SQL Server 2008 新特性

SQL Server 2008 新特性

SQL Server2008 在以前版本的基础上增加一系列新的特征，如你可以了解 SQL Server 2008 中的 T-SQL 基本对象和操作，如常量、标识符和分隔符、T-SQL 函数类型如聚合函数和纯量函数等、SQL Server 中的数据类型如数值型数据类型、字符型数据类型、临时数据类型、混合型数据类型以及新的安全特性等等。

SQL Server 2008 中的 T-SQL 基本对象

本文节选自《微软 SQL Server 2008：初学者指南》学习 SQL Server2008 T-SQL 支持的基本元素和基本操作。你将了解 T-SQL 的基本对象和操作，包括常量、标识符和分隔符。你还将了解到每个对象与操作的相应的数据类型。作者 Dusan Petkovic 介绍 SQL Server2008 中的 NULL 值、标量对象和全程标量。

- SQL Server 2008 中的T-SQL基本对象

SQL Server 2008 中的 T-SQL 函数类型

T-SQL 函数可以是聚合函数或者纯量函数。以下部分是对这些函数类型的描述：1、聚合函数；聚合函数又分为便捷型聚合函数、统计型聚合函数、自定义聚合函数、分析型聚合函数。2、纯量函数；纯量函数又分为数字函数、日期函数、字符串函数、系统函数、元数据函数……

- SQL Server 2008 中的T-SQL函数类型（一）
- SQL Server 2008 中的 T-SQL 函数类型（二）
- SQL Server 2008 中的 T-SQL 函数类型（三）

SQL Server 2008 中的数据类型

所有的列中的数值都必须属于同样的数据类型（除非指定了 SQL_VARIANT 数据类型值）。T-SQL 使用不同的数据类型，具体分类如下：数值型数据类型、字符型数据类型、临时数据类型、混合型数据类型、附带有 VARDECIMAL 的 DECIMAL 存储类型等等。

- 在SQL Server 2008 中使用T-SQL数据类型（一）
- 在SQL Server 2008 中使用T-SQL数据类型（二）
- 在SQL Server 2008 中使用T-SQL数据类型（三）
- SQL Server 2008 中的新日期数据类型具有灵活性（一）
- SQL Server 2008 中的新日期数据类型具有灵活性（二）

SQL Server 2008 设计

在大型商业智能系统或数据仓储上工作时，我们常常面临的问题就是数据及时性。本节向你介绍了 SQL Server 2008 中的 BI 性能：change data capture（CDC），CDC 能基于来源处理系统中的数据改变通知帮你在数据仓储或专用数据库实时更新数据。这在数据仓库和商业智能情况中是一个详细定义问题，但是是非常普遍的一个问题。

- 在 SQL Server 中用 CDC 提高 BI 报告准确性

SQL Server 2008 安全

SQL Server 2008 引进了一种新的审计性能，它能使 DBA 追踪数据库的使用并进行详细审计。我们能在服务器和数据库上安装审计，在单独的数据库对象上激活并用各种不同的格式保存，如二进制文件或 Windows Application 日志。在 SQL Server 2008 里安装审计，步骤如下：给每个 SQL Server 2008 具体实例创建一个 SQL Server 审计；创建服务

器审计规范、数据库审计规范或者其中的一个；激活 SQL Server 审计以及查看审计数据。

- 在 SQL Server 2008 中安装安全审计（一）
- 在 SQL Server 2008 中安装安全审计（二）
- 你需要做的、有关 SQL Server 2008 新的安全特性工作
- SQL Server 2008 Reporting Services 的新特性

SQL Server 2008 中的 T-SQL 基本对象

本文节选自《微软 SQL Server 2008：初学者指南》学习 SQL Server2008 T-SQL 支持的基本元素和基本操作。你将了解 T-SQL 的基本对象和操作，包括常量、标识符和分隔符。你还将了解到每个对象与操作的相应的数据类型。作者 Dusan Petkovic 介绍 SQL Server2008 中的 NULL 值、标量对象和全程标量。

SQL 的基本对象：

数据库语言引擎 T-SQL 和其他程序设计语言有着相同的特征：

- 字面值（也叫常量）
- 标识符
- 分隔符
- 保留关键字

以下部分主要介绍这些内容。

字面值

字面值是一个包括文字与数字，十六进制或数字常量。一个字符常量包含单引号（’’）或双引号（″ ″）字符集中的一个或多个字符（如果多次使用了双引号就更趋向于使用单引号）。

