第 5 章 Transact-SQL 程序设计

Transact-SQL 是 Microsoft 对 SQL 语言进行扩展而形成的一种数据库语言。SQL 是标准的数据库语言，几乎可以在所有关系数据库上使用。但 SQL 语言只能按照先后顺序逐条执行，它没有控制语句。Transact-SQL 的贡献主要是 SQL Server 在 SQL 语言的基础上添加了控制语句，是标准 SQL 的超集。

通过本章的学习，读者可以：

- 了解 Transact-SQL 语言的特点和构成元素
- 掌握 Transact-SQL 语言中常量和变量的定义和引用方法
- 掌握 Transact-SQL 运算符的使用方法
- 重点掌握 Transact-SQL 的控制语句，包括 IF 语句、CASE 函数、WAITFOR 语句等
- 了解常用的系统内置函数，掌握用户自定义函数的定义和引用方法

5.1 Transact-SQL 概述

5.1.1 关于 Transact-SQL 语言

我们知道，SQL 语句只能按照既定的顺序执行，在执行过程中不能根据某些中间结果有选择地或循环地执行某些语句块，不能像高级程序语言那样进行流程控制。这使得在程序开发中存在诸多不便。为此，微软公司对 SQL 语言进行了扩充，主要是在 SQL 语言的基础上添加了流程控制语句，从而得到一种结构化程序设计语言——Transact-SQL。

Transact-SQL 即事务 SQL，也简写为 T-SQL，它是微软公司对关系数据库标准语言 SQL 进行扩充的结果，是 SQL 语言的超集。Transact-SQL 支持所有的标准 SQL 语言操作。

作为一种标准的关系数据库语言，SQL 几乎可以在所有的关系数据库上使用。但由于 Transact-SQL 是微软公司对 SQL 扩充的结果，所以只有 SQL Server 支持 Transact-SQL，而其他关系数据库（如 Access、Oracle 等）不支持 Transact-SQL。无论是数据库管理员还是数据库应用程序开发人员，要想深入领会和掌握数据库技术，必须认真学习 Transact-SQL。除了拥有 SQL 语言所有的功能外，Transact-SQL 还具备对 SQL Server 数据库独特的管理功能。使用 Transact-SQL，用户不但可以直接实现对数据库的各种管理，而且还可以深入数据库系统内部，完成各种图形化管理工具所不能完成的管理任务。

5.1.2 Transact-SQL 的元素

1. 标识符

在数据库编程中，要访问任何一个逻辑对象（如变量、过程、触发器等）都需要通过其名

称来完成。逻辑对象的名称是利用合法的标识符来表示的，是在创建、定义对象时设置的，此后就可以通过名称来引用逻辑对象。

标识符有两种类型：常规标识符和分隔标识符。

常规标识符在使用时不需将其分隔开，但要符合标识符的格式规则。这些规则就是，标识符中的首字符必须是英文字母、数字、_（下画线）、@、# 或汉字，首字符后面可以是字母、数字、下画线、@和 $ 等字符，还可以包含汉字。标识符一般不能与 SQL Server 的关键字重复，也不应以@@开头（因为系统全局变量的标识符是以@@开头），不允许嵌入空格或其他特殊字符等。

分隔标识符是指包含在两个单引号('')或者方括号([ ])内的字符串，这些字符串中可以包含空格。

2. 数据类型

与其他编程语言一样，Transact-SQL 语言也有自己的数据类型。数据类型在定义数据对象（如列、变量和参数等）时是必须的。自 SQL Server 2008 版本开始就新增了 XML 数据类型，以用于保存 XML 数据。Transact-SQL 语言的其他数据类型与 SQL 的相同，已经在第 4 章中进行了说明，在此不赘言。

3. 函数

SQL Server 2008 内置了大量的函数，如时间函数、统计函数、游标函数等，极大地方便程序员的使用。

4. 表达式

表达式是由表示常量、变量、函数等的标识符通过运算符连接而成的、具有实际计算意义的合法字符串。有的表达式不一定含有运算符，实际上单个的常量、变量等都可以视为一个表达式，它们往往不含有运算符。

5. 注释

Transact-SQL 语言中有两种注释，一种是行单行注释，另一种是多行注释。它们分别用符号“——”（连续的两个减号）和“/* */”来实现。

6. 关键字

关键字也称为保留字，是 SQL Server 预留作专门用途的一类标识符。例如，ADD、EXCEPT、PERCENT 等都是保留关键字。用户定义的标识符不能与保留关键字重复。

