

T-SQL编程

Email: 18442056@QQ.com

- 回顾与复习

- 1. 算术运算符

- 算术运算符对两个表达式执行数学运算，参与运算的表达式必须是数值数据类型或能够进行算术运算的其它数据类型。SQL Server 2008提供的算术运算符如下表

运算符	名称	语法
+	加	Expression1 + Expression2
-	减	Expression1 - Expression2
*	乘	Expression1 * Expression2
/	除	Expression1 / Expression2
%	取余	Expression1 % Expression2

- 2. 赋值运算符
- 等号（=）是唯一的 Transact-SQL 赋值运算符。
- 3. 字符串连接运算符
- 加号（+）是字符串串联运算符，可以用它将字符串串联起来。其他所有字符串操作都使用字符串函数进行处理。

4. 比较运算符

比较运算符用来比较两个表达式值之间的大小关系，可以用于除了 **text**、**ntext** 或 **image** 数据类型之外的所有数据类型。运算的结果为 **True**、**False**，通常用来构造条件表达式。右表列出了 Transact-SQL 的比较运算符。

运算符	名称	语法
=	等于	Expression1 = Expression2
>	大于	Expression1 > Expression2
>=	大于等于	Expression1 >= Expression2
<	小于	Expression1 < Expression2
<=	小于等于	Expression1 <= Expression2
<>或!=	不等于	Expression1 <> Expression2
!>	不大于	Expression1 !> Expression2
!<	不小于	Expression1 !< Expression2

逻辑运算符

- 逻辑运算符用来对多个条件进行运算，运算的结果为True或False，通常用来表示复杂的条件表达式。下表列出了Transact-SQL 的逻辑运算符。

运算符	说明	语法
Not	对表达式的值取反	Not Expression
And	与，如果表达式的值都为True，结果为True，否则为False	Expression1 and Expression2
Or	或，如果表达式的值都为False，结果为False，否则为True	Expression1 or Expression2
Between...and	如果操作数在某个范围内，结果为True	Expression between A and B
In	如果操作数等于值列表中的任何一个，结果为True	Expression in (值表或子查询)
Like	如果字符型操作数与某个模式匹配，结果为True	Expression1 like Expression2
Exists	如果子查询结果不空，结果为True	Exists (子查询)
Any或Some	如果操作数与一系列值中的任何一个比较结果为True，结果为True	Expression >any(值表或子查询)
All	如果操作数与一系列值中所有值的比较结果为True，结果为True	Expression <all(值表或子查询)

变量的定义

declare @变量名 数据类型

为变量赋值:

set @变量名 = 值 | 表达式

select @变量名 = 值 | 表达式

函数

- 数学函数
- 字符串函数
- 时间函数
- 类型转换函数convert和cast

- sql server 不仅可以使使用T-SQL利用结构化查询语句实现基于表数据的操作，还可以使用类似高级程序语言的流控、函数来实现基于数据库表数据过程控制。

内容1：批处理和注释

1. 批处理

在程序中，我们可以使用GO语句将多条SQL语句进行分隔，两个GO之间的SQL语句可作为一个批处理。在一个批处理中可以包含一条或多条T-SQL语句，成为一个语句组。批处理中的语句是同时从应用程序发送到 SQL Server服务器执行，加快了执行速度。在书写批处理语句时，需要使用GO语句作为批处理命令的结束标志。

2. 注释

注释是对程序的说明解释。一般使用注释对程序进行说明，例如变量的含义、程序的功能描述、基本思想等，增加程序的可读性。SQL Server 2008提供了两类注释符。

- : 单行注释，注释语句写在注释符--的后面，以最近的回车符作为注释的结束。
- /*...*/: 多行注释，“/*”在开头，“*/”在结尾。

内容2：流程控制语句

1. Begin...End语句

Begin...End语句用于将多条Transact-SQL语句组成一个语句块，作为一个整体来执行。Begin...End语句的语法格式为：

Begin

SQL语句

End

说明：

Begin...End语句块中至少要包含一条SQL语句。

- Begin...End语句块常用在If条件语句和While循环语句中
- Begin...End语句允许嵌套。

2. If...Else条件语句

语法格式为：

IF 表达式

{

SQL语句

}

ELSE

{

SQL语句

}

案例1

--判断变量中的值是正数、负数或

```
declare @var int
```

```
set @var = 0
```

```
if @var > 0
```

```
begin
```

```
    print '正数'
```

```
end
```

```
else if @var < 0
```

```
begin
```

```
    print '负数'
```

```
end
```

```
else
```

```
    print '为0'
```