如果你想在单引号分割的字符串中用到单独的引号，你就可以在这一字符串中用连续的单引号。十六进制的常量表示不可打印的字符或者是其他二进制数据。每个十六进制的常量都以 0x 开头，后面附带有字符或者数字。例 4.1 和 4.2 就说明了有效和无效字符串常量和十六进制常量。

例 4.1

有效字符常量和十六进制的常量如下：

```
   'Philadelphia'
"Berkeley,  CA 94710"
'9876'
'省略符号如下所示：can''t'（注意两个连续的单引号）
0x53514C0D
例 4.2
```

下面的不属于字符常量：

```
   'AB'C'（奇数个单引号）
'New York"（同样类型的引号——单或双——都必须在每个字符串末尾使用）
```

数字常量包括所有有记号或没记号的整数、定点值和符点值（如例 4.3）。

例 4.3

以下属于数字常量

```
130
-130.00
-0.357E5 (科学计数法—nEm 表示 n 乘 10^m)
22.3E-3
```

一个常量通常有一种数据类型和长度，这二者都取决于常量格式。此外，每个数据常量都有一个精确度和比例因子。（不同字面值的数据类型在本章中会提到。）

分隔符

在 T-SQL 中，双引号有两层意思。除了应用字符串之外，双引号还能够用来做分隔符，也就是所谓的定界表示符（delimited identifier）。分割标识符是标识的一种特殊类型，通常将保留关键字当作标识符并且用数据库对象的名称命名空间。

注：单引号和双引号之间的区别就在于前者是用于 SQL92 标准。标识符这种情况中，这种标准用于区分常规和分割标识符。关键的两点就是分割标识符是用双引号引出的而且还区分大小写（T-SQL 还支持用方括号代替双引号）。双引号只用于分割字符串。一般来说，分割标识符说明了标识符的规格，对保留的关键字也起到了同样的作用。分割标识符还可以让你不用名字（标识符、变量名），这些名字也可以在将来的 SQL 标准中用作保留的关键字。并且，分割标识符还可能在标识符名中包含不合规定的字符，如空格。

在 T-SQL 中，使用双引号是来定义 SET 语句的 QUOTED_IDENTIFIER 选项的。如果这一选项设为 ON（即默认值），那么双引号中的标识符就被定义成了分割标识符。在这种情况下，双引号就不能用于分割字符串。

注：说明一个 T-SQL 语句的注释有两种方法。使用一对字符/* and */ ，注释就是对附着在里面的内容进行说名。这种情况中，注释内容可能扩展成很多行。另外，这些字符--（两个连字符）表示当前行剩下的就是注释。（两个连字符就—符合 ANSI SQL 标准，而/*和*/是 T-SQL 的扩展名。

标识符

T-SQL 中，标识符用于识别数据库对象如数据库、表和索引。他们通过字符串表示出来，这些字符串的长度可以达到 128 个字符，还包含字母、数字或者下面的字符：_、@，、#和$。每个名称都必须以一个字母或者以下字符中的一个字符开头：_、@或#。#在以它开头的表明或存储程序名表示一个临时对象，而以@开头的时间则表示一个变量。就像是之前提到的，这些规则并不适用于分割标识符（也叫作引用标识符），分割标识符可以将这些字符包含在内或者其中的任意字符开头（而不是分割副自己）。

保留关键字

每种程序设计语言都有一系列有保留意义的名称，他们被写入并用在定义格式中。这些名称叫做保留关键字。T-SQL 使用了大量这种名称，很多其他程序设计语言都不能用它们做对象名，除非这些对象是被详细划分或者是引用标识符。

注：在 T-SQL 中，所有数据类型的名称和系统功能，如 CHARACTER 和 INTEGER 都不是保留的关键字。它们是用来表示对象。（不要使用数据类型和系统功能作对象名称！这样会让 T-SQL 语句很难理解）