5.2 Transact-SQL 的变量和常量

在 Transact-SQL 中有两种类型的变量，一种是全局变量，另一种是局部变量。全局变量由 SQL Server 预先定义并负责维护的一类变量，它们主要用于保存 SQL Server 系统的某些参数值和性能统计数据，使用范围覆盖整个程序，用户对全局变量只能引用而不能修改或定义。局部变量是由用户根据需要定义的、使用范围只局限于某一个语句块或过程体内的一类变量。局部变量主要用于保存临时数据或由存储过程返回的结果。

5.2.1　变量的定义和使用

1. 全局变量

在 SQL Server 中，全局变量是以@@开头，后跟相应的字符串，如@@VERSION 等。如果想查看全局变量的值，可用 SELECT 语句或 print 语句来完成。例如，查看全局变量的值@@VERSION 的值，相应的 print 语句如下：

```
print @@VERSION;
```

该语句执行后在笔者计算机上输出如下结果：

```
Microsoft SQL Server 2017 (RTM) - 14.0.1000.169 (X64)
    Aug 22 2017 17:04:49
    Copyright (C) 2017 Microsoft Corporation
     Developer Edition (64-bit) on Windows 10 Home China 10.0 <X64> (Build
18362: )
```

表 5.1 列出了自 SQL Server 2008 开始提供的常用全局变量，以方便读者使用。

表 5.1　SQL Server 全局变量

全局变量名	说　明
@@CONNECTIONS	存储自上次启动 SQL Server 以来连接或试图进行连接的次数
@@CPU_BUSY	存储最近一次启动以来 CPU 的工作时间，单位为毫秒
@@CURSOR_ROWS	存储最后连接上并打开的游标中当前存在的合格行的数量
@@DATEFIRST	存储 DATAFIRST 参数值，该参数由 SET DATEFIRST 命令来设置(SET DATEFIRST 命令用来指定每周的第一天是星期几)
@@DBTS	存储当前数据库的时间戳值
@@ERROR	存储最近执行语句的错误代码
@@FETCH_STATUS	存储上一次 FETCH 语句的状态值
@@IDENTITY	存储最后插入行的标识列的列值
@@IDLE	存储自 SQL Server 最近一次启动以来 CPU 空闲的时间，单位为毫秒
@@IO_BUSY	存储自 SQL Server 最近一次启动以来 CPU 用于执行输入输出操作的时间，单位为毫秒
@@LANGID	存储当前语言的 ID 值
@@LANGUAGE	存储当前语言名称，如“简体中文”等
@@LOCK_TIMEOUT	存储当前会话等待锁的时间，单位为毫秒
@@MAX_CONNECTIONS	存储可以连接到 SQL Server 的最大连接数目
@@MAX_PRECISION	存储 decimal 和 numeric 数据类型的精确度
@@NESTLEVEL	保存存储过程或触发器的嵌套层
@@OPTIONS	存储当前 SET 选项的信息
@@PACK_RECEIVED	存储输入包的数目

续表

全局变量名	说　明
@@PACK_SENT	存储输出包的数目
@@PACKET_ERRORS	存储错误包的数目
@@PROCID	保存存储过程的 ID 值
@@REMSERVER	存储远程 SQL Server 2008 服务器名，NULL 表示没有远程服务器
@@ROWCOUNT	存储最近执行语句所影响的行的数目
@@SERVERNAME	存储 SQL Server 2008 本地服务器名和实例名
@@SERVICENAME	存储服务名
@@SPID	存储服务器 ID 值
@@TEXTSIZE	存储 TEXTSIZE 选项值
@@TIMETICKS	存储每一时钟的微秒数
@@TOTAL_ERRORS	存储磁盘的读写错误数
@@TOTAL_READ	存储磁盘读操作的数目
@@TOTAL_WRITE	存储磁盘写操作的数目
@@TRANCOUNT	存储处于激活状态的事务数目
@@VERSION	存储有关版本的信息，如版本号、处理器类型等

2. 局部变量

（1）定义局部变量。局部变量是由用户定义的，语法如下：

```
DECLARE @variable1 data_type[, @variable2 data_type, …]
```

其中，@variable1，@variable2，…为局部变量名，它们必须以单字符“@”开头；data_type 为数据类型，可以是系统数据类型，也可以是用户定义的数据类型，具体选择什么样的类型要根据实际需要而定。有关数据类型的说明见第 4 章的相关内容。

【例 5.1】 定义一个用于存储姓名的局部变量。

```
DECLARE @name_str varchar(8);
```

【例 5.2】 同时定义 3 个局部变量，分别用于存储学号、出生日期和平均成绩。

```
DECLARE @no_str varchar(8), @birthday_str smalldatetime, @avgrade_num numeric
(4,1);
```

