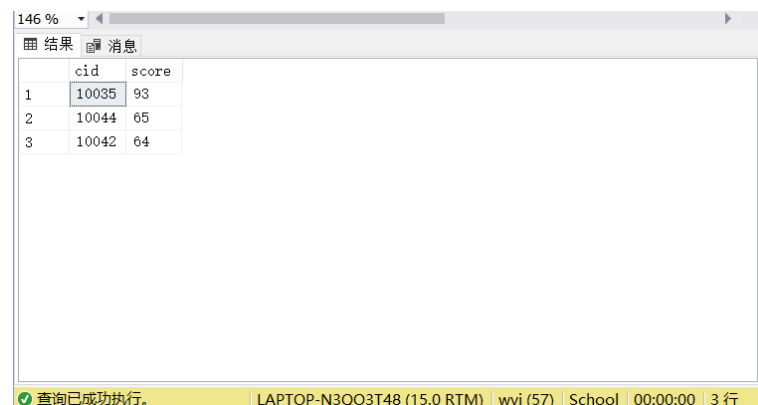


3) 使用 IF…ELSE…语句，查询 STUDENTS 表中学号为 800007595 的学生，如果学生存在，则输出学生的各科成绩，否则打印查无此人。

提示：USE School

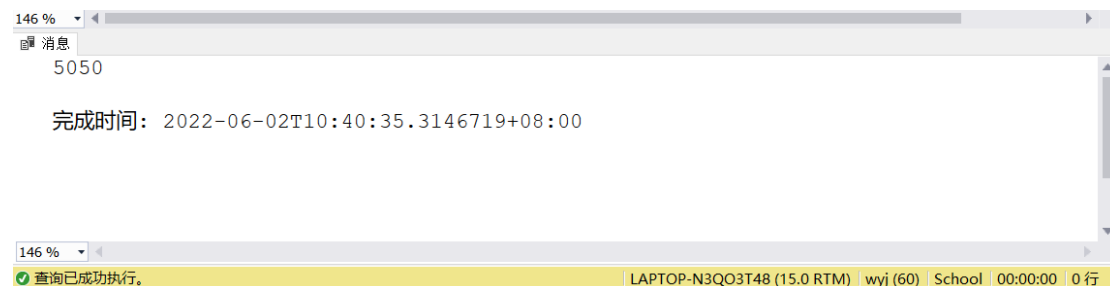
```
IF(EXISTS (...))
    BEGIN
        ...
    END
ELSE
    PRINT ...
```

```
use School
if exists(select * from STUDENTS where sid = '800007595')
begin
    select cid,score
    from CHOICES
    where CHOICES.sid = '800007595'
end
else
    print '查无此人'
```



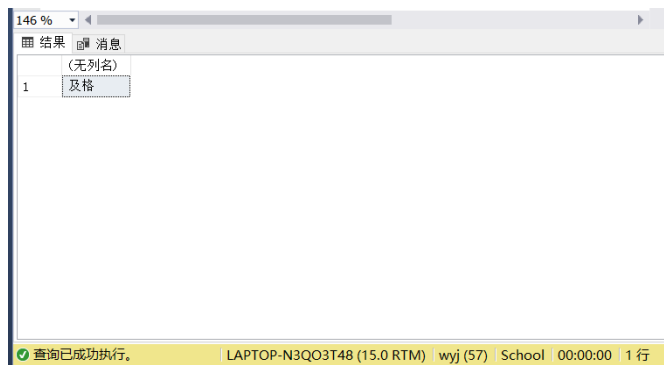
#### 4) 使用 WHILE 语句计算 1+2+3……+100

```
declare @sum int
declare @i int
set @sum = 0
set @i = 1
while(@i <= 100)
begin
    set @sum = @sum + @i
    set @i = @i + 1
end
if @@ERROR = 0
    print @sum
else
    print '执行失败'
```



