T-SQL 函数实战教程

T-SQL 函数实战教程

T-SQL 是微软在关系型数据库管理系统 SQL Server 中的 SQL-3 标准的实现，是微软对 SQL 的扩展，具有 SQL 的主要特点，同时增加了变量、运算符、函数、流程控制和注释等语言元素，使得其功能更加强大。并且 T-SQL 自带了许多实用的函数，DBA 可以利用这些函数来进行更有效地管理。在本次的技术手册中，我们将提供一系列的 T-SQL 函数教程，其中包括字符串操作函数、集合函数、元数据函数以及系统函数等，充分理解并掌握这些知识，对 DBA 来说十分必要。

T-SQL 字符串函数

T-SQL 自带了一些字符串处理函数，可以用来处理数值型数据和字符型数据。本部分对这些函数做了个简要介绍，并给出一些介绍如何使用的例子。

- T-SQL 字符串函数：截取字符串的值
- T-SQL 字符串函数：修改字符串的值
- T-SQL 字符串函数：转换字符串值
- T-SQL 字符串函数：获取字符串的相关信息
- T-SQL 字符串函数：对字符串执行“SOUNDEX”分析

T-SQL 集合函数

在 SQL Server 2008 中，Transact-SQL 有一些集合函数，这些函数支持你执行这类计算。一个 T-SQL 集合函数返回一个值，可以被作为一个表达式用在 SELECT 查询列表或者“COMPUTE”，“COMPUTE BY”或者“HAVING”从句中。

- T-SQL 集合函数基础知识

- 利用 T-SQL 集合函数计算行数
- 利用 T-SQL 集合函数操作分组数据
- 利用 T-SQL 集合函数验证总和校验
- 利用 T-SQL 集合函数获取统计数据

T-SQL 数学函数

SQL Server 2008 中的 T-SQL 数学函数用来对数值表达式执行数值运算。例如，你可以利用数值函数计算角度的正弦值和余弦值，返回数值的平方或者平方根，或者按给定的精度做四舍五入。

- 利用 T-SQL 数学函数做‘三角函数’运算（上）
- 利用 T-SQL 数学函数做‘三角函数’运算（下）
- T-SQL 代数函数与数值函数的用法（上）
- T-SQL 代数函数与数值函数的用法（下）

T-SQL 子查询

SQL Server 支持的 T-SQL 语言特性之一是子查询(subquery)，它可以内嵌在一个 SELECT、INSERT、UPDATE 或 DELETE 语句中。你可以定义一个 T-SQL 子查询，然后将其放在 SQL 语句中任何一个允许使用表达式的地方。

- 在 SQL Server 中如何使用子查询（上）
- 在 SQL Server 中如何使用子查询（下）
- T-SQL 子查询与操作符的组合：比较运算符
- T-SQL 子查询与操作符的组合：IN、NOT IN
- T-SQL 子查询与操作符的组合：EXIST、NOT EXIST

T-SQL 系统函数

一些 SQL Server 函数被归类为系统函数，这些函数支持你查看关于更新和插入操作的信息，提取服务器属性设置，访问当前会话及其用户的详细信息。你还可以利用系统函数执行许多语言相关的任务。

- T-SQL 系统函数：查看数据修改信息
- T-SQL 系统函数：查看服务器属性
- T-SQL 系统函数：查看用户信息
- T-SQL 系统函数：查看会话信息
- T-SQL 系统函数：查看一般信息
- T-SQL 系统函数：利用语言工具

T-SQL 元数据函数

SQL Server 2008 提供了称为元数据函数的一组函数。这些函数使你可以提取关于数据库和数据库对象的信息。例如，你可以提取赋予数据库，对象，表或者数据类型的 ID。

- 探索 SQL Server 2008 中的 T-SQL 元数据函数（上）
- 探索 SQL Server 2008 中的 T-SQL 元数据函数（下）
- 利用 T-SQL 元数据函数检查属性设置（上）
- 利用 T-SQL 元数据函数检查属性设置（下）
- 利用 T-SQL 元数据函数获取数据库文件信息

T-SQL 配置函数

Transact-SQL 提供了一组可以获取当前 SQL Server 实例配置选项设置信息的函数。这些函数被称为“配置函数”，这些语言元素支持你提取许多信息。比如：当前使用的语言名称和 ID，允许并发的用户最大连接数，或者是你所连接 SQL Server 实例的版本。

- SQL Server 配置函数实战教程（上）
- SQL Server 配置函数实战教程（下）

T-SQL 字符串函数：截取字符串的值

T-SQL 自带了一些字符串处理函数，可以用来处理数值型数据和字符型数据。本文共分两部分，对这些函数做了个简要介绍，并给出一些介绍如何使用的例子。我们在第一部分先介绍如何截取和修改字符串的值。第二部分介绍如何转换和手机字符串相关信息，以及如何执行“SOUNDEX”分析。

T-SQL 提供了丰富的字符串处理函数，你可以用这些函数方便地处理各种类型的字符串，比如 char 类型，nchar 类型，varchar 类型和 nvarchar 类型。例如，你可以用字符串函数删除字符串尾部的空格，截取字符串，或者转换数值型数据为字符型数据。字符串处理函数给你的 T-SQL 语句带来了灵活性，使你可以更容易地操作没按你的需要出现的字符串数据。

在本文中，我会描述 SQL Server 中的一些基本字符串处理函数，并举例说明它们的用法。我在 SQL Server 2008 上创建了一个本地实例来运行这些示例，基于“AdventureWorks2008”样例数据库进行。这些例子可以分为五类：截取字符串数据，修改数据，转换数据，获取数据相关信息，对数据执行 SOUNDEX 分析。这种组织方式是为了使大家对每个函数和它们之间的差异有一个更好的理解。

截取字符串的值

下面的一组函数详细描述了如何截取字符串的一部分。这些函数大部分都包含在下面的 SELECT 语句中：

```
SELECT
Name,
LEFT(Name, 12) AS BikeType,
SUBSTRING(Name, 9, 4) AS Model,
RIGHT(Name, 2) AS Size,
RTRIM(Style) AS Style
FROM
Production.Product
WHERE
ProductNumber LIKE 'bk-t%'
```

你可以看到，SELECT 语句中包括函数“LEFT”，“SUBSTRING”，“RIGHT”和“RTRIM”。在“AdventureWorks2008”数据库中运行该语句时，你会得到如下结果：

Name	BikeType	Model	Size	Style
Touring-2000 Blue, 60	Touring-2000	2000	60	U
Touring-1000 Yellow, 46	Touring-1000	1000	46	U
Touring-1000 Yellow, 50	Touring-1000	1000	50	U
Touring-1000 Yellow, 54	Touring-1000	1000	54	U
Touring-1000 Yellow, 60	Touring-1000	1000	60	U
Touring-3000 Blue, 54	Touring-3000	3000	54	U
Touring-3000 Blue, 58	Touring-3000	3000	58	U
Touring-3000 Blue, 62	Touring-3000	3000	62	U
Touring-3000 Yellow, 44	Touring-3000	3000	44	U
Touring-3000 Yellow, 50	Touring-3000	3000	50	U
Touring-3000 Yellow, 54	Touring-3000	3000	54	U
Touring-3000 Yellow, 58	Touring-3000	3000	58	U
Touring-3000 Yellow, 62	Touring-3000	3000	62	U
Touring-1000 Blue, 46	Touring-1000	1000	46	U
Touring-1000 Blue, 50	Touring-1000	1000	50	U
Touring-1000 Blue, 54	Touring-1000	1000	54	U
Touring-1000 Blue, 60	Touring-1000	1000	60	U
Touring-2000 Blue, 46	Touring-2000	2000	46	U
Touring-2000 Blue, 50	Touring-2000	2000	50	U
Touring-2000 Blue, 54	Touring-2000	2000	54	U
Touring-3000 Blue, 44	Touring-3000	3000	44	U
Touring-3000 Blue, 50	Touring-3000	3000	50	U

(查询到 22 行记录)

我们来分别看看这些函数。“LEFT”函数和“RIGHT”函数很相似。“LEFT”函数返回字符串左面的部分，“RIGHT”函数返回字符串右边的部分，这两个函数都根据指定的字符数进行截取。

例如，在上面的“SELECT”语句中，“LEFT”函数返回“Name”列的前 12 个字符。该函数有两个参数：第一个参数是字符串表达式，第二个参数是字符数。在这个例子中，字符串表达式是“Name”列，字符数是“12”。结果，“Name”列所有值的后半部分都被截掉了，只剩下 12 个字符返回。如果你留意结果集的第一行，你会发现“Name”列的值“Touring-2000 Blue”已经变成了“Touring-2000”。

“RIGHT”函数的用法也是一样的，它截取字符串右边的部分。在上面的例子中，“RIGHT”函数只返回“Name”列的最后两个字符。同样，“Name”列被指定为字符串表达式，设置的字符数为“2”，返回的结果为两位的字符串。

“SUBSTRING”函数可以提取字符串的任意部分。该函数有三个参数：第一个参数是字符串表达式，第二个参数是指定从哪里开始截取，第三个参数表示截取多长的字符串。在上面的例子中，字符串表达式仍然是“Name”列。第二个参数“9”的意思是我们应该从第九个字符开始(截取)，第三个参数“4”意思是获取四个字符。正如查询结果所示，“Model”列的每一行都只显示了第九到第十二个字符。例如，在第一行中，“2000”就从“Name”列中提取出来了。

本例中还有一个函数是“RTRIM”，它的功能是去掉字符串(在第一个参数中指定)尾部的所有空格。在本例中，RTRIM 函数用来删掉“Style”列尾部的空格。该列被配置为“NCHAR(2)”数据类型，意思是所有单字符值尾部都会补上一个空格(补足两位)。在某些应用程序和提取过程中，尾部的空格可能会引起问题(特别是在如果你试图对该值进行匹配的时候)。然而，“RTRIM”函数会删掉那个空格。

注意，“RTRIM”函数还有与它相对的另一个函数：“LTRIM”函数。它会删除字符串左侧的空格。尽管“RTRIM”更常用，“LTRIM”在某些情况下也很有用。

(作者：Robert Sheldon 译者：冯昀晖 来源：TT 中国)

原文标题：T-SQL 字符串函数：截取字符串的值

链接：http://www.searchdatabase.com.cn/showcontent_34436.htm

T-SQL 字符串函数：修改字符串的值

T-SQL 还包括可以在 SQL Server 中修改字符串值的一组函数。下面的语句就是一个例子：

```
SELECT
REPLACE(Name, 'Mountain', 'Mtn') AS ReplaceName,
STUFF(Name, 11, 2, 'x01') AS StuffName,
REVERSE(ProductNumber) AS ReverseNum,
STUFF(ProductNumber, 1, 2, REPLICATE(0, 3)) AS
ReplicateNum, ProductNumber + SPACE(4) + LOWER
(ProductLine) AS LowerProdLine
FROM
Production.Product
WHERE
ProductID = 771
```

这句 SQL 使用了几个函数来修改“Name”列和“ProductNumber”列的值。“Name”列的原始值是“Mountain-100 Silver，38”，“ProductNumber”列的原始值是“BK-M82S-38”。该语句返回如下结果：

```
ReplaceName          StuffName                ReverseNum
---------------      --------------------     ----------
Mtn-100 Silver, 38   Mountain-1x01 Silver, 38  83-S28M-KB

ReplicateNum         LowerProdLine
------------         ---------------
000-M82S-38          BK-M82S-38      m
(查询到 1 行记录)
```

该语句中使用的第一个函数是“REPLACE”，用来替换给定字符串值中指定的一组字符串为新的字符串。该函数有三个参数：第一个参数是字符串表达式，第二个参数是被替换的字符串，第三个参数是要替换的新字符串。在上面的例子中，REPLACE 函数指定“Name”列作为待操作的字符串。被替换的字符串是“Mountain”，新字符串是“Mtn”。结果，(替换后的)新值变成了“Mtn-100 Silver，38”。

接下来的函数是“STUFF”，它的作用是按照给定的位置删除一组指定的字符，并插入一组新的字符。该函数有四个参数：第一个参数是待操作的字符串表达式，第二个参数是待删除的字符串的起始位置，第三个参数是要删除字符的数量，第四个参数是要插入的字符串。在上面的例子中，“STUFF”函数指定“Name”列为待操作的字符串。要删除的字符串从第 11 个字符(第二个参数)开始，删除两个字符(第三个参数)。在那些字符删除后，新字符(x01)被插入到该位置，结果处理后的新字符串变成了“Mountain-1x01 Silver，38”。

函数“RESERVSE”的作用是把给定字符串按逆序排列。在本例中，待处理的字符串是“ProductNumber”列。结果，处理后的新产品编号变成了“83-S28M-KB”。

还有函数“REPLICATE”，该函数可以把指定字符串值重复指定的次数。该函数有两个参数：第一个参数是待处理的字符串值，第二个参数是重复次数。在上面的例子中，“0”被重复了三次。然而，要注意“REPLICATE”函数被嵌入到了“STUFF”函数中作为第四个参数。结果“ProductNumber”值的前两个字符被替换为了“000”。

“SPACE”函数与“REPLICATE”函数类似，它返回一组空格。该函数有一个参数，指定返回的空格数量。在上面的例子中，该组合值中添加了四个空格。

本例中还有个“LOWER”函数，它把字符串中的大写字符转换为小写字符。在上面的例子中，“ProductLine”的值被转换成了小写。T-SQL 还支持“UPPER”函数，它可以把小写字符转换为大写字符。

(作者：Robert Sheldon 译者：冯昀晖 来源：TT 中国)

原文标题：T-SQL 字符串函数：修改字符串的值

链接：http://www.searchdatabase.com.cn/showcontent_34437.htm

T-SQL 字符串函数：转换字符串值

在第一部分中，你学习了 T-SQL 字符串函数的一些基本知识以及运用方法。本文会进一步举例说明如何最大程度地利用好字符串处理函数，转换数字型和字符型数据类型，手机字符串相关信息，以及执行“SOUNDEX”分析。