案例2

- -- 查询《面向对象程序设计C#》的平均考试成绩，如果大于70分，则为理想，如果小于70分则不理想

```
if (select AVG(score) from V_StuScore_Statistics where
    courseName = '面向对象程序设计C#') >=70
```

```
begin
```

```
    print '该课程成绩较好'
```

```
    ○ ○ ○
```

3. Case分支语句

(1) 简单Case表达式

简单Case表达式将一个测试表达式与一组简单表达式进行比较，如果某个简单表达式与测试表达式的值相等，则返回相应结果表达式的值。简单Case表达式的语法格式如下：

Case 输入表达式

When 比较条件值 Then

输出结果

- [...n]
- [Else 其它输出结果]
- End

案例1

```
declare @sex varchar(1)
```

```
set @sex = '1'
```

```
select CASE @sex
```

```
WHEN '1' THEN '男'
```

```
WHEN '2' THEN '女'
```

```
ELSE '其他' END
```

（2）搜索Case表达式

搜索Case表达式中，各个When子句后都是布尔表达式。搜索Case表达式的语法格式如下：

Case

When 条件表达式 Then 输出结果

[...n]

[Else 其它输出结果]

End

案例2

```
declare @sex varchar(1)
set @sex = '1'

select CASE
WHEN @sex = '1' THEN '男'
WHEN @sex = '2' THEN '女'
ELSE '其他' END
```

案例3（对查询结果的替换）

- 检索学生信息，并从学生的电话号码的前3位判读出学生是属于“移动用户”，“电信用户”，“联通用户”。
- 比较简单case表达式和搜索case表达式的操作区别，并找出特点。

- 4. While循环语句

- While语句用来实现重复执行语句或语句块，当指定的条件为真时，重复执行循环语句。具体的语法格式如下：

While 条件表达式

{sql语句}

[Break]

{sql语句}

[Continue]

{sql语句}

案例1

- 使用循环输出1到10

```
declare @i int
set @i = 1
while @i < 10
begin
    select @i
    set @i = @i + 1
end
```

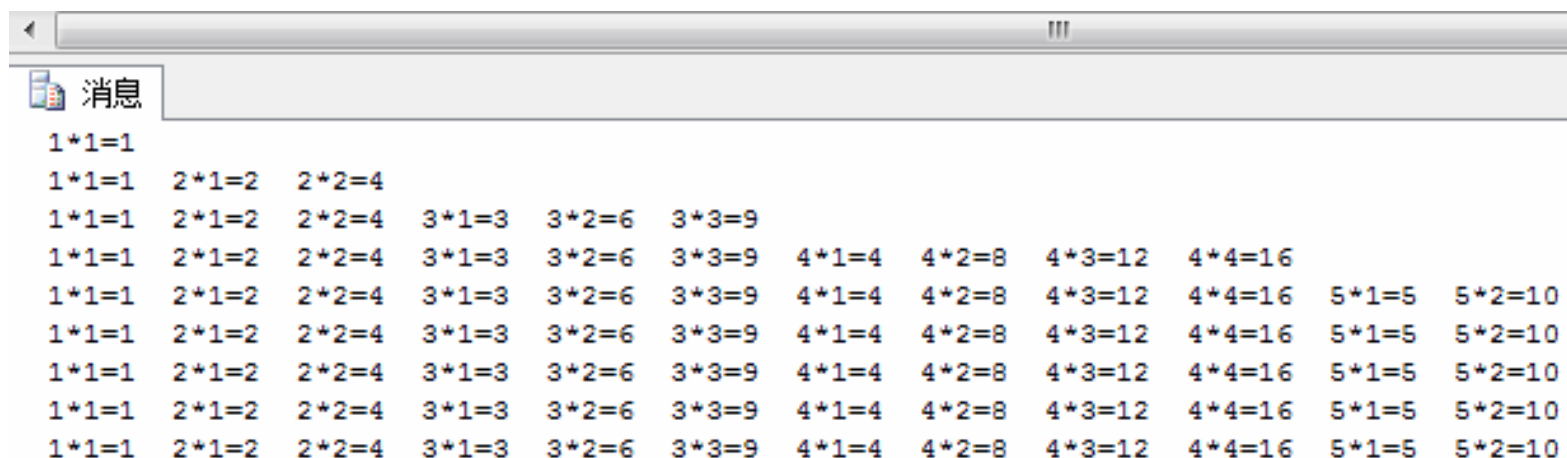
案例2

- 计算1到100的偶数的和

```
declare @i int,@sum int
set @i = 1
set @sum = 0
while @i<100
begin
    if (@i % 2) = 0
    begin
        set @sum = @sum + @i
    end
    set @i = @i + 1 -- 计数器+1
end
select @sum
```