（作者： SearchSQLServer.com 译者：April 来源：TT 中国）

SQL Server 2008 中的 T-SQL 函数类型（一）

T-SQL 函数

T-SQL 函数可以是聚合函数或者纯量函数。以下部分是对这些函数类型的描述：

聚合函数

聚合函数适用于列中的一组数据值。聚合函数返回单一值。T-SQL 支持几组聚合函数：

- 便捷型聚合函数
- 统计型聚合函数
- 自定义聚合函数
- 分析型聚合函数

统计型和分析型聚合函数在 24 章中已经讨论过了。自定义聚合函数不属于本章范围。以下是对便捷型聚合函数的描述：

AVG——计算列中数值得算数中项（平均值）。该列中包含的值必须是数字类型的。

MAX 和 MIN——分别计算列中的最大和最小数据值。该列中的值可以属于数字、字符串以及日期/时间型的。

SUM——计算列中的所有数据值之和。这些列中的值必须是数字。

COUNT——计算列中的（非零）数据值。COUNT(*)是唯一一种不适用于列的聚合函数。该函数是返回的是行数（不管列中的数字是否有 NULL 值）。

COUNT_BIG——和 COUNT 类似，唯一的区别就是 COUNT_BIG 返回的值属于 BIGINT 数据类型。

SELECT 语句便捷函数的使用在第六章中进行了详细介绍。

纯量函数

除了聚合函数之外，T-SQL 还提供了一些纯量函数，用于创建标量表达式。纯量函数主要针对单一值或列表值；和聚合函数相反，聚合函数主要是针对多行中的数据。纯量函数可以分成以下几类：

- 数字函数
- 日期函数
- 字符串函数
- 系统函数
- 元数据函数

以下是对这些函数类型的详细描述：

数字函数

T-SQL 中的数字函数就是修改数值的数字函数。可以使用以下列数字函数：

函数	说明
ABS(n)	返回数字表达式 n 的绝对值(也就是说负数值返回的结果就是正数)。例: SELECT ABS(-5.767) = 5.767， SELECT ABS(6.384) = 6.384

ACOS(n)	计算 n. n 的反余弦值，结果属于 FLOAT 数据类型。
ASIN(n)	计算 n. n 的正弦值，结果属于 FLOAT 的数据类型。
ATAN(n)	计算 n. n 的反正切值，结果属于 FLOAT 数据类型。
ATN2(n, m)	计算 n/m. n, m 的反正切值，结果属于 FLOAT 数据类型。
CEILING(n)	返回的整数值大于或等于具体参数。例: SELECT CEILING(4.88) = 5 SELECT CEILING(-4.88) = -4
COS(n)	计算 n. n 的余弦值，结果值属于 FLOAT 数据类型。
COT(n)	计算 n. n 的余切值，并且结果值属于 FLOAT 数据类型。
DEGREES(n)	将弧度转变成度数。例: SELECT DEGREES(PI()/2) = 90.0 SELECT DEGREES(0.75) = 42.97
EXP(n)	计算 e^n 的值。例: SELECT EXP(1) = 2.7183

FLOOR(n)	计算小于或等于给定值 n 的最大整数值。例: SELECT FLOOR(4.88) = 4
LOG(n)	计算 n 的自然对数（也就是说基数为 e）。例: SELECT LOG(4.67) = 1.54 SELECT LOG(0.12) = -2.12
LOG10(n)	计算 n 的对数（基数为 10）例:
PI()	返回圆周率值（3.14）。
POWER(x, y)	计算 x^y 值。例: SELECT POWER(3.12, 5) = 295.65 SELECT POWER(81, 0.5) = 9
RADIANS(n)	将度数转换成弧度。如: SELECT RADIANS(90.0) = 1.57 SELECT RADIANS(42.97) = 0.75
<? xml:namespace prefix = st1 ns = "urn:schemas-microsoft-com:office:smarttags" />RAND	返回 0-1 之间的任意值，结果属于 FLOAT 数据类型。

ROUND(n， p，[t])	对 n 进行四舍五入，精确度为 p。p 为正数时，就对小数点右边的数字进行四舍五入。如果是负数的话，就对小数点左边的数字进行四舍五入。可选参数 t 就删除了 n。例: SELECT ROUND(5.4567，3) = 5.4570 SELECT ROUND(345.4567，－1) = 350.0000 SELECT ROUND(345.4567，－1，1) = 340.0000
ROWCOUNT_BIG	返回系统执行的、受最后一行 T-SQL 语句影响的行数。该函数的返回值为 BIGINT 数据类型。
SIGN(n)	返回 n 值的符号数字 (+1 为正数，－1 f 为负数，0 就是 0).例:SELECT SIGN(0.88) = 1
SIN(n)	计算 n. n 的正弦，结果值属于 FLOAT 数据类型。
SQRT(n)	计算 n 的平方根。例: SELECT SQRT(9) = 3
SQUARE(n)	返回给定式的平方值。例: SELECT SQUARE(9) = 81