转换字符串值

T-SQL 提供可以转换字符串值的函数，如下面的 SELECT 语句所示：

```
SELECT
Name + CHAR(9) + STR(MakeFlag) AS Product,
ASCII(MakeFlag) AsciiMakeFlag,
RTRIM(Style) AS Style,
UNICODE(RTRIM(Style)) AS UnicodeStyle
FROM
Production.Product
WHERE
ProductID = 771
```

该语句返回如下执行结果：

```
Product                      AsciiMakeFlag Style UnicodeStyle
---------------------------- ------------- ----- -------------
Mountain-100 Silver, 38    1 49            U     85

(1 row(s) affected)
```

该语句中的第一个函数是“CHAR”，可以把一个 ASCII 码整数(从 0 到 255)转换为字符，该函数只有一个参数(一个 ASCII 码整数值)。该函数在给你的字符串插入控制字符时非常方便，比如 tab 键(ASCII 码为 9)，换行符号(ASCII 码为 10)或者回车符(ASCII 码为 13)。

在上面的例子中，“CHAR(9)”用来给“Product”值插入 tab 键。要注意，T-SQL 还支持“NCHAR”函数，它可以把 Unicode 整数转换为 Unicode 字符。接下来的函数是“STR”，它可以把数值型数据转换为字符型数据。“STR”函数的参数是待转换的数值型

表达式。在上面的例子中，“MakeFlag”bit 列被转换为字符串，所以可以与“Name”列进行连接。要记住，如果没有转换 bit 列就连接两个列，会显示一条错误信息。

T-SQL 还支持“ASCII”函数，可以转换单个字符为 ASCII 码整数。本例中利用“ASCII”函数转换“MakeFlag”值为整数。因为该列的值为“1”，所有转换后的值就是“49”。

本例中还使用了“UNICODE”函数，该函数用来把一个字符转换为 Unicode 整数。该函数只有一个参数：Unicode 字符表达式。在上面的例子中，“Style”的值被转换为 Unicode 整数。然而，很重要的一点是，要先用“RTRIM”函数删除字符串尾部的空格，因为“UNICODE”函数(与 ASCII 函数一样)一次只能转换一个字符。

(作者：Robert Sheldon 译者：冯昀晖 来源：TT 中国)

原文标题：T-SQL 字符串函数：转换字符串值

链接：http://www.searchdatabase.com.cn/showcontent_34577.htm

T-SQL 字符串函数：获取字符串的相关信息

获取字符串的相关信息

还有一些函数可以提供与字符串有关的信息，比如字符串长度或者值内部子串的位置。下面的 SELECT 语句就演示了几个提供字符串相关信息的函数：

```
SELECT
LEN(Description) AS LengthDescrip,
CHARINDEX('tech', Description) AS CharIndexDescrip,
PATINDEX('%bike%', Description) AS PatIndexDescrip
FROM
Production.ProductDescription
WHERE
ProductDescriptionID = 321
```

上面语句中的函数都是基于“Description”列进行操作的。对于该语句查询的这一产品，“Description”列包含的值是“Same technology as all of our Road series bikes. Perfect all-around bike for road or racing. ”。该 SELECT 语句返回如下结果：

LengthDescrip	CharIndexDescrip	PatIndexDescrip
----------	----------	----------
93	6	43

(1 row(s) affected)

“LEN”函数返回指定字符串的字符数量。该函数只有唯一参数就是待计数的字符串。拿上面的“Description”列为例，该字符串有 93 个字符。

“CHARINDEX”函数的功能是从某字符串中指定的开始位置查找某字符串所在位置。该函数有三个参数：第一个参数是要搜索的字符串，第二个参数是待搜索范围字符串，第三个参数是搜索开始的位置。最后一个参数是可选的。在上面的例子中，我们在“Description”列中搜索“tech”第一次出现的位置，它是从字符串的第六个位置开始的。

上面的语句中最后一个函数是“PATINDEX”。该函数在字符串中搜索第一次出现某字符串的位置。该函数有两个参数：第一个参数是要匹配的字符串，第二个参数是在其中进

行搜索的字符串。可以用通配符定义匹配模式，提供灵活的搜索模式类型。然而，如果你使用通配符“%”，你必须在待搜索字符的前面或者后面都加上它，除非你搜索以指定字符串开头或结尾的匹配。例如，在上面的语句中，“%bike%”被指定为在“Description”值中搜索的匹配模式。结果显示，第一次匹配的实例在第 43 个字符位置。

（作者：Robert Sheldon 译者：冯昀晖 来源：TT 中国）

原文标题：T-SQL 字符串函数：获取字符串的相关信息

链接：http://www.searchdatabase.com.cn/showcontent_34691.htm

T-SQL 字符串函数：对字符串执行“SOUNDEX”分析

我们要讨论的最后一组字符串函数与“SOUNDEX”有关，它是一种可以把字符值转换成四个字符构成编码的系统，目的是确定类似发音的单词。“SOUNDEX”函数返回的第一个字符与目标字符串的第一个字符串相同，后面三个字符是对辅音求值得到的数值代码。(元音被忽略，除非元音是第一个字母。)

在下面的 T-SQL 语句中，我们用与“SOUNDEX”有关的函数来分析两个名字：

```
DECLARE @Name1 varchar(30)
SET @Name1 =
(SELECT LastName FROM Person.Person
WHERE BusinessEntityID = 341)
DECLARE @Name2 varchar(30)
SET @Name2 =
(SELECT LastName FROM Person.Person
WHERE BusinessEntityID = 441)
SELECT
@Name1 AS Name1,
@Name2 AS Name2,
SOUNDEX(@name1)AS SoundexName1,
SOUNDEX(@Name2) AS SoundexName2,
DIFFERENCE(@Name1, @Name2) AS SoundexDiff
```

“SET”子查询返回的名字是“Allen”和“Alan”，它们被插入到变量并在 SELECT 语句中使用，返回结果如下：

Name1	Name2	SoundexName1	SoundexName2	SoundexDiff
-----	-----	----------	----------	----------
Allen	Alan	A450	A450	4

(1 row(s) affected)

第一次使用“SOUNDEX”函数时，分析名字“Allen”;第二次分析名字“Alan”。要注意，该函数只有一个参数，就是待分析的字符串。在本例中，“SOUNDEX”分析两个拼写返回相同的结果，说明名字非常相似。这一点在用于识别某些名字拼写有两处不同的人时很方便。“DEFFERENCE”函数也有类似功能，它有两个参数，这两个参数就是待比较的字符串，返回结果是一个整数，表示两个字符串之间差异的程度。返回的整数结果可以是

0 到 4 之间的值。值 0 表示两个字符串之间几乎没有相似之处，而值 4 表示两个字符串非常相似。在上面的例子中，该函数返回 4，因为两个名字非常相似。

如果你想知道不太相似的两个名字运行时的结果，请运行下面的语句：

```
DECLARE @Name3 varchar(30)
SET @Name3 =
(SELECT LastName FROM Person.Person
WHERE BusinessEntityID = 1829)
DECLARE @Name4 varchar(30)
SET @Name4 =
(SELECT LastName FROM Person.Person
WHERE BusinessEntityID = 2580)
SELECT
@Name3 AS Name3,
@Name4 AS Name4,
SOUNDEX(@Name3)AS SoundexName3,
SOUNDEX(@Name4) AS SoundexName4,
DIFFERENCE(@Name3, @Name4) AS SoundexDiff
```

在本例中，“SOUNDEX”函数对名字“Su”返回值“S000”，对名字“Suarez”返回值“S620”。此外，“DIFFERENCE”函数返回值 3，结果如下表所示：

Name3	Name4	SoundexName3	SoundexName4	SoundexDiff
-----	------	------------	------------	-----------
Su	Suarez	S000	S620	3

(1 row(s) affected)

“SOUNDEX”函数和字符串处理函数为处理字符数据提供了极大的灵活性。尽管这些例子相对简单一些，我们可以利用这些函数创建更复杂的语句。一定要看看 SQL Server 联机手册获取更多相关信息。

(作者：Robert Sheldon 译者：冯昀晖 来源：TT 中国)

原文标题：T-SQL 字符串函数：对字符串执行“SOUNDEX”分析

链接：http://www.searchdatabase.com.cn/showcontent_34693.htm

T-SQL 集合函数基础知识

在操作 SQL Server 数据库中的数据时，有时你可能会需要对一组数值执行计算。例如，你的数据可能包括你所在组织的大量个体零售销售额。因此，你需要算出某一年度的总销量或者每个销售代表的平均销售额。

在 SQL Server 2008 中，Transact-SQL 有一些集合函数，这些函数支持你执行这类计算。一个 T-SQL 集合函数返回一个值，可以被作为一个表达式用在 SELECT 查询列表或者“COMPUTE”，“COMPUTE BY”或者“HAVING”从句中。所有集合函数都忽略“null”值，除了“COUNT”函数。

在本文中，我会展示如何使用 SQL Server 2008 中可用的集合函数。本文创建的示例都基于 SQL Server 2008 本地实例以及“AdventureWorks2008 ”示例数据库。这些示例是按照功能进行组织的，当然这种组织只是为了帮助说明和展示这些函数的一种方式。

T-SQL 集合函数基础知识

我要讲解的第一组 T-SQL 函数是“AVG”函数和“SUM”函数。“AVG”函数返回一组数值的平均值，而“SUM”函数返回一组数值的总和。这两个函数都会对“null”值做忽略处理。

下面的例子演示了这两个函数的用法，从“AdventureWorks2008 ”数据库的“SalesOrderHeader ”表中获取汇总的销售数据：

```
SELECT
TerritoryID,
AVG(SubTotal) AS AvgSales,
SUM(SubTotal) AS TotalSales
FROM
Sales.SalesOrderHeader
GROUP BY
TerritoryID
ORDER BY
TerritoryID
```

在本例中，我按“Territory ID”列对值进行分组，然后对产生的每个分组求一次平均值以及计算基于“SubTotal ”列求出来的销售总量。

正如你所看到的，要使用 T-SQL 集合函数，我必须指定函数名，后面跟着我要针对该列做合计的列名(在圆括号内)。要注意该列实际上可以是满足该函数要求的任意表达式。关于在函数中可以使用什么表达式的详细说明，请参见 SQL Server 联机丛书相应的主题。

这就是“AVG”和“SUM”集合函数的全部用法。执行上面的语句会返回如下表所示的结果：

TerritoryID	AvgSales	TotalSales
1	4097.9283	18825882.9554
2	23832.1197	8388906.1628
3	24843.2959	9564668.9505
4	4538.4759	28247474.5874
5	19683.4076	9566136.1287
6	4784.5226	19458653.7247
7	3094.5025	8268510.7753
8	2049.3143	5375351.5168
9	1613.1046	11038475.394
10	2672.6063	8603119.917

如上面的结果所示，该语句返回了“SalesOrderHeader ”表中每个“Territory ID”的平均值和总销售额。

现在，让我们看看另外两个集合函数：“MAX”函数和“MIN”函数。“MAX”函数返回一组值中的最大值，而“MIN”函数返回最小值。这两个函数都会忽略“null”值。在下面的语句中，我修改了前面的例子，使用了“MAX”函数和“MIN”函数：

```
SELECT
TerritoryID,
AVG (SubTotal) AS AvgSales,
SUM(SubTotal) AS TotalSales,
MAX(SubTotal) AS MaxSale,
MIN(SubTotal) AS MinSale
FROM
Sales.SalesOrderHeader
GROUP BY
TerritoryID
ORDER BY
TerritoryID
```

要注意，我仍然必须指定函数名，然后在后面的括号里写上列名。现在，该语句执行结果展示了每个“Territory ID”的最大和最小销售额，如下表所示：

TerritoryID	AvgSales	TotalSales	MaxSale	MinSale
1	4097.9283	18825882.9554	135653.7237	2.29
2	23832.1197	8388906.1628	179754.1225	1.374
3	24843.2959	9564668.9505	157915.6701	7.28
4	4538.4759	28247474.5874	224356.4831	2.29
5	19683.4076	9566136.1287	153465.5585	2.29
6	4784.5226	19458653.7247	187383.1452	2.29
7	3094.5025	8268510.7753	182344.2664	3.99
8	2049.3143	5375351.5168	132392.0669	3.99
9	1613.1046	11038475.394	82270.261	2.29
10	2672.6063	8603119.917	142274.0759	2.748

“MAX”函数和“MIN”函数与“AVG”函数和“SUM”函数还存在一种差异，就是你还可以对字符和日期数据使用“MAX”函数和“MIN”函数。当处理字符数据时，最大值和最小值是基于列的自然顺序计算的。例如，在下面的例子中，我会查询到“Product”表中产品列表的第一个名称和最后一个名称：

```
SELECT
MIN(Name) AS FirstProduct,
MAX(Name) AS LastProduct
FROM
Production.Product
```

“name”列被指定为“nvarchar”数据类型。当我对该列使用“MAX”函数或者“MIN”函数时，返回的最大值和最小值是基于该列名称的字母顺序计算的。因此，该语句返回该列表中的第一个产品名称和最后一个产品名称，如下表所示：

FirstProduct	LastProduct
Adjustable Race	Women's Tights, S

如果我对日期时间列使用“MIN”函数或者“MAX”函数，返回的值会基于保存在该列中日期时间值的最早值或者最晚值进行计算。

（作者：Robert Sheldon 译者：冯昀晖 来源：TT 中国）

原文标题：T-SQL 集合函数基础知识

链接：http://www.searchdatabase.com.cn/showcontent_35279.htm

利用 T-SQL 集合函数计算行数

SQL Server 还有另一个很有用的集合函数，那就是“COUNT”函数。“COUNT”函数返回一组值的数量。与其他集合函数不同，你可以在计数时指定是否包括“null”值。此外，你可以指定是计算所有值，还是只计算不同的值。下面的“SELECT”语句展示了“COUNT”函数计算一组值数量的几种用法：

```
SELECT
COUNT(*) AS ProductCount,
COUNT(ProductModelID) AS ModelCount,
COUNT(DISTINCT ProductModelID) AS DistinctCount
FROM
Production.Product
```