（2）使用 SET 对局部变量赋初值。在定义局部变量以后，变量自动被赋予空值（NULL）。如果需要对已经定义的局部变量赋一个初值，可用 SET 语句来实现，其语法如下：

```
SET @variable = value;
```

其中，@variable 为局部变量名，value 为新赋的值。

【例 5.3】　对例 5.2 定义的 3 个变量@no_str、@birthday_str 和@avgrade_num 分别赋初值'20170112'、'2000-2-5'和 89.8。

这个赋值操作可使用以下 3 条 SET 语句来完成：

```
SET @no_str='20170112';
SET @birthday_str='2000-2-5';
SET @avgrade_num = 89.8;
```

注意，不能同时对多个变量进行赋值，这与同时对多个变量进行定义的情况不同。例如，如下的 SET 语句是错误的：

```
SET @no_str='20170112', @birthday_str='2000-2-5', @avgrade_num = 89.8;  --错误
```

(3) 使用 SELECT 对局部变量赋初值。SELECT 是查询语句，利用该语句可以将查询的结果赋给相应的局部变量。如果查询返回的结果包含多个值，则将最后一个值赋给局部变量。

使用 SELECT 对局部变量赋初值的语法格式如下：

```
SELECT @variable1= value1[, @variable2= value2, …]
FROM table_name
[WHERE …]
```

【例 5.4】　查询表 student，将姓名为“刘洋”的学生的学号、出生日期和平均成绩分别赋给局部变量@no_str、@birthday_str 和@avgrade_num。

该赋值操作用 SELECT 语句来实现则非常方便，其代码如下：

```
SELECT @no_str = s_no, @birthday_str = s_birthday, @avgrade_num = s_avgrade
FROM student
WHERE s_name = '刘洋';
```

局部变量在定义并赋值以后就可以当作一个常量值使用了。下面是一个使用局部变量的例子。

【例 5.5】　先定义局部变量@s_no 和@s_avgrade，然后对其赋值，最后利用这两个变量修改数据表 student 的相关信息。

```
USE MyDatabase --MyDatabase 使是用 CREATE DATABASE 语句创建的数据库，见第 6 章
GO
--定义局部变量
DECLARE @s_no varchar(8), @s_avgrade numeric(4,1);
--对局部变量赋值
SET @s_no='20170208';
SET @s_avgrade = 95.0;
--使用局部变量
UPDATE student SET s_avgrade = @s_avgrade
WHERE s_no = @s_no;
```

5.2.2　Transact-SQL 的常量

常量，也称为文字值或标量值，它是表示一个特定数据值的符号。常量的格式取决于它所表示的数据值的数据类型。按照数据值类型的不同，常量可以分为字符串常量、整型常量

等，以下分别说明。

1. 字符串常量

与其他编程语言一样，字符串常量是最常用的常量之一。

字符串常量是由两个单引号来定义的，是包含在两个单引号内的字符序列。这些字符包括字母数字字符(a～z、A～Z 和 0～9)以及特殊字符，如感叹号(!)、at 符(@)和数字号(#)等。默认情况下，SQL Server 2017 为字符串常量分配当前数据库的默认排序规则，但也可以用 COLLATE 子句为其指定排序规则。

例如，如下都是合法字符串常量：

```
'China'
'中华人民共和国'
```

如果字符串中包含一个嵌入的单引号，则需要在该单引号前再加上一个单引号，表示转义，这样才能定义包含单引号的字符串。

例如，如下包含单引号的字符串都是合法的：

```
'AbC''Dd!'                          --表示字符串“AbC'Dd!”
'xx: 20%''y''%.'
```

有许多程序员习惯用双引号来定义字符串常量。但在默认情况下，SQL Server 不允许使用这样的定义方式。然而，如果将 QUOTED_IDENTIFIER 选项设置为 OFF，则 SQL Server 同时支持运用双引号和单引号来定义字符串。

设置 QUOTED_IDENTIFIER 的方法用如下语句：

```
SET QUOTED_IDENTIFIER OFF;
```

在执行该语句后，QUOTED_IDENTIFIER 被设置为 OFF。这时除了单引号以外，还可以用双引号来定义字符串。例如，如下定义的字符串都是合法的：

```
'China'
'中华人民共和国'
'AbC''Dd!'                          --表示字符串“AbC'Dd!”
'xx: 20%y%.'
"China"
"中华人民共和国"
"AbC''Dd!"                          --表示字符串“AbC''Dd!”
"xx: 20%y%."
```