案例3

- 编程输出九九乘法表



```

1*1=1
1*1=1 2*1=2 2*2=4
1*1=1 2*1=2 2*2=4 3*1=3 3*2=6 3*3=9
1*1=1 2*1=2 2*2=4 3*1=3 3*2=6 3*3=9 4*1=4 4*2=8 4*3=12 4*4=16
1*1=1 2*1=2 2*2=4 3*1=3 3*2=6 3*3=9 4*1=4 4*2=8 4*3=12 4*4=16 5*1=5 5*2=10
1*1=1 2*1=2 2*2=4 3*1=3 3*2=6 3*3=9 4*1=4 4*2=8 4*3=12 4*4=16 5*1=5 5*2=10
1*1=1 2*1=2 2*2=4 3*1=3 3*2=6 3*3=9 4*1=4 4*2=8 4*3=12 4*4=16 5*1=5 5*2=10
1*1=1 2*1=2 2*2=4 3*1=3 3*2=6 3*3=9 4*1=4 4*2=8 4*3=12 4*4=16 5*1=5 5*2=10
1*1=1 2*1=2 2*2=4 3*1=3 3*2=6 3*3=9 4*1=4 4*2=8 4*3=12 4*4=16 5*1=5 5*2=10
  
```


• 5. Goto语句

- Goto语句用于实现程序跳转，作用是跳过Goto语句后面的SQL语句，并从标号所定义的位置处继续执行。
- 语法格式如下：
- **GOTO label**
- 说明：label为标号，标号的定义部分由标识符和冒号“:”组成，而GOTO后面的标号只写标示符。

• 6. Return语句

- 无条件终止批处理、存储过程或查询的执行，Return之后的语句是不执行的。具体的语法格式为：
- **Return [integer_expression]**
- 其中，integer_expression表示返回的整数值。返回值是可以省略的，这时系统将根据程序的执行情况返回一个整数，其中0表示执行成功，非0值则表示失败。

案例1

- 使用goto实现do while语句

```
declare @i int
```

```
set @i = 1
```

```
start:
```

```
print @i
```

```
set @i = @i + 1
```

```
if @i <= 10
```

```
goto start
```

- 7. Waitfor语句

- Waitfor语句用于暂停程序的执行，直到指定的时间间隔已过或到达指定的时间点。具体的语法格式为：
 - `Waitfor {Delay 'time_to_pass' | Time 'time_to_execute'}`
 - 其中， Delay关键字后的time_to_pass， 指定延迟的时间， 也就是等待的时间间隔， 等待的最长时间为24小时。Time关键字后的time_to_execute， 指定运行批处理、存储过程或事务的时间。time_to_pass和time_to_execute的格式为' hh:mm:ss'， 不能指定日期。

- 注意：
 - 1,执行 WAITFOR 语句时，该事务会处于运行状态，其他请求不能在同一事务下运行。
 - 2,WAITFOR会阻止事务的处理，所以在同一事务中所有执行都会等WAITFOR执行完成才会返回

案例1

- 每隔2秒输出当前时间

```
declare @i int
set @i = 1
while @i < 10
begin
    waitfor delay '00:00:02'
    print convert(varchar(30),getdate(),21)
    set @i = @i + 1
end
go
```

- 8. 屏幕输出语句Print

- 在程序运行过程中经常需要向屏幕输出一些中间结果或最后的结果值，Print语句用于实现这一功能，向屏幕输出信息，其语法格式为：

- `Print msg_str | @local_variable | string_expr`

- 参数说明如下。
- `msg_str`: 要输出的字符串，用单引号括起来。
- `@local_variable`: 任何有效的字符数据类型的局部变量。
- `string_expr`: 输出的字符串的表达式。可由字符串连接运算符连接的常量、函数和变量等组成。