正如你所看到的，该语句利用“COUNT”函数得到了三个不同的值。第一个例子中，该函数利用规定格式“COUNT(*)”来获得该集合的行数，包括“null”值。在本例中，该集合包括整个“Product”表。因为表中有 504 行数据，所以该函数返回值为“504”，如下表所示：

ProductCount	ModelCount	DistinctCount
504	295	119

“COUNT”函数的第二个例子只是统计了“ProductModelID ”列值的数量。该用法返回的值不包括“null”值，因此返回值是“295”，而不是“504”，如上表所示。

“COUNT”函数的第三个用法包含了“DISTINCT”关键字，表示只有不重复的，非“null”的值才被计算在内。结果，该函数的返回值只有“119”。

注意：大部分集合函数支持“DISTINCT”关键字，所以你可以指定集合函数只在指定组中不重复的数据中执行。详细信息请参考 SQL Server 联机丛书中关于如何在各个函数中使用“DISTINCT”的内容。

除了“COUNT”集合函数，SQL Server 还支持“COUNT_BIG”函数。这两个函数的唯一差异就在于“COUNT”函数返回“int”类型值，“COUNT_BIG”返回“bigint”值。

（作者：Robert Sheldon 译者：冯昀晖 来源：TT 中国）

原文标题：利用 T-SQL 集合函数计算行数

链接：http://www.searchdatabase.com.cn/showcontent_35280.htm

利用 T-SQL 集合函数操作分组数据

在 T-SQL 语句中使用“GROUP BY”从句时，你可以使用“ROLLUP”，“CUBE”或者“GOURPING SETS”操作符来指定包含在结果集中的附加统计数据。这部分汇总数据在“GROUP BY”列表中表示为“null”值。

这是“null”值的一种特殊用法。然而，在“GROUP BY”列中指定的该列可以包含“null”值，(这种情况下)在该列的结果中也会显示为“null”。要区分这两种类型的“null”值，可以使用 SQL Server 提供的“GROUPING”集合函数。该函数可以明确指定“GROUP BY”从句中指定的列是否参与集合计算。

如果结果集中的值参与集合运算了，则“GROUPING”函数返回值“1”。否则，该函数返回值为“0”。我们来看一个例子，了解一下它的用法。在下面的“SELECT”语句中，我基于“Product”表的“Color”列对数据进行分组：

```
SELECT
Color,
AVG(ListPrice) AS AvgPrice,
COUNT(*) AS TotalAmount,
GROUPING(Color) AS AggGroup
FROM
Production.Product
GROUP BY
Color WITH ROLLUP
```

你可以看到，“GROUP BY”从句带有“WITH ROLLUP”操作符。因为“Color”是“GROUP BY”从句中指定的列，所以该列可以被用在“SELECT”列表的“GROUPING”函数中。

注意：“GROUPING”函数只能被用在“SELECT”列表，“HAVING”从句或者“ORDER BY”从句中。

“GROUPING”函数常常用来指明在“Color”列中显示的值是否是一个分组值，还是参与集合运算的累积值。下表显示了该“SELECT”语句的查询结果：

Color	AvgPrice	TotalAmount	AggGroup
NULL	16.8641	248	0
Black	725.121	93	0
Blue	923.6792	26	0
Grey	125.00	1	0
Multi	59.865	8	0
Red	1401.95	38	0
Silver	850.3053	43	0
Silver/Black	64.0185	7	0
White	9.245	4	0
Yellow	959.0913	36	0
NULL	438.6662	504	1

正如你所预料的，“AggGroup”列对那些颜色 color 本身(黑色 black，蓝色 blue，灰色 grey，等等)显示值为“0”。这表示颜色 color 值不参与分组计算;实际上，它仅仅是一组颜色。在“Color”列显示值“NULL”时，“GROUPING”的使用就变得很重要了。

第一个“NULL”的情况，“AggGroup”列返回值为“0”，表示这个“NULL”不是一个集合计算值，而是那些没有对颜色信息赋值的分组数据。

而“Color”列中的第二个“NULL”显示的“AggGroup”值为 1，这表示这是“ROLLUP”集合计算的一部分，该组中的所有合计都参与了集合计算。因此，你知道平均价格和数量是为展现整个表总计的一个准备。换句话说，该行包含了所有颜色分组的累计值。

(作者：Robert Sheldon 译者：冯昀晖 来源：TT 中国)

原文标题：利用 T-SQL 集合函数操作分组数据

链接：http://www.searchdatabase.com.cn/showcontent_35862.htm

利用 T-SQL 集合函数验证总和校验

SQL Server 还提供了“CHECKSUM_AGG”函数，该函数支持对一组数值运行总和校验。这在检测某个表数据的变化时很方便。然而，要注意的是该总和校验只能被用在整数表达式上，在执行计算时会忽略“null”值。

下面的“SELECT”语句包括两个使用“CHECKSUM_AGG”的例子，一个带有“DISTINCT”关键字，另一个没带：

```
SELECT
CHECKSUM_AGG(CAST(Quantity AS int)) AS ChecksumAgg,
CHECKSUM_AGG(DISTINCT CAST(Quantity AS int)) AS
ChecksumDistinct
FROM Production.ProductInventory;
```

该语句从“ProductInventory ”表提取数据。我们在“Quantity ”列执行总和校验，但是因为该列被配置为“smallint”数据类型，所以在运行总和校验之前，我必须把它转化成“int”数据类型。下表显示了该语句返回的查询结果：

ChecksumAgg	ChecksumDistinct
262	78

正如你所看到的，总和校验对“Quantity ”列中非 null 值计算的结果是“262”，而对于不重复的，非 null 的值计算结果是“78”。

（作者：Robert Sheldon 译者：冯昀晖 来源：TT 中国）

原文标题：利用 T-SQL 集合函数验证总和校验

链接：http://www.searchdatabase.com.cn/showcontent_35864.htm

利用 T-SQL 集合函数获取统计数据

SQL Server 有四个集合函数可以支持你对数值型列的一组值做数据统计：

- STDEV：返回值的统计标准方差。
- STDEVP：返回总体值的统计标准方差。
- VAR：返回值的统计方差。
- VARP：返回总体值的统计方差。

上面每个函数都会忽略“null”值，返回值都是浮点类型值。返回值表示该组值偏离平均值的程度(标准方差是方差的平方根)。抱歉的是，我不是专业学统计的，所以你要想了解更多关于如何计算标准方差和方差(以及统计群体与那些计算有多契合)的详细知识，那得去找本统计学方面的书好好查查了。与此同时，下面的语句展示了这些函数的用法：

```
SELECT
STDEV(ListPrice) AS StdevPrice,
STDEVP(ListPrice) AS StdevpPrice,
VAR(ListPrice) AS VarPrice,
VARP(ListPrice) AS VarpPrice
FROM
Production.Product
```

在上面的这个语句中，我用到了上面所有这四个集合函数，获取对“Product ”表的“ListPrice ”列的统计数据。下表展示了该语句返回的结果：

StdevPrice	StdevpPrice	VarPrice	VarpPrice
773.602842630487	772.834998428742	598461.358125969	597273.934796354

这就是在 SQL Server 中使用集合函数的全部知识。如你所见，在利用 T-SQL 提取数据时，不管你是否使用“GROUP BY”对数据进行分组,它们都是非常有用的利器。要了解关于这些函数的更多细节，请查阅 SQL Server 联机丛书可用的相关主题。(联机丛书上)每个主题带有一个附加示例来演示函数的用法。

(作者：Robert Sheldon 译者：冯昀晖 来源：TT 中国)

原文标题：利用 T-SQL 集合函数获取统计数据

链接：http://www.searchdatabase.com.cn/showcontent_35869.htm

利用 T-SQL 数学函数做‘三角函数’运算（上）

SQL Server 2008 中的 T-SQL 数学函数用来对数值表达式执行数值运算。例如，你可以利用数值函数计算角度的正弦值和余弦值，返回数值的平方或者平方根，或者按给定的精度做四舍五入。然而，要注意数学函数不能与算术运算符加(+)或者减(-)相混淆。

在本文中，我将分三部分描述 SQL Server 2008 中支持的数学函数：三角函数，代数函数和数值函数。这种组织方式是为解释这些函数并对特定的函数互相比较提供一个粗略的结构。这些函数既不彼此依赖，也不需要被以特殊的顺序展示。

在 T-SQL 中使用三角函数

我们要了解的第一组函数是三角函数，这些函数涉及到角度，弧度，度数，以及 pi(π)的值。首先要介绍的这两个函数就属于这种类别的，它们是“SIN”和“ASIN”。“SIN”函数返回指定角度的正弦值。“ASIN”函数(也称为反正弦函数)返回指定正弦值的角度。

接下来，我们通过一个例子看看怎样使用“SIN”函数和“ASIN”函数。在下面的这组语句中，我声明了两个变量，并给它们赋值，然后在一个“SELECT”语句中调用了它们：

```
DECLARE @angle1 float
SET @angle1 = 52.64
DECLARE @sine float
SET @sine = SIN(@angle1)
SELECT
@sine AS Sine,
ASIN(@sine) AS Arcsine
```

首先，我定义了一个叫做“@angle1”的变量，设置为浮点类型。然后，我把它的值赋值为“52.46”。

接下来，我创建了一个叫做“@sine”的变量，它也是一个浮点类型值，然后设置“@sine”变量的值为用“SIN”函数计算“@angle1”变量的正弦值。“SIN”函数要求传递的参数是一个浮点类型值或者是可以隐式转换为浮点类型的值。

注意：对于任何要求浮点表达式作为参数的函数，传入的参数必须是浮点类型或者是可以隐式转换为浮点类型的值。

最后，我使用“SELECT”语句获取“@sine”的值和该值的反正弦值。要计算反正弦值，我使用“ASIN”函数计算正弦值的角度。既然该函数计算正弦值的角度，那么传递给该函数的值必须在“-1”到“1”的范围内。该函数返回该角度的弧度值。下面的结果展示了“52.64”的正弦值以及该正弦值的反正弦值：

Sine	Arcsine
0.694032454911975	0.767075111026485

(作者：Robert Sheldon 译者：冯昀晖 来源：TT 中国)

原文标题：利用 T-SQL 数学函数做‘三角函数’运算（上）

链接：http://www.searchdatabase.com.cn/showcontent_37214.htm

利用 T-SQL 数学函数做‘三角函数’运算（下）

另一组与“SIN”和“ASIN”类似的函数是“COS”函数和“ACOS”函数。“COS 函数返回指定角度的余弦值，而”ACOS”函数(被称为反余弦函数)返回指定余弦值的角度。下面的例子展示了这两个函数的用法：

```
DECLARE @angle2 float
SET @angle2 = 52.64
DECLARE @cosine float
SET @cosine = COS(@angle2)
SELECT
@Cosine AS Cosine,
ACOS(@cosine) AS Arccosine
```

首先，我声明“@angle2”变量为浮点类型，然后给它赋值为“52.64”。接下来我声明了“@cosine”变量，也是浮点类型，然后利用“COS”函数把“@angle2”的余弦值赋值给“@cosine”。最后，我创建了一个“SELECT”语句，计算“@cosine ”的值和它的反余弦值。要注意，我利用“ACOS”函数计算反余弦值。该反余弦值显示为弧度，如下面的结果列表所示：

Cosine	Arccosine
-0.719943714139416	2.37451754256331

正如上面的结果所示，“52.64 ”的余弦值是“-0.719943714139416”，而该余弦值的反余弦值是“2.37451754256331”。

另外两个与前面的函数类似的函数是“TAN”和“ATAN”。“TAN”函数返回浮点表达式的正切值，而“ATAN”函数返回正切值对应的角度(以弧度方式展现)。下面的例子展示了这两个函数的用法：

```
DECLARE @angle3 float
SET @angle3 = 52.64
DECLARE @tangent float
SET @tangent = SIN(@angle3)
SELECT
@tangent AS Tangent,
ATAN(@tangent) AS Arctangent
```

如你所见，这个例子与前两个例子几乎是一样的，只有返回的正切值和反正切值不一样，结果如下表所示：

Tangent	Arctangent
0.694032454911975	0.606709662224033

SQL Server 还支持“COT”函数，它返回指定角度的余切值。请看下面的例子：

```
DECLARE @angle4 float
SET @angle4 = 52.64
DECLARE @cotangent float
SET @cotangent = COT(@angle4)
SELECT @cotangent AS Cotangent
```

在这个例子中，我声明了“@angle4”变量，并赋值为“52.64”。然后，我声明了“@cotangent”变量，并把计算“@angle4”的余切值赋值给它。接下来，我用一个“SELECT”语句调用“@cotangent”变量。该语句返回结果如下：

Cotangent
-1.03733436245532

现在，我们来看看“ATN2”函数。这个函数计算正 x 轴与原点到指定点(用“x，y”表示)组成直线之间的夹角。下面的例子展示了它的用法：

```
DECLARE @x float
SET @x = 52.64
DECLARE @y float
SET @y = 192.79
SELECT ATN2(@x, @y) AS Angle
```

要注意，我事先已经声明了“@x”和“@y”变量并给它们赋值了，然后在“ATN2”函数中把它们用作参数。接下来，我在“SELECT”语句中使用“ATN2”函数计算这个夹角，结果如下表所示：

Angle
0.266546088052137

如你所见，x 轴与“x，y”定义的射线之间的夹角是“0.266546088052137”度。

在 SQL Server 中，你可以利用“RADIANS ”函数把角度转换为弧度，利用“DEGREES ”函数把弧度转换为角度，如下面的例子所示：

```
DECLARE @degrees float
SET @degrees = 22.5
DECLARE @radians float
SET @radians = RADIANS(@degrees)
SELECT
@radians AS [Radians],
DEGREES(@radians) AS [Degrees]
```