#### 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。

```
select
    case
        when score >= 80 then '优秀'
        when score >= 60 and score < 80 then '及格'
        when score < 60 then '不及格'
    end
from CHOICES
where sid = '800007595' and cid = '10042';
```



6) 使用 T-SQL 命令 CREATE PROC 语句可创建存储过程，基本语法格式如下：

CREATE PROC[UDURE] <>

( [ { @参数 数据类型 } [ = 默认值 ] [ OUTPUT ] ] [ , ...n ] )

[ WITH { RECOMPILE | ENCRYPTION | RECOMPILE, ENCRYPTION } ]

AS

<SQL 语句>[...n]

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）

执行已存在的存储过程使用语句格式如下：

[EXECUTE] 存储过程名 [输入参数值]

执行创建的存储过程，通过输出参数分析执行结果。

定义存储过程：

```
create proc UpdateScore
    @sid char(10), @msg char(8) output
as
begin
    set @msg = '更改成功'
    if(exists(select * from CHOICES where sid = @sid))
    begin
        update CHOICES set score = 60 where sid = @sid and score < 60
        update CHOICES set score = 80 where sid = @sid and score > 80
        if(@@ERROR = 0)
            set @msg = '更改成功'
        else
            set @msg = '更改失败'
```

```

end
else
    set @msg = '查无此人'
end

```

执行存储过程：

```

declare @printMsg char(8)
execute UpdateScore '800007595', @printMsg output
print @printMsg
select * from CHOICES where sid = '800007595';

```

执行存储过程前：

no	sid	tid	cid	score
1	513921122	800007595	248213502	10035 93
2	527427359	800007595	224059084	10044 65
3	562145282	800007595	233622158	10042 64

执行存储过程后：

no	sid	tid	cid	score
1	513921122	800007595	248213502	10035 80
2	527427359	800007595	224059084	10044 65
3	562145282	800007595	233622158	10042 64

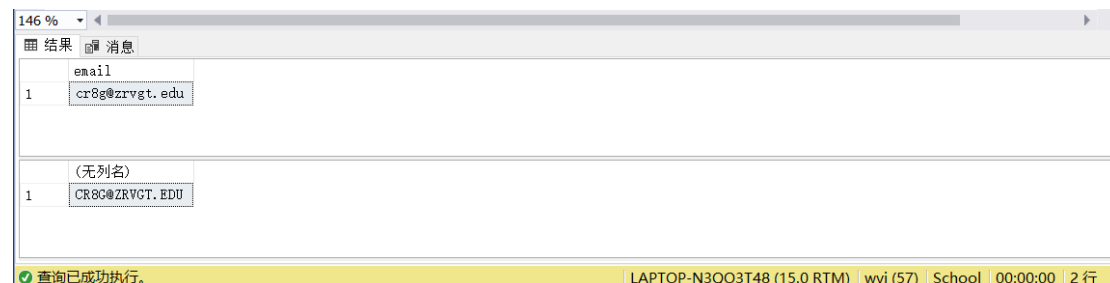
(0 行受影响)  
 (1 行受影响)  
 更改成功  
 完成时间: 2022-06-02T16:06:46.9513403+08:00

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。

①查询学号为 800007595 的学生的 email 转换成大写输出：

```
select email from STUDENTS where sid = '800007595';  
select UPPER(email) from STUDENTS where sid = '800007595';
```