TAN(n)	计算 n. n 的正切，结果值属于 FLOAT 数据类型。

(作者： SearchSQLServer.com 译者： April 来源：TT 中国)

SQL Server 2008 中的 T-SQL 函数类型（二）

日期函数计算表达式的日期和时间或者是从时间间隔中返回该值。T-SQL 语句支持以下函数：

函数	说明
GETDATE()	返回目前系统日期和时间。例: SELECT GETDATE() = 2008-01-01 13:03:31.390
DATEPART(item, date)	返回日期指定的 item，date 为一个整数。例: SELECT DATEPART(month, '01.01.2005') = 1 (1 = January) SELECT DATEPART(weekday, '01.01.2005') = 7 (7 = Sunday)
DATENAME(item, date)	返回日期指定的 item，date 为一个字符串。例: SELECT DATENAME(weekday, '01.01.2005') = Saturday
DATEDIFF(item, dat1, dat2)	计算两个日期部分 dat1 和 dat2 之间的区别，返回的结果为 item 表示单元的整数值。 例:

	SELECT DATEDIFF(year, BirthDate, GETDATE()) AS age FROM employee; ->返回每个员工的年龄
DATEADD(i, n, d)	将 i 值单元里的数字 n 增加到指定日期 d。例: SELECT DATEADD(DAY, 3, HireDate) AS age FROM employee; ->在每个员工聘用日期的基础上增加 3 天（详见 sample 数据库）

字符串函数

字符串函数用于处理列中的数据值，通常属于字符型数据类型。T-SQL 支持以下字符串函数：

函数	说明
ASCII(character)	将具体的字符转换为相应的整数（ASCII）代码。返回结果为正数。例： SELECT ASCII('A') = 65
CHAR(integer)	将 ASCII 代码转换为相应的字符。例： SELECT CHAR(65) = 'A'。
CHARINDEX(z1, z2)	返回部分字符串 z1 在字符串 z2 中首次出现的起始位置。如果 z1 没有在 z2 中出现，那么返回值就为 0。例: SELECT CHARINDEX('bl', 'table') =

	3。
DIFFERENCE(z1, z2)	返回值为 0-4 之间的整数，这就是 z1 和 z2 这两个字符串 SOUNDEX 之间的区别。SOUNDEX 返回的数字指定的是字符串的语音。这种方法能够判断有相同发音的字符串）例: SELECT DIFFERENCE('spelling', 'telling') = 2 (发音有些相近， 0 =发音部相同)。
LEFT(z, length)	返回字符串 z 中的第一个字符长度。
LEN(z)	返回指定的字符串表达式的字符个数而不是字节个数，包括后面的空格。
LOWER(z1)	将字符串 z1 中所有的大写字母转换成小写字母。小写字母和数字以及其它的字母保持不变。例: SELECT LOWER('BiG') = 'big'。
LTRIM(z)	去掉字符串 z 开头的空格。例: SELECT LTRIM(' String') = 'String'。
NCHAR(i)	返回由统一码标准定义的、有指定整数代码的统一码字符。
QUOTENAME(char_string)	返回有分隔符的统一码字符串，使输入字符串变成有效分隔符。
PATINDEX(%p%, expr)	返回指定表达式 expr 中模式 p 第一次出现的起始位置，如果没有找到这个模式，那么返回值就为零。例:

	1) SELECT PATINDEX('%gs%', 'longstring') = 4; 2) SELECT RIGHT(ContactName, LEN(ContactName)-PATINDEX('% %', ContactName)) AS First_name FROM Customers; (第二个查询从 customers 列中返回所有名。)
REPLACE(str1, str2, str3)	将所有 str1 中出现的 str2 替换为 str3。例: SELECT REPLACE('shave', 's', 'be') = behave
REPLICATE(z, i)	将字符串 z 重复 i 次。例: SELECT REPLICATE('a', 10) = 'aaaaaaaaaa'
REVERSE(z)	将字符串 z 显示为倒序。例: SELECT REVERSE('calculate') = 'etaluclac'
RIGHT(z, length)	在字符串 z 中返回最后字符的长度。例: SELECT RIGHT('Notebook', 4) = 'book'
RTRIM(z)	取消字符串 z 最后的空格。例: SELECT RTRIM('Notebook ') = 'Notebook'