要执行该函数，我首先声明了“@degrees”变量，并给它赋值“22.5”度。接下来，我声明了“@radians”变量，然后利用“RADIANS”函数把“@degrees”的值转换为弧度，并赋给“@radians”变量。然后我定义了“SELECT”语句来计算“@radians”的值。接下来，我使用“DEGREES”函数把“@radians”值转换回角度，如下面结果所示：

Radians	Degrees
0.392699081698724	22.5

如你所见，“22.5”度转换成了“0.392699081698724”弧度，而我把该弧度转换回来成角度时，仍然是“22.5”。

我想说明的另一个函数是“PI”，它只是计算“π”的值，请看下面的示例：

```
DECLARE @pi float
SET @pi = PI()
SELECT
@pi AS [PI],
DEGREES(@pi/2) AS [Degrees]
```

首先，我利用“PI”函数设置“@pi”变量的值为“π”，然后在“SELECT”语句中使用该函数，我计算了“π”的值，并把“π/2”转换为角度。该语句返回的结果如下所示：

PI	Degrees
3.14159265358979	90

除了显示“π”的值，该结果显示了“π/2”是 90 度。

(作者：Robert Sheldon 译者：冯昀晖 来源：TT 中国)

原文标题：利用 T-SQL 数学函数做‘三角函数’运算（下）

链接：http://www.searchdatabase.com.cn/showcontent_37217.htm

T-SQL 代数函数与数值函数的用法（上）

本系列文章的第一部分分析了 SQL Server 2008 中 T-SQL 三角函数的基础知识，并举例说明了如何使用这些函数。接下来这一节详细讲解了其它数学函数：代数函数和数值函数，我们分别举例加以说明。

使用代数函数

在本文中，我要介绍的 T-SQL 函数具体包括计算平方根，指数，或者数值的对数。前两个这种类型的函数是“SQUARE ”和“SQRT”。“SQUARE ”函数返回一个数值的平方，而“SQRT ”函数返回平方根。请看下面的例子：

```
DECLARE @root1 float
SET @root1 = 4
SELECT
SQUARE(@root1) AS [Square],
SQRT(@root1) AS SquareRoot
```

首先，我声明“@root1”变量为浮点类型，然后给该变量赋值为“4”。接着，我使用“SELECT”语句对该变量运行“SQUARE”函数和“SQRT”函数。该“SELECT”语句返回如下结果：

Square	SquareRoot
16	2

正如你所预期的，“4”的平方是“16”，而平方根是“2”。

接下来，我们来看一个例子，其中用到了 EXP，LOG 和 LOG10 函数。“EXP”函数返回一个浮点表达式的指数值，“LOG”函数返回数值的对数值，“LOG10”函数返回基于 10 的对数值(常对数)。下面的例子使用了这三个函数：

```
DECLARE @root2 float
SET @root2 = 4
SELECT
EXP(@root2) AS Exponential,
LOG(@root2) AS [Log],
LOG10(@root2) AS [Log10]
```

在我声明并为“@root2”变量赋值以后，我定义了一个“SELECT”语句来调用这三个函数。对于每个函数，我都传递“@root2”变量为参数。该语句返回如下结果：

Exponential	Log	Log10
54.5981500331442	1.38629436111989	0.602059991327962

如你所见，“@root2”被赋值为“4”，该语句返回“@root2”变量值的指数，对数和常对数。

我们要讲的最后一个代数函数是“POWER”，它计算指定浮点表达式值的指定幂次方。“POWER”函数有两个参数：浮点表达式底数和表达式要计算的乘方数。在下面的例子中，我要计算 2 的 6 次幂：

```
DECLARE @base float
SET @base = 2
DECLARE @power float
SET @power = 6
SELECT POWER(@base, @power) AS [Power]
```

要执行这个函数，我声明了两个浮点型变量。我用“@base”作为“POWER ”函数的第一参数，用“@power”变量作为第二个参数。该语句返回如下结果：

Power
64

如你所见，2 的 6 次幂是 64。

（作者：Robert Sheldon 译者：冯昀晖 来源：TT 中国）

原文标题：T-SQL 代数函数与数值函数的用法（上）

链接：http://www.searchdatabase.com.cn/showcontent_37564.htm

T-SQL 代数函数与数值函数的用法（下）

使用 T-SQL 的数值函数

按照我们前面的分类，我们要介绍的最后一类函数是数值处理函数。总的来说，这些函数都是以某种方式修改指定数值，比如对数值做四舍五入或者计算它的绝对值。

我们先介绍三个这种类型的函数，它们是：“ROUND”，“FLOOR”和“CEILING”。“ROUND”函数对数值做指定精度的四舍五入，而“FLOOR”函数返回小于或等于指定值的最大整数(向下取整)。“CEILING”函数返回大于或等于指定值的最小整数(向上取整)。下面的例子展示了这三个函数的用法：

```
DECLARE @exp1 float
SET @exp1 = 54.784392
SELECT
ROUND(@exp1, 2) AS [Round],
FLOOR(@exp1) AS [Floor],
CEILING(@exp1) AS [Ceiling]
```

首先，我声明了“@exp1”变量，并给它赋值“54.784392”。然后，我在“SELECT”语句中调用了每个函数。要注意，“ROUND”函数有两个参数：第一个参数是要处理的数值，在本例中就是“@exp1”变量;第二个参数是“2”，表示应该四舍五入保留的精度，它表示小数点后保留的位数。

“FLOOR”函数和“CEILING”函数都有一个参数，本例中就是“@exp1”变量。该“SELECT”语句返回的结果如下：

Round	Floor	Ceiling
54.78	54	55

接下来我们要介绍的函数是“ABS”函数，它返回一个数值表达式的绝对正值。该函数返回的结果取决于数值表达式是正的还是负的，或者是零。下面的例子展示了这三种情况：

```
DECLARE @abs1 float
SET @abs1 = 1.93
DECLARE @abs2 float
```

```
SET @abs2 = 0
DECLARE @abs3 float
SET @abs3 = -1.93
SELECT
ABS(@abs1) AS Absolute1,
ABS(@abs2) AS Absolute2,
ABS(@abs3) AS Absolute3
```

要注意，我声明并设置了三个变量，并在“SELECT”语句中调用了三次“ABS”，每个变量调用一次。该语句返回结果如下：

Absolute1	Absolute2	Absolute3
1.93	0	1.93

如你所见，值为“1.93”的变量值和值为“-1.93”的变量值返回的结果是相同的，而值为“0”的变量返回值仍然是“0”。

另一个与数值表达式正负号有关的函数是“SIGN”。如果输入值是正数的话，该函数返回“+1”，如果输入是“0”则返回“0”，而如果输入值是负数则返回“1”。下面的例子展示了这三种情况：

```
DECLARE @sign1 float
SET @sign1 = 1.93
DECLARE @sign2 float
SET @sign2 = 0
DECLARE @sign3 float
SET @sign3 = -1.93
SELECT
SIGN(@sign1) AS Sign1,
SIGN(@sign2) AS Sign2,
SIGN(@sign3) AS Sign3
```

我声明了三个变量并给它们赋值，然后我在“SELECT”语句中使用了“SIGN”函数，返回每个变量值的正负号。结果如下所示：

Sign1	Sign2	Sign3
1	0	-1

如你所见，值为“1.93”的变量返回值为“1”，值为“0”的变量返回值为“0”，而值为“-1.93”的变量返回值为“-1”。

我们要讲的最后一个函数是“RAND”。该函数返回 0 到 1 之间的随机数。下面的语句展示了该函数的用法：

```
SELECT RAND()
```

每次你运行这个语句时，它会返回 0 到 1 范围内的不同值。然而，如果你在调用“RAND”函数时指定了随机种子，情况就不一样了。例如，在下面的“SELECT”语句中“RAND”函数就使用“10”作为随机数种子：

```
SELECT RAND(10)
```

每次你(在一次连接中)使用这个随机数种子调用“RAND”函数时，它会返回相同的结果。例如，我在一次连接中重复运行了上面这个“SELECT”语句，每次都会得到下面这个值：

```
0.713759689954247
```

注意，这个值仍然在“0”到“1”之间。

T-SQL 数学函数总结

如你所见，在需要对数值做各种计算时，你可以使用数学函数做各种类型的运算。在前面的例子中，我尽量演示了这些函数的用法，以及它们返回的数据类型。要注意，这些例子并没有考虑到这些函数之间的互相调用。为此，我推荐你在 SQL Server 联机帮助中查找“函数”主题来了解更多细节。到目前为止，你应该已经掌握了使用 T-SQL 数学函数的良好基础。

(作者：Robert Sheldon 译者：冯昀晖 来源：TT 中国)

原文标题：T-SQL 代数函数与数值函数的用法（下）

链接：http://www.searchdatabase.com.cn/showcontent_37566.htm

在 SQL Server 中如何使用子查询（上）

SQL Server 支持的 T-SQL 语言特性之一是子查询(subquery)，它可以内嵌在一个 SELECT、INSERT、UPDATE 或 DELETE 语句中。你可以定义一个 T-SQL 子查询，然后将其放在 SQL 语句中任何一个允许使用表达式的地方。例如，你可以在一个 SELECT 语句的 SELECT 列表中内嵌一个子查询，或者在 WHERE 子句中包含一个子查询作为一个查询条件。

当把一个 T-SQL 子查询内嵌到 SQL 语句的时候，你必须用括号把它括起来。此外，子查询不能包含在 COMPUTE 或者 FOR BROWSE 子句，而且也不能包含在 ORDER BY 子句，除非在 SELECT 子句中使用了 TOP 运算符。

注意：有时，子查询又被称为内部选择或内部查询。嵌入子查询的语句被称为外部选择或外部查询。

你可以将包含子查询的语句改写为用连接方式表达的语句。在某些情况下，这将会提高性能，尤其是你必须通过子查询返回的大的结果集来遍历由外部查询返回的每一行。在过去的文章中，我写了很多关于在 SQL Server 中创建连接的基础知识。

使用 T-SQL 子查询作为表达式

在一个 T-SQL 语句中包含子查询是一种常见的方式，就像在 SELECT 列表中使用列表达式一样。你只须简单地像其他列表达式一样包括子查询。让我们来看几个例子说明子查询是如何工作的。

注意：本文中的例子是基于 AdventureWorks2008 示例数据库，其被安装在本地的 SQL Server 2008 实例中。

在下面的例子中，我定义了一个返回销售代表平均销售金额的子查询：

```
SELECT
FirstName + ' ' + LastName AS FullName,
ROUND(SalesYTD, 2) AS SalesYTD,
(
SELECT ROUND(AVG(SalesYTD), 2)
FROM Sales.vSalesPerson
WHERE JobTitle = 'Sales Representative'
) AS AvgSales
```

```
FROM
Sales.vSalesPerson
WHERE
BusinessEntityID = 275
```

SELECT 列表中第三个列表达式就是子查询。请注意，它是用括号括起来的。然后我定义了一个叫 AvgSales 的列别名。如果你单独运行这个子查询语句，那么将返回一个 3,054,352.75 的数字。由于子查询被定义为一个列表达式，那么返回值就是外部语句结果集的一个列值，如下表所示：

FullName	SalesYTD	AvgSales
Michael Blythe	4557045.05	3054352.75

在前面的例子中，子查询返回一个独立于外部语句的值。换句话说，你不需要任何外部语句的数据就可以从子查询中检索到你要的结果。你只需要知道所有销售代表的平均销售额。

(作者：Robert Sheldon 译者：冯昀晖 来源：TT 中国)

原文标题：在 SQL Server 中如何使用子查询（上）

链接：http://www.searchdatabase.com.cn/showcontent_38201.htm

在 SQL Server 中如何使用子查询（下）

有些时候，你需要外部语句能够告诉子查询一些信息。在这种情况下，可以使用关联子查询，即包含一个把外部语句和子查询语句链接起来的查询条件。例如：下面 SELECT 语句的子查询通过 ProductSubcategoryID 列链接到外部语句：

```
SELECT
p.ProductID,
p.Name,
(
SELECT ps.Name
FROM Production.ProductSubcategory ps
WHERE p.ProductSubcategoryID = ps.ProductSubcategoryID
) AS SubcategoryName
FROM Production.Product p
WHERE p.ProductSubcategoryID IS NOT NULL
AND p.Name LIKE '%seat%'
```

在这个子查询的 WHERE 子句中，我把子查询语句的 ProductSubcategory 表的 ProductSubcategoryID 列和外部语句 Product 表的 ProductSubcategoryID 列关联起来。请注意，我使用表的别名来引用表(用 p 代表 Product 表和用 ps 代表 ProductSubcategory 表）。这允许我在内部查询中引用外部表。

由于使用关联子查询，子查询基于 ProductSubcategoryID 列返回正确的 ProductSubcategory 名称。下表显示由 SELECT 语句返回的结果集：

ProductID	Name	SubcategoryName
908	LL Mountain Seat/Saddle	Saddles
909	ML Mountain Seat/Saddle	Saddles
910	HL Mountain Seat/Saddle	Saddles
911	LL Road Seat/Saddle	Saddles
912	ML Road Seat/Saddle	Saddles
913	HL Road Seat/Saddle	Saddles
914	LL Touring Seat/Saddle	Saddles
915	ML Touring Seat/Saddle	Saddles
916	HL Touring Seat/Saddle	Saddles

我在前面的例子中使用表别名，能够很容易地确定 ProductSubcategoryID 列属于哪些表，但并不总是需要使用表别名。默认情况下，子查询假定一个未限定的列名属于在子查询中指定的表。如果子查询表不包括该列，那么就假定该表属于外部语句中的表。

所有这一切意味着，如果一个列在子查询的表和外部语句的表中都存在，你就必须限定该列属于你希望的外部语句中的表，如下面的例子所示：

```
SELECT
ProductID,
Name,
(
SELECT Name
FROM Production.ProductSubcategory
WHERE ProductSubcategoryID = Production.Product.
ProductSubcategoryID
) AS SubcategoryName
FROM Production.Product
WHERE ProductSubcategoryID IS NOT NULL
AND Name LIKE '%seat%'
```