结果如下：



	email
1	cr8g@zrvgt.edu

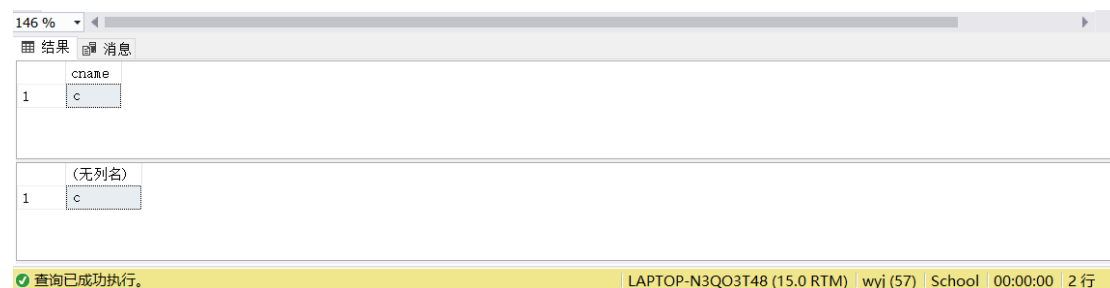
  

	(无列名)
1	CR8G@ZRVGT.EDU

查询已成功执行。 | LAPTOP-N3QO3T48 (15.0 RTM) | wyj (57) | School | 00:00:00 | 2 行

②查询其选修课程名的前三个字符：

```
select COURSES.cname from COURSES, CHOICES  
where CHOICES.sid = '800007595' and COURSES.cid = CHOICES.cid;  
select SUBSTRING(COURSES.cname, 1, 3) from COURSES, CHOICES  
where CHOICES.sid = '800007595' and COURSES.cid = CHOICES.cid;
```



	cname
1	c

	(无列名)
1	c

查询已成功执行。 | LAPTOP-N3QO3T48 (15.0 RTM) | wyj (57) | School | 00:00:00 | 2 行

8) 用户自定义函数分为：标量值函数、内联表值函数、多语句表值函数。

实验要求：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。

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。

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。

提示：CREATE FUNCTION <函数名>

(<参数>)

RETURNS @tb\_scores TABLE(<返回表属性>)

AS

BEGIN

```

INSERT @tb_scores
SELECT...
RETURN
END

```

执行：SELECT \* FROM 函数名（' <课程名>'）

① 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。

创建函数：

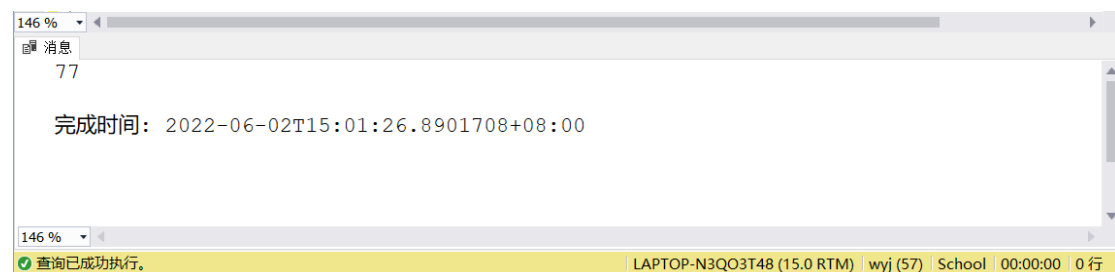
```

create function AvgScore(@sid char(10))
returns int
as
begin
    declare @avgscore int
    select @avgscore = (select AVG(score) from CHOICES where sid =
@sid)
    return @avgscore
end

```

执行函数：

```
print dbo.AvgScore('826310502');
```



② 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。

创建函数：

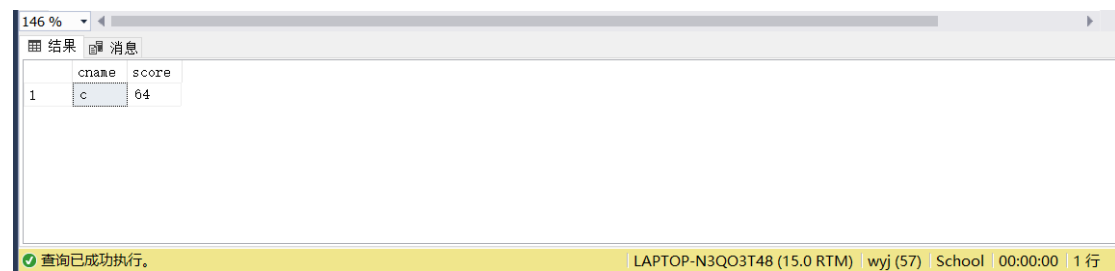
```

create function query(@sname char(30))
returns table
as
return (
    select COURSES.cname, CHOICES.score
    from COURSES, CHOICES
    where CHOICES.sid = (select sid from STUDENTS where sname =
@sname)
    and COURSES.cid = CHOICES.cid
)