需要注意的是，当用双引号定义字符串时，如果该字符串中包含单引号，则不能在单引号前再加上另一个单引号，否则将得到另外的一种字符串。例如，'AbC"Dd!'定义的是字符串“AbC'Dd!”，而"AbC"Dd!"定义则是字符串“AbC"Dd!”。

SQL Server 将空字符串解释为单个空格。

如果不需要用双引号来定义字符串，则只要将 QUOTED_IDENTIFIER 恢复为默认值 ON 即可。需要执行如下语句：

```
SET QUOTED_IDENTIFIER ON;
```

SQL Server 2017 支持 Unicode 字符串。Unicode 字符串是指按照 Unicode 标准来存

储的字符串。但在形式上与普通字符串相似，不同的是它前面有一个 N 标识符（N 代表 SQL-92 标准中的区域语言），且前缀 N 必须是大写字母。例如，'China'是普通的字符串常量，而 N'China'则是 Unicode 字符串常量。

2. 整型常量

整型常量也用得很多，它是不用引号括起来且不包含小数点的数字字符串。例如，2007、－14 等都是整型常量。例如，下面是定义整型常量及对其赋值的例子：

```
DECLARE @i integer
SET @i = 99;
```

3. 日期时间常量

日期时间常量通常是用字符串常量来表示，但前提是字符串常量能够隐式转换为日期时间型数据，其格式为“yyyy-mm-dd hh:mm:ss.nnn”或“yyyy/mm/dd hh:mm:ss.nnn”，其中 yyyy 表示年份，第一个 mm 表示月份，dd 表示月份中的日期，hh 表示小时，第二个 mm 表示分钟，ss 表示秒，nnn 表示毫秒。如果“yyyy-mm-dd”缺省，则日期部分默认为 1900 年 01 月 01 日；如果“hh:mm:ss.nnn”缺省，则时间部分默认为 00 时 00 分 00.000 秒。

例如，下面是一些将日期时间型常量赋给日期时间型变量的例子：

```
DECLARE @dt datetime
SET @dt = '2017-01-03 21:55:56.890'    --2017 年 01 月 03 日 21 时 55 分 56.890 秒
SET @dt = '2017/01/03'                 --2017 年 01 月 03 日 0 时 0 分 0 秒
SET @dt = '2017-01-03'                 --2017 年 01 月 03 日 0 时 0 分 0 秒
SET @dt = '21:55:56.890'               --1900 年 01 月 01 日 21 时 55 分 56.890 秒
```

4. 二进制常量

二进制常量是用前缀为 0x 的十六进制数字的字符串来表示，但这些字符串不需要使用单引号括起。例如，如下是将二进制常量赋给二进制变量的例子：

```
DECLARE @bi binary(50)
SET @bi = 0xAE
SET @bi = 0x12Ef
SET @bi = 0x69048AEFDD010E
SET @bi = 0x0
```

5. 数值型常量

数值型常量包括 3 种类型：decimal 型常量、float 型常量和 real 型常量。

decimal 型常量是包含小数点的数字字符串，但这些字符串不需单引号括起来（定点表示）。例如，下面是 decimal 型常量的例子：

```
3.14159
-1.0
```

float 型常量和 real 型常量都是使用科学记数法来表示（浮点表示）。例如：

```
101.5E5
-0.5E-2
```

6. 位常量

位常量使用数字0或1来表示，并且不用单引号括起来。如果使用一个大于1的数字，则该数字将转换为1。例如：

```
DECLARE @b bit
SET @b = 0;
```

7. 货币常量

货币常量是前缀为可选的小数点和可选的货币符号的数字字符串，且不用单引号括起来。SQL Server 2008不强制采用任何种类的分组规则，例如，在代表货币的字符串中不允许每隔3个数字用一个逗号隔开。例如，以下是货币常量的例子：

```
$20000.2                                    --而$20,000.2是错误的货币常量
$200
```

8. 唯一标识常量

这是指uniqueidentifier类型的常量，它使用字符或二进制字符串格式来指定。例如：

```
'6F9619FF-8B86-D011-B42D-00C04FC964FF'
0xff19966f868b11d0b42d00c04fc964ff
```

以上介绍了8种类型的常量，它们主要运用于对变量和字段赋值、构造表达式、构造子句等。在今后的介绍中将进一步领会到它的使用方法。

5.3 Transact-SQL 运算符

运算符是用来指定要在一个或多个表达式中执行操作的一种符号。在SQL Server 2017中，使用的运算符包括算术运算符、逻辑运算符、赋值运算符、字符串连接运算符、位运算符和比较运算符等。