正如你所看到的，我没有使用表别名。相反，我用架构名加上表名来限定子查询中第二个列名使用的是外部语句的表。该语句返回与上例相同的结果。

注意：你可能已经注意到，在第一个例子的子查询中引用了与外部查询相同的表。但是，在随后的两个例子中，子查询引用的表和外部语句的表不同。你可以采取两种方法在语句中添加子查询。

到目前为止我们研究的这些例子都是将 SELECT 语句作为外部语句，但你也可以将子查询用在 INSERT、UPDATE 和 DELETE 语句中。例如，下面的 INSERT 语句中子查询用来确定将值插入到一列中：

```
INSERT INTOProduction.Illustration (Diagram)
VALUES
(
(
SELECT Diagram
FROM Production.Illustration
WHERE IllustrationID = 7
)
)
```

子查询返回一个 XML 值，然后插入到 Ilustration 表的 Diagram 列。请注意，子查询要用一对括号括起来，另外一对括号则是将所有的值括起来。你在 INSERT 语句中使用子查询返回一个值，就像在语句中使用的其他任何值表达式一样。

你还可以在 DELETE 语句中包含子查询。例如，下面的例子使用子查询来确定 Ilustration 表中插入的最新一行：

```
DELETE Production.Illustration
    WHERE IllustrationID =
                (
    SELECT MAX(IllustrationID)
    FROM Production.Illustration
                )
```

正如你所看到的，子查询作为 WHERE 子句查询条件的一部分。当 IllustrationID 值等于表中 IllustrationID 最大值时，该行被删除。

（作者：Robert Sheldon 译者：冯昀晖 来源：TT 中国）

原文标题：在 SQL Server 中如何使用子查询（下）

链接：http://www.searchdatabase.com.cn/showcontent_38202.htm

T-SQL 子查询与操作符的组合：比较运算符

使用子查询和比较运算符

在 WHERE 子句中将子查询作为查询条件是一个常见的场景，特别是与比较运算符一起使用。让我们来看几个使用比较运算符和子查询的 SELECT 语句例子。在下面的 SELECT 语句中，我使用子查询来查找“brakes”子类的 ProductSubcategoryID 值：

```
SELECT
ProductID,
Name,
ProductSubcategoryID
FROM Production.Product
WHERE ProductSubcategoryID =
(
SELECT ProductSubcategoryID
FROM Production.ProductSubcategory
WHERE Name = 'brakes'
)
```

在这个例子中，子查询返回值等于 6 的 ProductSubcategoryID。然后该值与 Product 表中的 ProductSubcategoryID 列值相比较。如果值相等，则该行被返回。请注意，我使用等于(=)比较运算符用来比较 ProductSubcategoryID 列值和子查询结果。下表显示了由外部 SELECT 语句返回的结果集：

ProductID	Name	ProductSubcategoryID
907	Rear Brakes	6
948	Front Brakes	6

你还可以使用其他比较运算符用来比较一个列值和子查询的结果集。例如，下面的语句使用不等于(<>)比较运算符：

```
SELECT
ProductID,
Name,
ProductSubcategoryID
FROM Production.Product
WHERE ProductSubcategoryID <>
```

```
(
SELECT ProductSubcategoryID
FROM Production.ProductSubcategory
WHERE Name = 'mountain bikes'
)
```

在这个例子中，子查询返回“Mountain Bikes”子类的 ProductSubcategoryID 值。因此，外部的 SELECT 语句将返回除了那些 ProductSubcategoryID 列值不等于 1 的所有行。

在下例中，我使用大于(>)比较运算符比较 SalesYTD 列值和子查询返回值：

```
SELECT
FirstName + ' ' + LastName AS FullName,
ROUND(SalesYTD, 2) AS SalesYTD
FROM
Sales.vSalesPerson
WHERE
SalesYTD >
(
SELECT AVG(SalesYTD)
FROM Sales.vSalesPerson
WHERE JobTitle = 'Sales Representative'
)
```

这个子查询返回所有销售代表的平均销售金额 3,054,352.7524。因此，外部语句的结果集是销售额超过这一金额的所有记录，如下表所示：

FullName	SalesYTD
Michael Blythe	4557045.05
Linda Mitchell	5200475.23
Jillian Carson	3857163.63
José Saraiva	3189356.25
David Campbell	3587378.43
Jae Pak	5015682.38
Ranjit Varkey Chudukatil	3827950.24

当你使用比较运算符比较一个值和子查询时，该子查询必须返回单值，除非还包括 ALL 或 ANY 运算符。对于 ALL 运算符来说，被比较的列值必须大于子查询返回的所有的值。对于 ANY 运算符来说，被比较的列值大于子查询返回的任何一个值即可。

在下例中，我使用 ALL 运算符比较 SalesYTD 列值和子查询返回值：

```
SELECT
FirstName + ' ' + LastName AS FullName,
JobTitle,
ROUND(SalesYTD, 2) AS SalesYTD
FROM
Sales.vSalesPerson
WHERE
SalesYTD > ALL
(
SELECT AVG(SalesYTD)
FROM Sales.vSalesPerson
GROUP BY JobTitle
)
```

因为我使用了 ALL 运算符，子查询可以返回多个值。在这个例子中，子查询返回每个工作组的平均销售数量。下表显示了子查询返回的结果集：

636440.251
677558.4653
219088.8836
3054352.7524

SalesYTD 列值必须大于所有子查询返回的值。这意味着 SalesYTD 列值必须超过 3,054,352.7524。下表显示了外部 SELECT 语句返回的结果集：

FullName	JobTitle	SalesYTD
Michael Blythe	Sales Representative	4557045.05
Linda Mitchell	Sales Representative	5200475.23
Jillian Carson	Sales Representative	3857163.63
José Saraiva	Sales Representative	3189356.25
David Campbell	Sales Representative	3587378.43
Jae Pak	Sales Representative	5015682.38
Ranjit Varkey Chudukatil	Sales Representative	3827950.24

正如你所看到的，只有销售额超过 3,054,352.7524 的销售代表才会包括在结果中。

（作者：Robert Sheldon 译者：冯昀晖 来源：TT 中国）

原文标题：T-SQL 子查询与操作符的组合：比较运算符

链接：http://www.searchdatabase.com.cn/showcontent_38349.htm

T-SQL 子查询与操作符的组合：IN、NOT IN

除了在 WHERE 子句中使用包含比较运算符的子查询，你可以使用 IN 或 NOT IN 运算符。这些运算符让你比较一个值与由子查询返回的零或更多的值。对于 IN 运算符来说，如果该值不在子查询结果集中，查询条件将返回 false 且该行不包括在内。否则，一个 true 被返回，行被包括在内。(NOT IN 运算符正好与之相反)

在下例中，我使用 IN 运算符来比较 TerritoryID 列值和子查询返回值：

```
SELECT
BusinessEntityID,
TerritoryID,
SalesQuota,
SalesYTD
FROM
Sales.SalesPerson
WHERE
TerritoryID IN
(
SELECT TerritoryID
FROM Sales.SalesTerritory
WHERE Name = 'northeast'
OR Name = 'northwest'
)
ORDER BY TerritoryID, BusinessEntityID
```

子查询返回东北地区和西北地区的 TerritoryID 值，分别是 2 和 1。因此，在 SalesPerson 表中的 TerritoryID 列值必须等于 1 或 2 才会被包括在外部表的结果集中，如下表所示：

BusinessEntityID	TerritoryID	SalesQuota	SalesYTD
280	1	250000.00	0.00
283	1	250000.00	3587378.4257
284	1	300000.00	1931620.1835
275	2	300000.00	4557045.0459

下面的例子是先前那个相同的，只是改用了 NOT IN 运算符：

```
SELECT
BusinessEntityID,
TerritoryID,
SalesQuota,
SalesYTD
FROM
Sales.SalesPerson
WHERE
TerritoryID NOT IN
(
SELECT TerritoryID
FROM Sales.SalesTerritory
WHERE Name = 'northeast'
OR Name = 'northwest'
)
ORDER BY TerritoryID, BusinessEntityID
```

现在，外部语句返回除了 TerritoryID 值等于 1 或 2 之外的所有行，如下表所示：

BusinessEntityID	TerritoryID	SalesQuota	SalesYTD
277	3	250000.00	3857163.6332
276	4	250000.00	5200475.2313
281	4	250000.00	3018725.4858
279	5	300000.00	2811012.7151
278	6	250000.00	1764938.9859
282	6	250000.00	3189356.2465
290	7	250000.00	3827950.238
288	8	250000.00	2241204.0424
286	9	250000.00	1758385.926
289	10	250000.00	5015682.3752

你也可以在除 SELECT 之外的语句中使用 IN 或 NOT IN 运算符。例如，下面的 UPDATE 语句中比较 ModifiedDate 列值和子查询返回值：

```
UPDATEProduction.Illustration
SET Diagram = NULL
WHERE ModifiedDate IN
(
SELECT ModifiedDate
FROM Production.Illustration
```

```
        WHERE ModifiedDate > '2005-01-01 00:00:00.000'
                                    )
```

子查询返回所有的修改日期晚于 2005 年 1 月 1 日的记录行。如果 Illustration 表中 ModifiedDate 列值小于该日期的行就会被返回，该行的 Diagram 列值被更新为 NULL。否则，不改变。

(作者：Robert Sheldon 译者：冯昀晖 来源：TT 中国)

原文标题：T-SQL 子查询与操作符的组合：IN、NOT IN

链接：http://www.searchdatabase.com.cn/showcontent_38351.htm

T-SQL 子查询与操作符的组合：EXIST、NOT EXIST

你可以在 WHERE 子句中使用另一套 EXIST 和 NOT EXIST 运算符和子查询。EXIST 运算符检查子查询是否返回值。如果一个或多个值返回，则该行返回;否则不是。(NOT EXIST 运算符与之恰恰相反)

在下面的 SELECT 语句中，检查是否存在一个叫作“brake”的产品名称：

```
SELECT
ProductID,
Name,
ProductModelID
FROM Production.Product
WHERE EXISTS
(
SELECT *
FROM Production.ProductModel
WHERE ProductModelID = Production.Product.
ProductModelID
AND Name LIKE '%brakes%'
)
```

首先注意的是，子查询的 SELECT 语句在 SELECT 列表中使用了星号(*)。既然你仅仅检查是否存在一个返回行，列名是没有必要的。还要注意，我创建了一个关联子查询以便外部查询和子查询的 ProductModelID 值可以匹配。因此，每一行必须匹配 ProductModelID 列值且名称必须包括“brakes”。如果子查询返回一个值，EXIST 运算符返回 true，并且该行被返回。下表显示了外部语句返回的结果集：

ProductID	Name	ProductModelID
907	Rear Brakes	128
948	Front Brakes	102

你也可以很轻松地使用 NOT EXIST 运算符检查是否有结果返回。下面的例子与先前那个相同，除了使用 NOT EXIST 运算符：

```
SELECT
ProductID,
Name,
```

```
                ProductModelID
            FROM Production.Product
            WHERE NOT EXISTS
                (
                SELECT *
            FROM Production.ProductModel
        WHERE ProductModelID = Production.Product.
                ProductModelID
            AND Name LIKE '%brakes%'
                )
```

现在的语句返回 502 行，而不是 2 行;每次子查询不返回结果，结果集就返回一行。

正如上面的例子所示，当你需要使用数据子集的时候，子查询提供了很大的灵活性。有关子查询的详细信息以及更多范例，请参阅 SQL Server 联机丛书。此外，一定要记住，在某些情况下使用子查询可能是一个更好的解决方案。在此期间，尝试了不同类型的子查询和多变的复杂性，以更好地理解它们的作用。

(作者：Robert Sheldon 译者：冯昀晖 来源：TT 中国)

原文标题：T-SQL 子查询与操作符的组合：EXIST、NOT EXIST

链接：http://www.searchdatabase.com.cn/showcontent_38352.htm

T-SQL 系统函数：查看数据修改信息

SQL Server 内建有可以支持你执行各种任务的函数，比如转换数据，操作字符串值，以及执行数学计算的函数。一些 SQL Server 函数被归类为系统函数，这些函数支持你查看关于更新和插入操作的信息，提取服务器属性设置，访问当前会话及其用户的详细信息。你还可以利用系统函数执行许多语言相关的任务。

本文将为你介绍 SQL Server 中包含的许多系统函数。尽管我不能覆盖到每一个函数，我会描述尽可能多的函数，并介绍不同类型的函数以便给你一个全面的基础了解。

注意：本文中有一些例子，我在本地 SQL Server 2008 实例已安装的“AdventureWorks2008”示例数据库环境中运行了这些例子。你也可以在 SQL Server2005 上运行这些例子，但是要确保根据需要修改数据库名称。同时，如果你在 SQL Server 2005 上运行这些例子，你的结果中有一些会与本文中展示的结果有细微的差异。

查看有关数据修改的信息

SQL Server 的一些系统函数支持获取你的更新和插入操作的明细信息。例如，你可以获取插入到某个表中的最后一个标识值，或者对某个表增加或者更新的行数。要展示这些函数如何使用，我定义了下面的表并给它填充了几行数据：

```
USE AdventureWorks2008;
IF OBJECT_ID('TableA', 'U') IS NOT NULL
DROP TABLE dbo.TableA;
CREATE TABLE dbo.TableA
(
ColA INT IDENTITY(101, 100) NOT NULL,
ColB VARCHAR(20) NOT NULL
);
INSERT INTO TableA (ColB) VALUES('red');
INSERT INTO TableA (ColB) VALUES('blue');
INSERT INTO TableA (ColB) VALUES('green');
INSERT INTO TableA (ColB) VALUES('yellow');
INSERT INTO TableA (ColB) VALUES('orange');
SELECT * FROM TableA;
```

下面是非常简单的一个表，该表中填充了五行数据。请注意“ColA”列被定义为一个身份列，起始值为“101”，增量为“100”。最后一个“SELECT 语句”执行后产生如下结果：