```

执行函数：

```
select * from query('uxqbkjn');
```



	cname	score
1	c	64

查询已成功执行。 | LAPTOP-N3QO3T48 (15.0 RTM) | wyj (57) | School | 00:00:00 | 1 行

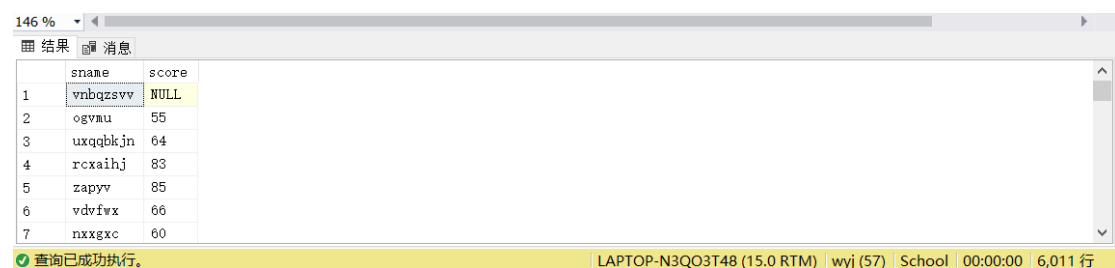
③ 创建多语句内联表值函数，要求根据课程名称查询所有选修该课程的学生姓名和分数。

创建函数：

```
create function CourseQuery(@cname char(10))
returns @NameAndScore table(sname char(30), score int)
as
begin
    insert @NameAndScore select STUDENTS.sname, CHOICES.score
    from STUDENTS, CHOICES, COURSES
    where COURSES.cname=@cname
        and COURSES.cid=CHOICES.cid and CHOICES.sid=STUDENTS.sid
    return
end
```

执行函数：

```
select * from CourseQuery('c');
```



	sname	score
1	vnbqzsvv	NULL
2	ogvau	55
3	uxqbkjn	64
4	rcxaihj	83
5	zapyv	85
6	vdvfwx	66
7	nxxgxc	60

查询已成功执行。 | LAPTOP-N3QO3T48 (15.0 RTM) | wyj (57) | School | 00:00:00 | 6,011 行

④分析存储过程和存储函数的异同点

I) 存储过程和存储函数的相同点：它们都是 SQL 扩展的过程化功能，且基本结构都是块。两者在写法结构上类似，都可以提高运行效率。经编译和优化后，它们都存储在数据库服务器中，使用的时候调用即可。

II) 存储过程和存储函数的不同点：

- 编程上的限制不同。存储过程的限制比较少，而存储函数的限制比较多，比如不能使用临时表等。
- 返回值个数不同。存储过程可以有一个或多个返回值，也可以没有返回值。存储函数有且只有一个返回值。



- 执行方式不同。存储过程作为一个独立的部分执行。存储函数可以嵌入到 SQL 中，作为查询语句的一个部分来调用。
- 应用场景不同。存储过程用于实现比较复杂的功能。而存储函数实现的功能针对性比较强。

9) 游标不同于查询语句，查询语句只能对整个结果集进行同一种操作，而游标允许定位在结果集的特定行，从结果集的当前位置检索一行或多行，支持对结果集中当前位置的行进行数据修改，为其他用户对显示在结果集中的数据库数据所做的更改提供不同级别的可见性支持，提供脚本、存储过程和触发器中用于访问结果集中数据的 T-SQL 语句。

使用游标必须按照下面顺序：声明游标，打开游标，读取游标中的数据，关闭游标，释放游标。

游标声明：DECLARE <游标名> [INSENSITIVE] [SCROLL] CURSOR

FOR <SELECT 语句>

[FOR { READ ONLY | UPDATE [ OF <表名列表>]}]