1. 算术运算符

算术运算符包括加(+)、减(-)、乘(*)、除(/)和取模(%)等5种运算符。它们用于执行对两个表达式的运算，这两个表达式的返回值必须是数值数据类型，包括货币型。

加(+)和减(-)运算符还可以用于对日期时间类型值的算术运算。

2. 逻辑运算符

逻辑运算符用于对某些条件进行测试，返回值为TRUE或FALSE。逻辑运算符包括ALL、AND、ANY、BETWEEN、EXISTS、IN、LIKE、NOT、OR、SOME等，其含义说明如表5.2所示，其中有部分运算符已在第5章中介绍过。

表5.2 逻辑运算符及其含义

逻辑运算符	含　义
ALL	在一组的比较中只有所有的比较都返回TRUE，则运算结果返回TRUE，否则返回FALSE

续表

逻辑运算符	含　义
AND	对两个表达式进行逻辑与运算，即如果两个表达式的返回值均为 TRUE，则运算结果返回 TRUE，否则返回 FALSE
ANY	在一组的比较中只要有一个 TRUE，则运算结果返回 TRUE，否则返回 FALSE
BETWEEN	测试操作数是否在 BETWEEN 指定的范围之内，如果在则返回 TRUE，否则返回 FALSE
EXISTS	测试查询结果是否包含某些行，如果包含则返回 TRUE，否则返回 FALSE
IN	测试操作数是否在 IN 后面的表达式列表中，如果在则返回 TRUE，否则返回 FALSE
LIKE	测试操作数是否与 LIKE 后面指定的模式相匹配，如果匹配返回 TRUE，否则返回 FALSE
NOT	对表达式的逻辑值取反
OR	对两个表达式进行逻辑与或运算，即如果两个表达式的返回值均为 FALSE，则运算结果返回 FALSE，否则返回 TRUE
SOME	在一组的比较中只要有部分比较都返回 TRUE，则运算结果返回 TRUE，否则返回 FALSE

3. 赋值运算符

赋值运算符就是等号“=”，它是 Transact-SQL 中唯一的赋值运算符。例如，5.2 节对局部变量的赋值操作实际上已经使用了赋值运算符。

除了用作赋值操作以外，赋值运算符还可以用于建立字段标题和定义字段值的表达式之间的关系。例如，如下语句创建了两个字段，其中第一个字段的列标题为“中国”，所有字段值均为“China”；第二个字段的列标题为“姓名”，该字段的字段值来自表 student 中的 s_name 字段值。

```
SELECT 中国= 'China', 姓名= s_name
FROM student
```

执行结果如下：

```
中国     姓名
----------------------
China   刘洋
China   王晓珂
China   王伟志
China   岳志强
China   贾簿
China   李思思
China   蒙恬
China   张宇
```

4. 字符串连接运算符

在 SQL Server 中，字符串连接运算符为加号“+”，表示要将两个字符串连接起来而形成一个新的字符串。该运算符可以操作的字符串类型包括 char、varchar、text 以及 nchar、

nvarchar、ntext 等。以下是字符串连接的几个例子：

```
'abc'+'defg'       --结果为'abcdefg'
'abc' + '' + 'def'  --结果为'abcdef'(默认),当兼容级别设置为 65 时结果为'abc def'
```

针对字符串的操作有很多种，如取子串等。但在 SQL Server 中仅有字符串连接操作由运算符“+”来完成，而所有其他的字符串操作都使用字符串函数来进行处理。

5. 位运算符

位运算符是表示在两个操作数之间执行按位进行运算的符号，操作数必须为整型数据类型之一，如 bit、tinyint、smallint、int、bigint 等，还可以是二进制数据类型(image 数据类型除外)。表 5.3 列出了位运算符及其含义。

表 5.3　位运算符及其含义

位 运 算 符	含　义
&	对两个操作数按位逻辑与
|	对两个操作数按位逻辑或
^	对两个操作数按位逻辑异或
~	对一个操作数按位逻辑取非

6. 比较运算符

比较运算符用于测试两个表达式的值之间的关系，这种关系是指等于、大于、小于、大于或等于、小于或等于、不等于、不小于、不大于等。比较运算符几乎适用于所有的表达式(除了 text、ntext 或 image 数据类型的表达式外)。表 5.4 列出了 Transact-SQL 支持的比较运算符。

表 5.4　比较运算符

运　算　符	含　义
=	等于
>	大于
<	小于
>=	大于或等于
<=	小于或等于
<>	不等于
!=	不等于
!<	不小于
!>	不大于

7. 运算符的优先级

运算符执行顺序的不同会导致不同的运算结果，所以正确地理解运算符的优先级是必