ColA	ColB
101	red
201	blue
301	green
401	yellow
501	orange

现在，我们来看一个例子，其中使用了几个系统函数提取关于刚插入上面那个表的数据相关的信息。下面的“SELECT 语句”包括五个函数，其中三个与身份列值有关，而另两个与行数有关：

```
SELECT
IDENT_CURRENT('TableA') AS LastValue,
IDENT_SEED('TableA') AS SeedValue,
IDENT_INCR('TableA') AS IncrValue,
@@ROWCOUNT AS RowsAffected,
ROWCOUNT_BIG() AS BigRowsAffected;
```

第一个函数是“IDENT_CURRENT”，它提取插入表“TableA”的最后一个身份值。如你所见，表“TableA”被指定为该函数的一个入参。该函数返回的信息与添加数据时的会话无关，也与添加时执行的语句范围无关。这一点非常重要，因为 SQL Server 支持其他身份相关的函数，包括专门针对当前会话(@@IDENTITY)和当前范围和会话(SCOPE_IDENTITY)的函数。

上面的例子中接下来的两个函数返回指定表中身份列的设置信息。“IDENT_SEED”函数返回起始值，而“IDENT_INCR”函数返回增量值。下面的结果集包括由这三个身份相关函数返回的值：

LastValue	SeedValue	IncrValue	RowsAffected	BigBigRowsAffected
501	101	100	5	5

正如你所预料的，插入表“TableA”的最后一个身份值是“501”，起始值是“101”，而增量值是“100”。

现在，我们再来看看本例中接下来的两个函数。“@@ROWCOUNT ”函数返回最后一个语句影响的行数。“ROWCOUNT_BIG”函数也做同样的事情，唯一区别是返回值类型是“bigint”。如果你预计影响的行数会多于 20 亿条数据，你应该使用这个函数。

要注意，这两个函数都不是关联于某个表的，因为这两个函数基于最后一条执行的语句获取数据。也就是说，如果给某个表插入了 200 行，该函数会返回 200。如果“SELECT”语句返回 30 行，那么该函数返回值就是 30。在上面的例子中，两个函数返回的值都是 5，因为最后运行的是一条“SELECT”语句，该语句返回 5 行数据。

（作者：Robert Sheldon 译者：冯昀晖 来源：TT 中国）

原文标题：T-SQL 系统函数：查看数据修改信息

链接：http://www.searchdatabase.com.cn/showcontent_39615.htm

T-SQL 系统函数：查看服务器属性

本系列文章关注的是 T-SQL 系统函数，详细介绍了如何查看有关服务器属性和用户的详细信息。

请查看该系列文章第一篇：T-SQL 系统函数：查看数据修改信息

查看服务器属性

有时，你可能想要检索特定的服务器属性值，这种情况下，你可以使用“SERVERPROPERTY”函数。该函数有一个参数(属性名称)，请看下面的示例：

```
SELECT
SERVERPROPERTY('Edition') AS SrvEdition,
SERVERPROPERTY('InstanceName') AS SrvInstance,
SERVERPROPERTY('Collation') AS SrvCollation;
```

在这个“SELECT”语句中，我检索了“Edition”，“InstanceName ”和“Collation ”属性的设置，它们以参数的形式传递到函数中。该“SELECT”语句返回下面结果：

SrvEdition	SrvInstance	SrvCollation
Developer Edition	SQLSRV2008	SQL_Latin1_General_CP1_CI_AS

你的结果可能有所不同，特别是前两个参数，但是你可以理解这个意思。该函数使你可以简单地获取这些设置信息。

注意：在使用必须传递指定属性作为入参的函数时，请参考该函数在 SQL Server 联机丛书中的专题介绍，可以查询到属性列表。

另一种你可以获取的属性设置类型是与排序规则属性有关。你在本例中使用的函数是“COLLATIONPROPERTY”，如下面的示例所示：

```
SELECT
COLLATIONPROPERTY('SQL_Latin1_General_CP1_CI_AS', 'Version')
AS CollVersion,
COLLATIONPROPERTY('SQL_Latin1_General_CP1_CI_AS', 'CodePage')
AS CollPage;
```

如你所见，“COLLATIONPROPERTY ”函数有两个参数：排序规则名称和你想查看的指定属性。在本例中，排序规则名称是“SQL_Latin1_General_CP1_CI_AS ”，属性是“Version”和“CodePage”。这些属性的设置都显示在下面的结果中：

CollVersion	CollPage
0	1252

注意，“Version”值是“0”，“CodePage”值是“1252”。如果我传入的是不同的排序规则，我的结果就是针对该排序规则的结果。

（作者：Robert Sheldon 译者：冯昀晖 来源：TT 中国）

原文标题：T-SQL 系统函数：查看服务器属性

链接：http://www.searchdatabase.com.cn/showcontent_39619.htm

T-SQL 系统函数：查看用户信息

本系列文章关注的是 T-SQL 系统函数，详细介绍了如何查看有关服务器属性和用户的详细信息。

请查看该系列文章第一篇：T-SQL 系统函数：查看数据修改信息

查看用户信息

SQL Server 还支持专门针对数据库用户和用户身份号码的几个函数。下面的例子使用了几个这种函数来查询用户信息：

```
SELECT
USER_NAME() AS UserName,
USER_ID() AS UserId,
SUSER_SID() AS UserSid,
SYSTEM_USER AS SystemUser;
```

这些函数非常简单，“USER_NAME”函数返回当前数据库用户的名称，“USER_ID”函数返回当前用户的数据库 ID。此外，“SUSER_SID”函数返回当前用户的安全数据库 ID，而“SYSTEM_USER”函数返回当前登录情况。下面的结果展示了该查询语句返回的数据：

UserName	UserId	UserSid	SystemUser
dbo	1	0x0105000000000005150000000FA4F0C2FDBEB0C5007E53B2BF4010000	WINSRV\Administrator

结果本身是有自解释性的，但是，要注意你可以给本例中的这三个函数传递参数。例如，你可以传递数据库用户 ID 给“USER_NAME”函数来获取相应的用户名。你还可以传递用户名参数给“USER_ID”函数来获取该用户的 ID。同样，你可以传递用户登录情况给“SUSER_SID”函数来获取相应的用户名。

下面的例子展示了在传递参数的情况下，如何使用这些函数：

```
SELECT
USER_NAME(1) AS UserName,
USER_ID('dbo') AS UserId,
SUSER_SID('WINSRV\Administrator') AS UserSid;
```

该语句返回如下结果：

UserName	UserId	UserSid
dbo	1	0x0105000000000005150000000FA4F0C2FDBEB0C5007E53B2BF4010000

你可以看到，这些函数在不同的情况下使用都是功能完善的，取决于你的需要。

(作者：Robert Sheldon 译者：冯昀晖 来源：TT 中国)

原文标题：T-SQL 系统函数：查看用户信息

链接：http://www.searchdatabase.com.cn/showcontent_39620.htm

T-SQL 系统函数：查看会话信息

本系列文章的第一部分讲述了如何利用系统函数提取有关数据修改，服务器属性和用户的信息。本文介绍了其它系统函数的用法，包括如何查看会话信息，如何利用 SQL Server 语言工具增强你的 T-SQL 语句功能。

查看会话信息

除了利用系统函数查看用户信息，你还可以查看有关当前会话的信息(可以包括会话用户)。在下面的例子中，我使用了几个专门针对会话的函数来提取数据：

```
SELECT
SESSION_USER AS SessionUser,
APP_NAME() AS AppName,
SESSIONPROPERTY('ANSI_NULLS') AS AnsiNulls,
SESSIONPROPERTY('QUOTED_IDENTIFIER') AS QuotedIDs;
```

“SESSION_USER”函数返回当前会话范围内当前数据库的当前用户，而“APP_NAME”函数返回启动当前会话的应用程序名称。“SESSIONPROPERTY”函数返回当前会话的具体“SET”选项值。

在本例中，“SELECT”语句包括两个“SESSIONPROPERTY ”函数。第一个提取“ANSI_NULLS”设置的值，第二个获取“QUOTED_IDENTIFIER”设置的值。下面的结果展示了该“SELECT”语句返回的信息：

SessionUser	AppName	AnsiNulls	QuotedIDs
dbo	Microsoft SQL Server Management Studio - Query	1	1

要注意，当前用户是 dbo，而启动程序是 SQL Server Management Studio。对于“SET”选项，值“1”代表打开设置。如果被关闭了，“SESSIONPROPERTY ”函数就返回值“0”。

(作者：Robert Sheldon 译者：冯昀晖 来源：TT 中国)

原文标题：T-SQL 系统函数：查看会话信息

链接：http://www.searchdatabase.com.cn/showcontent_39807.htm

T-SQL 系统函数：查看一般信息

本系列文章的第一部分讲述了如何利用系统函数提取有关数据修改，服务器属性和用户的信息。本文介绍了其它系统函数的用法，包括如何查看会话信息，如何利用 SQL Server 语言工具增强你的 T-SQL 语句功能。

查看一般信息

SQL Server 提供了许多系统函数，使你可以获取有关系统的一般信息。下面的“SELECT”语句展示了这样的几个函数：

```
SELECT
CURRENT_TIMESTAMP AS CurrentTimestamp,
GETANSINULL() AS DefaulNullability,
HOST_NAME() AS HostName,
HOST_ID() AS HostId;
```

正如你所预料的，“CURRENT_TIMESTAMP ”函数返回从操作系统获取的时间戳，而“GETANSINULL”函数返回当前会话当前数据库的默认为空性，“HOST_NAME ”函数返回工作站的名称，“HOST_ID”函数返回连接到 SQL Server 上的客户端计算机的应用程序进程 ID(PID)。

该“SELECT”语句返回下面的结果：

CurrentTimestamp	DefaultNullability	HostName	HostId
2010-08-04 17:33:45.590	1	WINSRV	3896

一些系统函数可以返回多个结果。例如，“fn_helpcollations”函数返回 SQL Server 实例支持的排序列表清单。请看下面的示例：

```
SELECT name AS CollName
FROM fn_helpcollations()
WHERE name LIKE 'greek_ci_ai%';
```

要注意，我把该函数当作了一个表来使用。我返回名称列，并限制返回某些“Greek”排序规则的数据，如下面的结果所示：

CollName
Greek_CI_AI
Greek_CI_AI_WS
Greek_CI_AI_KS
Greek_CI_AI_KS_WS

如你所见，只有四个排序规则符合“WHERE”语句中指定的搜索条件。然而，请注意，利用“fn_helpcollations”函数提取这些信息时多么简单啊。

（作者：Robert Sheldon 译者：冯昀晖 来源：TT 中国）

原文标题：T-SQL 系统函数：查看一般信息

链接：http://www.searchdatabase.com.cn/showcontent_39815.htm

T-SQL 系统函数：利用语言工具

本系列文章的第一部分讲述了如何利用系统函数提取有关数据修改，服务器属性和用户的信息。本文介绍了其它系统函数的用法，包括如何查看会话信息，如何利用 SQL Server 语言工具增强你的 T-SQL 语句功能。

利用语言工具

总结起来，我们来看看你能在 T-SQL 语句中用作工具的函数。这些函数类型通常帮助获取指定类型的数据。例如，下面的“SELECT”语句利用了 T-SQL 的“NEWID ”函数来创建“uniqueidentifier”类型的唯一数值。

```
SELECT NEWID() AS NewGuid;
```

从下面的结果中你可以看到，该语句返回一个新的值，显示为“uniqueidentifier”类型：

你可以使用的另一个函数是“PARSENAME”，它支持你解析完全合格的组件对象名称。该函数有两个参数：完全合格的对象名称和你想识别的组件名称。例如，下面的“SELECT”语句中我使用“PARSENAME ”函数来解析独立组件“Address”表名。

```
SELECT
PARSENAME('WinSrv.AdventureWorks2008.Person.Address', 4)
AS ServerName,
PARSENAME('WinSrv.AdventureWorks2008.Person.Address', 3)
AS DbName,
PARSENAME('WinSrv.AdventureWorks2008.Person.Address', 2)
AS SchemaName,
PARSENAME('WinSrv.AdventureWorks2008.Person.Address', 1)
AS ObjectName;
```

如下面的结果所示，该函数解析了该表完全合格名称的每个组件：

在下个例子中，我利用“ISDATE ”函数和“ISNUMERIC ”函数来识别指定值的数据类型：

```
SELECT
```

```
ISDATE('2010-08-04 17:33:45.590') AS IsDatetime,
ISNUMERIC('12ec3') AS IsNumber;
```

“ISDATE ”函数识别作为入参传递的值是否是合法的日期时间值，而“ISNUMERIC ”函数识别入参传递的值是否是合法的数值。如果该值的类型争取，该函数就会返回值“1”，而如果该值的类型不正确，该函数返回的值为“0”。对于上面的例子，该语句返回结果如下：

如你所见，第一个值是合法的日期时间值，而第二个值不是合法的数值。

另一个有用的函数是“ISNULL ”函数，如果源值是 null 的话，该函数返回指定的值。例如，下面的“SELECT”语句我使用了“ISNULL ”函数，如果“SalesQuota ”的值是 null 的话就返回“0”：

```
SELECT
FirstName + ' ' + LastName AS FullName,
ISNULL(SalesQuota, 0) AS SalesQuota
FROM
Sales.vSalesPerson
WHERE
CountryRegionName = 'United States';、
```

在源数据中，有三行数据的“SalesQuota”值是 null，它们是：Stephen Jiang, Amy Alberts 和 Syed Abbas。对于这三行，“ISNULL”函数对“SalesQuota ”值返回“0”，如下表所示：

正如上面的例子和我前面的文章展示的那样，在操作 SQL Server 数据时系统函数非常有用。然而，我没有覆盖到每种分类中的所有可用函数。系统函数还包括错误处理函数，在你的 T-SQL 代码包含“TRY…CATCH”块时可以使用。

(作者：Robert Sheldon 译者：冯昀晖 来源：TT 中国)