打开游标：OPEN { { [GLOBAL] <游标名> } | <游标变量名> }

提取数据：FETCH

[ [ NEXT | PRIOR | FIRST | LAST | ABSOLUTE { n | @整型变量 } | RELATIVE { n | @整型变量 } ] FROM ]

{ { [GLOBAL] <游标名> } | @游标变量名 }

[ INTO @变量名列表 ]

关闭游标：CLOSE { { [GLOBAL] <游标名> } | <游标变量名> }

释放游标：DEALLOCATE { { [GLOBAL] <游标名> } | <游标变量名> }

实验要求：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。

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。

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。

① 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。

```

declare @cname char(30)
declare @score int

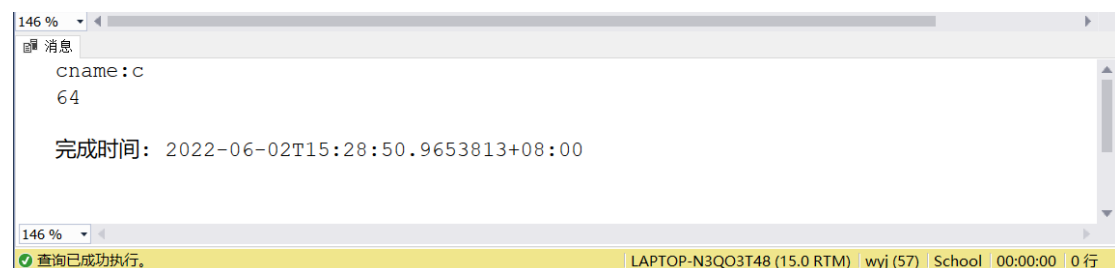
declare printCNameAndScore cursor
for
select COURSES.cname, CHOICES.score
from COURSES, CHOICES
where CHOICES.sid = '800007595' and COURSES.cid = CHOICES.cid

open printCNameAndScore
fetch next from printCNameAndScore into @cname, @score

while @@FETCH_STATUS = 0
begin
    print 'cname:' + @cname
    print @score
    fetch next from printCNameAndScore into @cname, @score
end

close printCNameAndScore
deallocate printCNameAndScore

```



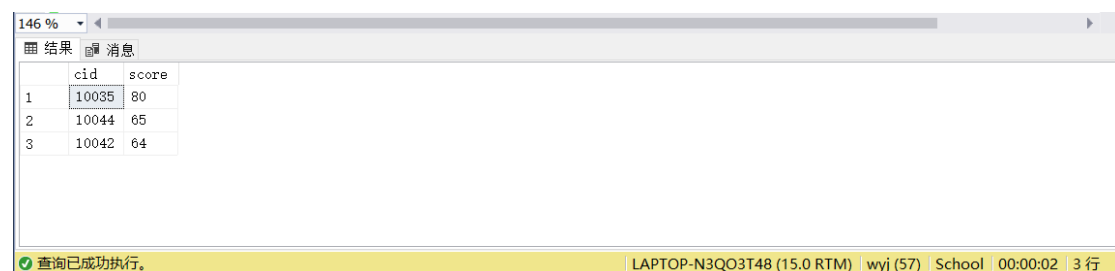
② 定义一个游标，将学号为 800007595 的学生的第二门选修课程成绩（成绩降序排列）改为 75 分。

在修改前查看：

```

select cid, score from choices where sid='800007595'
order by score desc

```



可知学号为 800007595 的学生的第二门选修课程成绩（成绩降序排列）为 65 分。

定义游标:

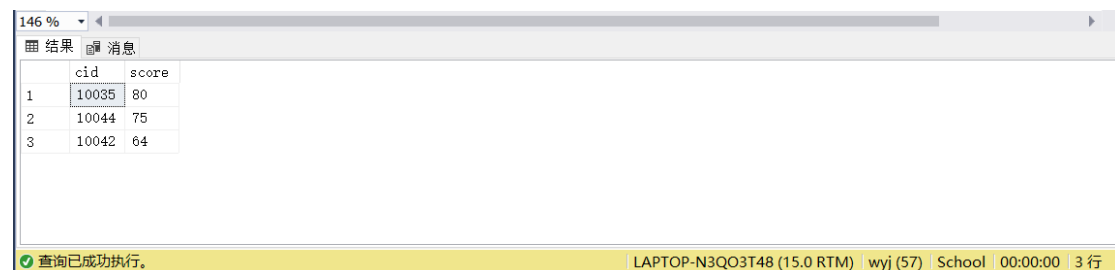
```
declare @no int
declare @score int
declare @i int
set @i = 1
declare updateSecondScore cursor
for
select no, score
from CHOICES
where CHOICES.sid = '800007595'
order by score desc;

open updateSecondScore
fetch next from updateSecondScore into @no, @score

while @@FETCH_STATUS = 0
begin
    if @i = 2
        update CHOICES set score = 75 where no = @no
    fetch next from updateSecondScore into @no, @score
    set @i = @i + 1
end

close updateSecondScore
deallocate updateSecondScore
```

修改之后查看:



	cid	score
1	10035	80
2	10044	75
3	10042	64

可以看到原本的 65 分被修改成了 75 分。

③ 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。

创建一个表 Test:

```
create table Test
(no char(5),
value char(10));
```

```
insert into Test
values('10001', 'abc'),
('10002', 'abc'),
('10003', 'ghj');
```

```
select * from Test
```

结果如下:

dbo.Test

- 列
  - no (char(5), null)
  - value (char(10), null)
- 键
- 约束
- 触发器
- 索引
- 统计信息

	no	value
1	10001	abc
2	10002	abc
3	10003	ghj

查询已成功执行。 | LAPTOP-N3QO3T48 (15.0 RTM) | wyj (57) | School | 00:00:00 | 3 行

定义游标, 尝试删除记录:

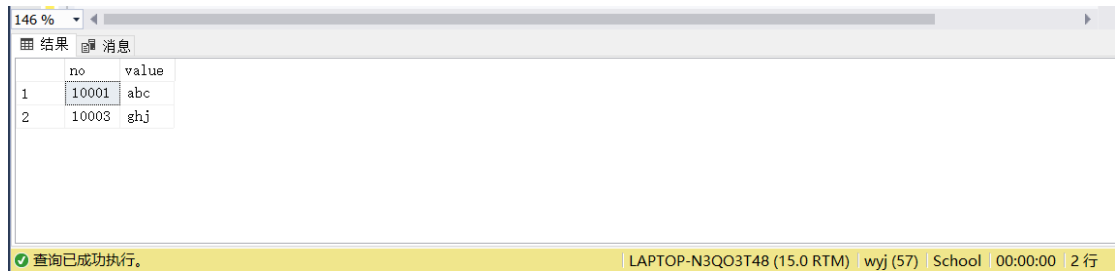
```
declare @no char(5)
declare @value char(10)
```

```
declare deleteValue cursor
for
select * from Test;
```

```
open deleteValue
fetch next from deleteValue into @no, @value
while @@FETCH_STATUS = 0
begin
    if @no = '10002'
        delete from Test where no = @no
    fetch next from deleteValue into @no, @value
end
```

```
close deleteValue
deallocate deleteValue
```

删除后查看表 Test:



	no	value
1	10001	abc
2	10003	ghj

可知允许删除，no 为 ‘10002’ 的记录已经被成功删除。

### 实验总结：

通过本次实验，我熟悉了变量定义、流程控制、存储过程、存储函数、游标等内容，对这些课上学到的理论知识加以应用，获得了更具象的理解。本次实验主要难点在于过程语言的语法和格式，熟练掌握变量定义和定义之后的使用，就可以完成本次实验的内容。