SOUNDEX(a)	返回四个字符的 SOUNDEX 代码判断两个字符串中的相似性。例: SELECT SOUNDEX('spelling') = S145
SPACE(length)	返回一个字符串，length 为其指定的空间长度。例: SELECT SPACE = ' '
STR(f, [len [, d]])	将指定的 float 表达式 f 转换成字符串。len 是指字符串的长度，包括小数点、正负号、数字和空格（默认值为 10），d 为小数点右边的被返回的数字。例: SELECT STR(3.45678, 4, 2) = '3.46'
STUFF(z1, a, length, z2)	用字符串 z2 中的位于 a 处的部分字符串代替 z1 中的部分字符串，代替 z1 中的 length 字符。例: SELECT STUFF('Notebook', 5, 0, ' in a ') = 'Note in a book' SELECT STUFF('Notebook', 1, 4, 'Hand') = 'Handbook'
SUBSTRING(z, a, length)	在字符串 z 中的 a 处创建部分字符串，length 为新创建字符串的长度。例: SELECT SUBSTRING('wardrobe', 1, 4) = 'ward'
UNICODE	返回由统一码定义的整数值，该值为输入表达式的第一个字符。
UPPER(z)	将字符串 z 中的所有小写字母转换成大

	写字母。大写字母和数字不变。例: SELECT UPPER('loWer') = 'LOWER'

(作者：SearchSQLServer.com 译者：April 来源：TT 中国)

SQL Server 2008 中的 T-SQL 函数类型（三）

系统函数

T-SQL 系统函数提供了更多有关数据库对象的信息。大部分系统函数用的是内部数字标识符（ID），系统将标识付值给每个数据库对象。使用这类标识符，系统就能独立识别每个数据库对象。系统函数提供了一些有关数据库系统的信息。以下为一些描述系统函数的表（要查看完整的系统函数清单，请参考联机丛书）。

函数	说明
CAST(a AS type [(length)]	将表达式 a 转换成指定的数据类型 type （如果可能的话）。A 可以是任一有效表达式。例: SELECT CAST(3000000000 AS BIGINT) = 3000000000
COALESCE(a1, a2, …)	返回给定清单上的表达式 a1、a2……，并且第一个表达式的返回值不是 NULL 值。
COL_LENGTH(obj, col)	返回 col 列的长度，该长度值属于数据库对象（表或视图）obj 。 例: SELECT COL_LENGTH('customers', 'cust_ID') = 10

CONVERT(type[(length)], a)	和 CAST 相等，但是对这两个参数指定条件不同。CONVERT 能用于任意数据类型。
CURRENT_TIMESTAMP	返回目前的日期和时间。例：SELECT CURRENT_TIMESTAMP = '2008-01-01 17:22:55.670'
CURRENT_USER	返回目前用户的姓名。
DATALENGTH(z)	计算表达式 z 的结果长度（字节）例： SELECT DATALENGTH(ProductName) FROM products. （该查询返回每个域名的长度）
GETANSINULL('dbname')	如果按照 ANSI SQL 标准在数据库 dbname 中使用 NULL 值，那么返回值为 1（参考本章末对 NULL 值的详细说明）。例： SELECT GETANSINULL('AdventureWorks') = 1
ISNULL(expr, value)	如果 expr 不为零，就返回 expr 值；否则就返回 value （查看例 5.22）。
ISNUMERIC(expression)	判断表达式是否属于无效的数字型。
NEWID()	创建由 16 个字节组成的二进制字符串存储 UNIQUEIDENTIFIER 数据