原文标题：T-SQL 系统函数：利用语言工具

链接：http://www.searchdatabase.com.cn/showcontent_39819.htm

探索 SQL Server 2008 中的 T-SQL 元数据函数（上）

SQL Server 2008 提供了称为元数据函数的一组函数。这些函数使你可以提取关于数据库和数据库对象的信息。例如，你可以提取赋予数据库，对象，表或者数据类型的 ID。你还可以查看各种类型对象的属性设置，比如索引是否为簇索引，或者列是否允许为空值。

这里我将给你介绍几个元数据函数，并给出一些示例来演示它们的用法。在我创建这些例子时，我都是在“AdventureWorks2008”示例数据库的上下文环境中运行的。一些示例还引用了“AdventureWorks2008”数据库，但是你可以很容易地用其他数据库替代。

注意：尽管我讲到了许多元数据函数，我也没有把所有元数据函数都讲完。请一定参考 SQL Server 2008 联机丛书中的“元数据函数(Transact-SQL)”来获取完整列表。从那里你可以链接到每个函数的主题介绍中，了解有关函数如何使用的详细说明。

查看对象名称和 ID

我们要看的前两个元数据函数是“DB_ID”和“DB_NAME”。你可以猜到，“DB_ID”返回指定数据库或者当前数据库的 ID，而“DB_NAME”返回指定数据库或者当前数据库的名称。下面的 SELECT 语句演示了如何使用这两个函数：

```
SELECT
    DB_ID() AS Id_DefaultDB,
    DB_ID('AdventureWorksDW2008') AS Id_SpecificDb,
    DB_NAME() AS Name_DefaultDb,
    DB_NAME(10) AS Name_SpecificDb;
```

“DB_ID”函数可以接收一个参数：一个字符串值，代表数据库的名称。我在“DB_ID”的第一个例子中没有包含该参数，因此该函数返回当前数据库的 ID，本例中就是“AdventureWorks2008”。在“DB_ID”函数的第二个例子中，我指定了“AdventureWorksDW2008 ”数据库作为参数。不过，要注意该名称必须用单引号括起来。

“DB_NAME”函数的用法也类似，只有一点差异就是它的参数是一个整形(Int)值，而不是字符串。在“DB_NAME”的第一个例子中，仍然是针对当前数据库。在“DB_NAME”的第二个例子中，我指定的参数值是 10，它是我系统中“AdventureWorksDW2008 ”数据库的 ID。

该 SELECT 语句返回结果如下表所示：

Id_DefaultDb	Id_SpecificDb	Name_DefaultDb	Name_SpecificDb
8	10	AdventureWorks2008	AdventureWorksDW2008

“DB_ID”的第一个例子返回的值是 8，这是我系统中“AdventureWorks2008 ”数据库的 ID，而该函数的二个例子返回的是 10。正如你所预料的，“DB_NAME”函数的第一个例子返回“AdventureWorks2008 ”数据库，第二个例子返回“AdventureWorksDW2008”。

接下来我们要介绍的两个元数据函数是“SCHEMA_ID”和“SCHEMA_NAME”，它们与“DB_ID”和“DB_NAME”也类似。“SCHEMA_ID”函数返回与对象名称有关的对象 ID，接收一个非必填的字符串参数(对象名称)。“SCHEMA_NAME”函数返回指定对象 ID 的对象名称，接收一个非必填的整形参数(对象 ID)。下面的 SELECT 语句展示了这两个函数的用法：

```
SELECT
  SCHEMA_ID() AS Id_DefaultSchema,
SCHEMA_ID('Sales') AS Id_SpecificSchema,
  SCHEMA_NAME() AS Name_DefaultSchema,
  SCHEMA_NAME(9) AS Name_SpecificSchema;
```

如果你没有指定对象名称或者 ID，函数就会返回该对象相关联的调用者的信息。例如，与我关联的该对象是“dbo”，对象 ID 是 1。那意味着“SCHEMA_ID”的第一个例子返回值为 1，而“SCHEMA_NAME”的第一个例子返回值是“dbo”，如下面的结果所示：

Id_DefaultSchema	Id_SpecificSchema	Name_DefaultSchema	Name_SpecificSchema
1	9	dbo	Sales

要注意我使用“Sales”作为“SCHEMA_ID”第二个例子的参数。如该结果所示，与该对象关联的 ID 是 9，这一点可以由“SCHEMA_NAME”函数第二个例子返回的结果得到验证。

(作者：Robert Sheldon 译者：冯昀晖 来源：TT 中国)

原文标题：探索 SQL Server 2008 中的 T-SQL 元数据函数（上）

链接：http://www.searchdatabase.com.cn/showcontent_41083.htm

探索 SQL Server 2008 中的 T-SQL 元数据函数（下）

另一组类似的元数据函数是“OBJECT_ID”和“OBJECT_NAME”。第一个函数返回模式范围内的任何对象，比如表或者视图。第二个函数返回模式范围内指定 ID 的任何对象名称。

“OBJECT_ID”函数接收一个参数：对象名称。如果你没有指定完整修饰的名称，那么该对象被认为是属于当前活动的数据库。如果你想提取另一个数据库中对象的 ID，你必须增加数据库名称以使该名称完整。“OBJECT_NAME”函数可以接收一个或者两个参数。第一个参数就是你想提取的对象名称对应的 ID。第二个可选参数是包含该对象的数据库 ID。如果你想获取非当前数据库对象名称，你应该指定数据库 ID。

下面的 SELECT 语句展示了如何使用“OBJECT_ID”函数和“OBJECT_NAME”函数：

```
SELECT
OBJECT_ID('Sales.SalesPerson')
AS Id_DefaultDb,
OBJECT_ID('AdventureWorksDW2008.dbo.FactInternetSales')
AS Id_SpecificDb,
OBJECT_NAME(1298103665)
AS Name_DefaultDb,
OBJECT_NAME(309576141, 10)
AS Name_SpecificDb;
```

在“ OBJECT_ID”的第一个例子中，我指定了当前活动数据库（“AdventureWorks2008”数据库）中模式范围内对象的名称（“Sales.SalesPerson”表）。在该函数的第二个例子中我指定了一个全修饰的名称：“AdventureWorksDW2008”数据库中的“dbo.FactInternetSales”表。

我在“OBJECT_NAME”函数的第一个例子中用到的参数是“AdventureWorks2008 ”数据库中“SalesPerson”表的 ID。在该函数的第二个例子中我首先指定了“SalesPerson ”表的 ID，然后是“AdventureWorksDW2008 ”数据库的 ID。该 SELECT 语句返回结果如下：

Id_DefaultDb	Id_SpecificDb	Name_DefaultDb	Name_SpecificDb
1298103665	309576141	SalesPerson	FactInternetSales

你可以看到，这些函数返回指定对象的名称和 ID，但是你可以容易地把这些 ID 和名称与其他类型对象连接。要得到 SQL Server 2008 中提供的模式范围内的对象清单，请查看 SQL Server 联机丛书“sys.objects (Transact-SQL)”主题。

另一组可以用来查找对象名称和它们 ID 的函数是“TYPE_ID”和“ TYPE_NAME”。你可以用这些函数查找 ID 和指定数据类型的名称。“TYPE_ID”函数接收数据类型名称作为它的参数，而“ TYPE_NAME”函数接收类型 ID 作为参数。例如，下面的 SELECT 语句返回 nvarchar 数据类型 ID，并返回 ID 是 231 的数据类型名称。

```
SELECT
TYPE_ID('nvarchar') AS TypeId,
TYPE_NAME(231) AS TypeName;
```

该 SELECT 语句返回如下结果：

TypeId	TypeName
231	nvarchar

正如你所预料的，结果确认了 nvarchar 数据类型的 ID 是 231。

(作者：Robert Sheldon 译者：冯昀晖 来源：TT 中国)

原文标题：探索 SQL Server 2008 中的 T-SQL 元数据函数（下）

链接：http://www.searchdatabase.com.cn/showcontent_41084.htm

利用 T-SQL 元数据函数检查属性设置（上）

本系列文章的第一部分介绍了元数据的基本情况，以及如何利用它们查看对象名称和 ID。本文讲解了其它元数据函数的用法，比如如何查看各种对象的属性设置。然后，第三部分会探索对数据库文件和文件组信息的使用。

查看对象属性

下一组元数据函数支持你获取各种数据库对象的属性设置。在大部分这些函数中，你必须提供至少两个参数。一个参数指定具体对象，另一个指定你想获取的属性设置。

注意：要查看函数支持的属性列表，请查看 SQL Server 联机丛书关于该函数的主题。该主题提供了对该属性返回值的每个属性和明细的描述。

我们要看的第一个函数是“DATABASEPROPERTYEX”。下面的 SELECT 语句列出了该函数的三个例子，每个例子中都指定了不同的属性：

```
SELECT
DATABASEPROPERTYEX('AdventureWorks2008',
    'Collation') AS Collation,
DATABASEPROPERTYEX('AdventureWorks2008',
    'IsAutoShrink') AS AutoShrink,
DATABASEPROPERTYEX('AdventureWorks2008',
'IsFullTextEnabled') AS FullTextEnabled;
```

你可以看到，我首先指定了数据库(AdventureWorks2008)的名称，然后指定了属性名称(分别是：Collation, IsAutoShrink 和 IsFullTextEnabled)。该 SELECT 语句返回结果如下：

Collation	AutoShrink	FullTextEnabled
SQL_Latin1_General_CP1_CI_AS	0	1

“Collation”属性返回指定名称数据库的默认排序规则，而“IsAutoShrink” 属性指定数据库文件是否设置为自动收缩模式。对于名称以“Is”打头的属性，返回值“1”表示“True”，返回值“0”表示“False”。如下面的结果所示，“AdventureWorks2008 ”数据库没有设置为自动收缩模式，而“IsFullTextEnabled ”表示指定数据库启用了全文搜索功能。

“OBJECTPROPERTYEX ”函数返回启用范围对象的属性设置。该函数有两个参数：对象 ID 和属性名称。例如，下面的 SELECT 语句返回“SalesPerson ”表的属性设置，它的对象 ID 是“1298103665”：

```
SELECT
OBJECTPROPERTYEX(1298103665, 'BaseType')
AS BaseType,
OBJECTPROPERTYEX(1298103665, 'IsIndexed')
AS Indexed,
OBJECTPROPERTYEX(1298103665, 'IsUserTable')
AS UserTable;
```

我指定了三个属性：“BaseType”表示对象的基本类型，“IsIndexed ”表示索引是否定义在该对象上，“IsUserTable ”表示对象是否是用户定义的表。该 SELECT 语句返回如下结果：

BaseType	Indexed	UserTable
U	1	1

因为“BaseType”属性返回值为“U”，我们知道该对象是一个用户定义的表。值“U”是该对象的基本 ID。你可以在 SQL Server 联机丛书“sys.objects (Transact-SQL)”主题中找到一个基本 ID 列表。

“IsIndexed ”属性返回为“True”(1)，因此我们知道该表是有索引的，“IsUserTable ”属性也返回“True”，因此我们知道该对象是一个用户定义的表。

(作者：Robert Sheldon 译者：冯昀晖 来源：TT 中国)

原文标题：利用 T-SQL 元数据函数检查属性设置（上）

链接：http://www.searchdatabase.com.cn/showcontent_41271.htm

利用 T-SQL 元数据函数检查属性设置（下）

如果你想查看有关索引的属性，你可以利用“INDEXPROPERTY ”函数。该函数有三个参数：索引定义于其上的对象 ID，索引名称和属性名称。例如，下面的 SELECT 语句获取关于“PK_SalesPerson_BusinessEntityID”索引的信息，该索引定义在“SalesPerson ”表上：

```
SELECT
    INDEXPROPERTY(1298103665,
'PK_SalesPerson_BusinessEntityID',
 'IsClustered') AS ClusteredInd,
    INDEXPROPERTY(1298103665,
'PK_SalesPerson_BusinessEntityID',
 'IsFullTextKey') AS FullTextKey,
    INDEXPROPERTY(1298103665,
'PK_SalesPerson_BusinessEntityID',
    'IsUnique') AS UniqueInd;
```

对于“INDEXPROPERTY ”函数的每个例子，我为“SalesPerson ”表提供了对象 ID，还有该索引的名称。最后，我指定了属性。“IsClustered ”属性表示该索引是否是簇索引，“IsFullTextKey ”属性表示该索引是否是该表的全文搜索键，“IsUnique ”属性告诉你该索引是否是唯一索引。

如下表所示，“PK_SalesPerson_BusinessEntityID”索引是簇索引，唯一性索引，但是不是全文搜索键：

ClusteredInd	FullTextKey	UniqueInd
1	0	1

另一个有用的函数是“COLUMNPROPERTY”，返回有关指定列的属性设置。与“INDEXPROPERTY ”函数一样，“COLUMNPROPERTY ”函数也有三个参数：列定义于其上的对象 ID，列名称和属性名称，如下面的示例所示：

```
SELECT
COLUMNPROPERTY(1298103665, 'SalesQuota',
    'AllowsNull') AS AllowsNull,
COLUMNPROPERTY(1298103665, 'SalesQuota',
    'ColumnId') AS ColumnId,
```

```
COLUMNPROPERTY(1298103665, 'SalesQuota',
'IsComputed') AS Computed;
```

在本例中，我获取了“SalesPerson ”表中“SalesQuota ”列的属性信息。“AllowsNull ”属性表示该列是否允许为空值，“ColumnID ”属性返回该列的赋值 ID，“IsComputed ”属性展示该列是否是一个计算列。如下表所示，“SalesQuota ”列云系空值，该列的 ID 是 3，它不是计算列：

AllowsNull	ColumnId	Computed
1	3	0

我们要学习的另一个与属性有关的函数是“TYPEPROPERTY”，它返回有关指定数据类型的信息。该函数有两个参数：数据类型名称和属性名称。下面的例子返回有关“money”数据类型的信息：

```
SELECT
TYPEPROPERTY('money', 'AllowsNull')
AS AllowsNull,
TYPEPROPERTY('money', 'Precision')
AS Precision,
TYPEPROPERTY('money', 'Scale')
AS Scale;
```

在我指定了第一个参数(数据类型名称)之后，我又指定了属性。“AllowsNull ”属性表示该数据类型是否允许空值，“Precision ”属性表示数值的最大值或者该值允许的最大字符数，“Scale ”属性告诉你运行的小数点位数。在上面的 SELECT 语句中，三个属性返回结果如下：

AllowsNull	Precision	Scale
1	19	4

你可以看到，“money”数据类型允许空值，支持 19 个字符，4 位小数。

(作者：Robert Sheldon 译者：冯昀晖 来源：TT 中国)

原文标题：利用 T-SQL 元数据函数检查属性设置（下）

链接：http://www.searchdatabase.com.cn/showcontent_41273.htm

利用 T-SQL 元数据函数获取数据库文件信息

本系列文章的第一部分讲解了如何使用基本的元数据函数查看对象名称和 ID，第二部分介绍了另外的元数据函数的用法，比如如何查看各种对象的属性设置。在本文中，你将学会如何利用元数据函数查看数据库文件和文件组的详细信息。

查看文件信息

除了获取对象和属性的信息，你还可以利用元数据函数查看有关数据库文件和它们文件组的有关详细信息。例如，在下面的例子中，我利用“FILE_IDEX”，“FILE_NAME”，“FILEGROUP_ID”和“ FILEGROUP_NAME”函数来获取支持“AdventureWorks2008 ”数据库的有关文件和文件组的数据：

```
SELECT
FILE_IDEX('AdventureWorks2008_Data') AS FileId,
FILE_NAME(1) AS FileName,
FILEGROUP_ID('primary') AS FileGroupId,
FILEGROUP_NAME(1) AS FileGroupName;
```

“FILE_IDEX”函数返回与数据库文件相关联的 ID。你可以看到，调用该函数时，你必须传递一个文件名作为参数。“ FILE_NAME”函数根据你参数中输入的 ID 返回对应的文件名。“ FILEGROUP_ID”函数返回文件组相关联的 ID。同样，你必须提供文件组的名称。最后，“FILEGROUP_NAME”函数根据你提供的 ID 返回文件组的名称。下面的 SELECT 语句返回了如下结果：

FileId	FileName	FileGroupId	FileGroupName
1	AdventureWorks2008_Data	1	PRIMARY

结果确认，“AdventureWorks2008_Data”文件的 ID 是 1，主文件组的 ID 也是 1。

SQL Server 还支持用元数据函数查看文件和文件组的属性。例如，你可以利用“FILEPROPERTY ”函数获取指定文件的属性设置。要使用该函数，你必须指定文件名以及属性名称，请看下面的 SQL 语句：

```
SELECT
FILEPROPERTY('AdventureWorks2008_Data',
'IsReadOnly') AS ReadOnly,
```

```
        FILEPROPERTY('AdventureWorks2008_Data',
            'IsPrimaryFile') AS PrimaryFile,
        FILEPROPERTY('AdventureWorks2008_Data',
            'SpaceUsed') AS SpaceUsed;
```

你可以看到，“FILEPROPERTY ”函数与你之前看到的属性函数类似。首先，我指定了文件名(AdventureWorks2008_Data)，然后又指定了属性名。“IsReadOnly ”属性表示文件是否是只读的，“IsPrimaryFile ”属性表示该文件是否是主数据库文件，“SpaceUsed ”属性表示已经给该文件分配了多少页。如下表所示，该文件不是只读的，但是它是主文件，它已经被分配了 23240 页：

ReadOnly	PrimaryFile	SpaceUsed
0	1	23240

SQL Server 还支持“FILEGROUPPROPERTY ”函数，你可以用它查看文件组明细信息。同样，你必须指定两个参数：文件组名称和属性名称。在下面的 SELECT 语句中，我获取了主文件组的信息：

```
SELECT
    FILEGROUPPROPERTY('primary', 'IsReadOnly')
        AS ReadOnly,
    FILEGROUPPROPERTY('primary', 'IsUserDefinedFg')
        AS UserDefined,
    FILEGROUPPROPERTY('primary', 'IsDefault')
        AS DefaultFg;
```

正如你所预料的，“IsReadOnly ”属性表示该文件组是否是只读，“IsUserDefinedFg ”属性表示是否是用户定义的文件组，“IsDefault ”属性表示该文件组是否是默认的文件组。下面的结果展示了该文件组既不是只读的，也不是用户定义的，但是它是默认的文件组：

ReadOnly	UserDefined	DefaultFg
0	0	1

到目前为止，你应该已经掌握元数据函数了。在这一组简短的系列文章中，我给你介绍了各种元数据函数，但是，出于简洁性考虑，我没有讲解到每个函数的细节，以及在 T-SQL 语句中，你可以怎样进一步地利用这些函数。也就是说，我没有深入介绍这些函数的限制和约束，但是你可以通过阅读 SQL Server 联机丛书中相关函数的主题来了解这些细节信息。经过了这一系列的学习，你现在应该对这些函数足够熟悉了，可以访问每个数据库中可用的大部分元数据了。

(作者：Robert Sheldon 译者：冯昀晖 来源：TT 中国)

原文标题：利用 T-SQL 元数据函数获取数据库文件信息

链接：http://www.searchdatabase.com.cn/showcontent_41323.htm

SQL Server 配置函数实战教程（上）

Transact-SQL 提供了一组可以获取当前 SQL Server 实例配置选项设置信息的函数。这些函数被称为“配置函数”，这些语言元素支持你提取许多信息，比如：当前使用的语言名称和 ID，允许并发的用户最大连接数，或者是你所连接 SQL Server 实例的版本。

通过 SQL Server 配置函数提取数据是非常简单的。在大部分情况下，你可以创建一个简单的 SELECT 语句，在其中调用你想使用的函数。在本文中，我介绍了许多这种好用的配置函数，并给你举了一些例子来说明如何使用它们。我是在本地的 SQL Server 2008 实例上创建的示例，但是它们也可以在 SQL Server 2005 上运行。然而，因为 SQL Server 配置随着实例的不同而不同，所以我这里展示的结果可能与你在你系统上获取到的信息上有所不同，但是它们至少能给你对预期的结果有一个认识。

我们要了解的第一个配置函数是“@@DATEFIRST”。在 SQL Server 中，自动赋予会话的配置设置之一就是日历周的第一天。默认情况下是星期日(当然你可以通过“SET”语句来修改默认值)。你可以使用“@@DATEFIRST”函数来获取设置的第一天信息，如下面的语句所示：

```
SELECT @@DATEFIRST AS FirstDay;
```

因为我没有修改我使用的 SQL Sever 实例中的第一天设置，所以该语句返回的结果就是星期日。然而，要注意该函数只返回与星期日等价的数值，所以这里返回的就是“7”。这样的话，我修改了 SELECT 语句，在其中增加了“CASE”表达式，这样就可以返回该天的名称，而不是与它等价的数值。请看下面的例子：

```
SELECT
CASE @@DATEFIRST
WHEN 1 THEN 'Monday'
WHEN 2 THEN 'Tuesday'
WHEN 3 THEN 'Wednesday'
WHEN 4 THEN 'Thursday'
WHEN 5 THEN 'Friday'
WHEN 6 THEN 'Saturday'
WHEN 7 THEN 'Sunday'
END AS FirstDay;
```

不出你的所料，该语句现在返回名称“星期日”，因为“@@DATEFIRST”函数返回的值是“7”。

我们要了解的下一个函数是“@@DBTS”。该函数返回指定数据库中最后使用的时间戳值。例如，如果你在带有时间戳列的表中执行了插入或者更新数据，该列就被当前时间戳值更新了。然后，你可以利用“@@DBTS”函数来获取该时间戳值。请看我下面列出的例子：

```
USE AdventureWorks2008;
SELECT @@DBTS AS DbTimeStamp;
```

请注意，我首先指定了数据库的名称。那是因为时间戳值是与具体的每个数据库有关的。你可以指定任何数据库，如果你想提取该数据库最近的时间戳值。接下来我运行了包含有“@@DBTS”函数的 SELECT 语句。

“@@DBTS”函数返回的时间戳值是“varbinary”数据类型，在本例中，它的值是“0x0000000000020791”。要记住时间戳值与时间或者日期没有任何关系。它们只是展示数据被修改的顺序，这种做法最初是为了支持数据库恢复算法。

还有另一种 SQL Server 设置你可以用配置函数访问，那就是当前语言信息。要提取这一信息，需要使用两个函数，它们是：“@@LANGID”或者“@@LANGUAGE”。“@@LANGID”函数返回本地语言识别码，而“@@LANGUAGE”函数返回语言名称。例如，下面的 SELECT 语句返回语言 ID 和名称：

```
SELECT
@@LANGID AS CurrentLangID,
@@LANGUAGE AS CurrentLang;
```

执行结果如下所示，该 SELECT 语句返回语言 ID 值“0”和语言名称值“us_english”：

CurrentLangID	CurrentLang
0	us_english

这些值是默认 SQL Server 安装返回的值。

（作者：Robert Sheldon 译者：冯昀晖 来源：TT 中国）

原文标题：SQL Server 配置函数实战教程（上）

链接：http://www.searchdatabase.com.cn/showcontent_42101.htm

SQL Server 配置函数实战教程（下）

现在我们来看看如何获取 SQL Server 实例允许的用户连接最大数量。要做到这一点，需要使用“@@MAX_CONNECTIONS”函数，请看下面的例子：

```
SELECT @@MAX_CONNECTIONS AS MaxConnections;
```

在我的系统中，该 SELECT 语句返回值是“32，767”。请注意，连接数多少取决于你的 SQL Server 版本和硬件与应用程序的限制。

下一个函数是“@@MAX_PRECISION”，它返回“decimal”和“numeric”数据类型的精度级别。默认情况下，精度值是 38，下面这个 SELECT 语句会返回这个值：

```
SELECT @@MAX_PRECISION AS MaxPrecision;
```

现在，我们来看看“@@OPTIONS”函数。SQL Server 支持许多用户选项，这些选项可以使用“SET”命令配置。这些选项设置被保存为二进制值，因此可以通过配置这些二进制来修改。“@@OPTIONS”函数返回一个代表二进制值的整数。例如，如果我运行下面的这句 SELECT 语句，“@@OPTIONS”函数会返回值“5496”。

```
SELECT @@OPTIONS AS SetOptions;
```

假设我现在修改了一个用户选项，例如，在下面的语句中，我设置“NOCOUNT”选项为“ON”，然后我再执行“@@OPTIONS”函数的时候，就会返回新的选项值：

```
SET NOCOUNT ON;
SELECT @@OPTIONS AS SetOptions;
```

现在，该 SELECT 语句返回的值是“6008”，它代表存储用户设置的二进制值已经发生了改变。然而，如果我运行下面的“SET”语句来设置“NOCOUNT”选项为“OFF”，然后再调用“@@OPTIONS”函数，该函数还会返回值“5496”。

```
SET NOCOUNT OFF;
SELECT @@OPTIONS AS SetOptions;
```

你还可以使用配置函数来获取 SQL Server 实例有关的一般信息。在下面的 SELECT 语句中我查询了服务器名称，服务名称，Session ID 和文本大小：

```
SELECT
@@SERVERNAME AS ServerName,
@@SERVICENAME AS ServiceName,
@@SPID AS SessionID,
@@TEXTSIZE AS TxtSize;
```

我们来分别看看这些函数：

- **@@SERVERNAME**：返回你所连接的 SQL Server 实例名称。如果是默认实例的话，该名称就是 SQL Server 安装所在操作系统服务器。如果是命名的 SQL Server 实例的话，该函数返回服务器名称和实例名称。
- **@@SERVICENAME**：返回用于特定实例的 SQL Server 服务名称。如果是默认 SQL Server 实例，使用的就是“MSSQLSERVER”。如果安装的是命名实例的话，就返回它的名称。
- **@@SPID**：返回当前用户进程的 session ID(前身是“server process ID”)。
- **@@TEXTSIZE**：返回“TEXTSIZE”选项当前设置值的字节数。该选项决定了一个 SELECT 语句返回的“varchar(max)”，“nvarchar(max)”，“varbinary(max)”，“text”，“ntext”和“image”类型数据的大小。

你可以在下面的结果中看到，当前 SQL Server 实例的名称是“SRV023\SQLSRV2008”，服务名称是“SQLSRV2008”，session ID 是“54”，“TEXTSIZE”设置是“2，147，483，647”字节。

ServerName	ServiceName	SessionID	TxtSize
SRV023\SQLSRV2008	SQLSRV2008	54	2147483647

我们要介绍的另一个配置函数是“@@VERSION”，它返回当前 SQL Server 实例的版本信息，处理器架构，构建日期和操作系统信息。请看下面的 SQL：

```
SELECT @@VERSION AS InstanceVersion;
```

在我的系统中，该“SELECT”语句返回下面的结果：

```
Microsoft SQL Server 2008 (SP1) -
10.0.2531.0 (Intel X86) Mar 29 2009 10:27:29 Copyright (c) 1988-
2008 Microsoft Corporation Developer Edition on Windows NT 5.1 (Build 2600: Se
rvice Pack 3)
```

这就是 SQL Server 配置函数的全部用法。如你所见，你可以简单地在 SELECT 语句中调用这些函数。你可能也注意到了，所有这些配置函数前面都有两个符号(@@)，这是为了便于识别。在本文中，我讲到了 SQL Server 中提供的大部分配置函数，但是还不是全部。

要获得完整列表，请在 SQL Server 联机丛书中查看“配置函数(Transact-SQL)”主题。其中每一个函数都可以链接到该函数的详细描述页。

(作者：Robert Sheldon 译者：冯昀晖 来源：TT 中国)

原文标题：SQL Server 配置函数实战教程（下）

链接：http://www.searchdatabase.com.cn/showcontent_42103.htm