	类型。
NEWSEQUENTIALID()	在指定的计算机上创建 GUID，它比该函数之前产生的 GUID 值要大。我们只可以将这个函数设置为默认值。
NULLIF(expr1, expr2)	如果表达式 expr1 和 expr2 相等，返回 NULL 值。例: SELECT NULLIF(project_no, 'p1') FROM projects（该查询返回带有 project_no = 'p1'的项目值为 NULL）。
SERVERPROPERTY(propertyname)	返回数据库服务器的属性信息。
SYSTEM_USER	返回目前用户的登陆 ID。例: SELECT SYSTEM_USER = LTB13942dusan
USER_ID([user_name])	返回用户 user_name 的标识符。如果没有指定名字，就检索当前用户的标识符。例: SELECT USER_ID('guest') = 2
USER_NAME([id])	返回带有标识符 id 的用户名字。如果没有指定用户名，就检索当前用户的名字。例: SELECT USER_NAME = 'guest'

所有字符串函数可以以任意顺序嵌入，例如：REVERSE(CURRENT_USER)。

元数据函数

一般来说，元数据函数返回的是有关指定数据库和数据对象的信息。下表介绍了几种元数据函数（查看完整元数据函数清单，请参考联机丛书）。

函数	说明
COL_NAME(tab_id, col_id)	返回列名，这些列都属于带有 ID tab_id 以及列 ID col_id 的表。例: SELECT COL_NAME(OBJECT_ID('employee'), 3) = 'emp_lname'
COLUMNPROPERTY(id, col, property)	返回指定列的信息。例: SELECT COLUMNPROPERTY(object_id('project'), 'project_no', 'PRECISION') = 4
DATABASEPROPERTY(database, property)	返回指定的数据库和属性指定数据库属性值。例: SELECT DATABASEPROPERTY('sample', 'IsNullConcat') = 0. (IsNullConcat 属性和 CONCAT_NULL_YIELDS_NULL 选项一致，在本章末尾进行了介绍。)
DB_ID([db_name])	返回数据库 db_name 的标识符，即返回当前数据库的标识符。例: SELECT DB_ID('AdventureWorks') =

	6
DB_NAME([db_id])	返回带有标识符 db_id 的数据库名。如果没有指定标识符，就显示当前数据库名称。例: SELECT DB_NAME(6) = 'AdventureWorks'
INDEX_COL(table， i， no)	返回表 table 中的索引列，该索引列由索引标识符 i 及该列在索引中的位置 no 指定。
INDEXPROPERTY(obj_id, index_name， property)	返回指定表标识号、索引或统计名称及属性名称的指定索引值或统计属性值。
OBJECT_NAME(obj_id)	返回有标识符 obj_id 的数据库对象名称例: SELECT OBJECT_NAME(453576654) = 'products'
OBJECT_ID(obj_name)	返回数据库对象 obj_name 的标识符。例: SELECT OBJECT_ID('products') = 453576654
OBJECTPROPERTY(obj_id, property)	返回当前数据库对象。

(作者：SearchSQLServer.com 译者：April 来源：TT 中国)

在 SQL Server 2008 中使用 T-SQL 数据类型（一）

数据类型

所有的列中的数值都必须属于同样的数据类型（除非指定了 SQL_VARIANT 数据类型值）。T-SQL 使用不同的数据类型，具体分类如下：

- 数值型数据类型
- 字符型数据类型
- 临时（如日期或时间）数据类型
- 混合型数据类型
- 附带有 VARDECIMAL 的 DECIMAL 存储类型

下面详细说明所有的数据类型。

数值数据类型

数值数据类型用于代表数据。下表中列出了所有的数值型数据类型：

数据类型	详细说明
INTEGER	表示能够用4个字节保存的整数值，范围包括－2,147,483,648至2,147,483,647。INT是INTEGER的缩写形式。
SMALLINT	表示能够用2个字节保存的整数值。这些值的范围从－32768至32767。
TINYINT	表示能够用1个字节保存的、零以上的正数值。范围为0至255。
BIGINT	表示能够用8个字节保存的整数值。范围为－2^63至2^63 － 1。
DECIMAL(p,[s])	描述定点值。p(精确度)指定数字个数和假定的小数点s（刻度）数字。DECIMAL值可以用5-17个字节保存，由p值决定字节个数。DEC为DECIMAL的缩写。
NUMERIC(p,[s])	和DECIMAL相同。
REAL	适用于浮点值。包括正数值和负数值。正数值范围大约为2.23E － 308至1.79E +308，负数值为－1.18E － 38至－1.18E + 38（0也可以包含在内）。
FLOAT[(p)]	表示浮点值，如REAL。p定义精确度，即p<25表示单精度（4个字节）、p>=25为双倍精密度（8个字节）。
MONEY	表示币值。MONEY值和8字节的DECIMAL值一样，都是小数点后4个数字。
SMALLMONEY	和MONEY数据类型一样，只是用4个字节保存。

字符型数据类型

字符型数据类型有两种普通形式。它们可以是单字节的字符串，还可以使 Unicode 字符串（Unicode 用一些字节指定一个字符）。此外，字符串有固定或可变长度。以下为字符型数据类型：

数据类型	解释
CHAR[(n)]	表示单字节、有固定长度的字符串，其中为字符串内的字符个数。n的最大值为8000。CHAR(n)为CHARACTER(n)的另外一种书写形式。如果省略了n，那么字符串的长度就假定为1。
VARCHAR[(	表示单字节的字符可变长度的字符串（0 < n ≤ 8000）。和CHAR数据类型相反，VARCHAR数据值能够在它的实际长度中保存。这种数据类型有两个同义词：CHAR VARYING和CHARACTER VARYING。
NCHAR[(n)	保存固定长度的Unicode字符。CHAR和NCHAR数据类型之间最主要的区别为NCHAR数据类型的每个字符都可以用2个字节保存，而CHAR数据类型的每个字符串只用1个字节的存储空间。因此，NCHAR列中字符个数最多为4000。
NVARCHAR[	保存具有可变长度的Unicode字符串。VARCHAR和NVARCHAR数据类型之间的最主要的区别为每个NVARCHAR字符都能用2个字节保存，而每个VARCHAR字符只用1个字节的存储空间。因此NVARCHAR列中的字符个数最多为4000。

注：VARCHAR 数据类型和 CHAR 数据类型基本一样，只有一点不同：如果一个 CHAR(n)字符串比 n 个字符要短，那么其余的字符串就用空格来填充。VARCHAR 数据类型经常用实际长度来存储。

（作者：SearchSQLServer.com 译者：April 来源：TT 中国）

在 SQL Server 2008 中使用 T-SQL 数据类型（二）

临时数据类型

T-SQL 支持下面的临时数据类型：

- DATETIME
- SMALLDATETIME
- DATE
- TIME
- DATETIME2
- DATETIMEOFFSET

DATETIME 和 SMALLDATETIME 数据类型指定了日期和时间，每个值都为整数并分别用 4 个字节或 2 个字节保存。DATETIME 和 SMALLDATETIME 的值是作为两个单独的数值保存的。DATETIME 日期值的范围为 01/01/1900 至 12/31/9999。SMALLDATETIME 相应的值为 01/01/1900 至 06/06/2079。时间部分能够再用 4 个字节(或者用两个字节保存 SMALLDATETIME)保存，它表示午夜过后的一秒的三百分之一（DATETIME）或分钟（SMALLDATETIME）。

如果你只想保存日期或时间部分，那么使用 DATETIME 和 SMALLDATETIME 非常不方便。出于这种原因，SQL Server 2008 介绍了新的数据类型：DATE 和 TIME，这两种数据类型只能分别保存 DATETIME 数据的 DATE 或 TIME 部分。DATE 数据类型用 3 个字节的保存并且范围为 01/01/0001 至 12/31/9999。TIME 数据类型用 3 - 5 个字节保存，且精确到 100ns。

DATETIME2 数据类型也是一种新的数据类型，用于保存精确度很高的日期和时间数据。这种数据类型能够根据要求定义可变长度。 存储大小为 6-8 个字节。时间部分的精确度为 100ns。这种数据类型不包括 Daylight Saving Time。

到现在为止我们介绍的临时数据类型都不支持时域。DATETIMEOFFSET 这种新的数据类型就有时区偏移量。所以，它用 6-8 个字节来保存。这种数据类型的其他部分都和 DATETIME2 相应量类似。

T-SQL 日期值默认为单引号或双引号里的字符串，格式为'mmm dd yyyy'（e.g.，'Jan 10 1993'）。（注意月份、日其和年之间的顺序是能够由 SET DATEFORMAT 语句控制的。此外，系统还能识别带有分隔符/或 - 的月份数值）。同样，这一时间值的格式为：'hh:mm'，Database Engine 使用 24 小时制（如 23:24）。

注：T-SQL 支持很多 DATETIME 值得输入格式。就如同你已经知道的，两个对象都能单独分开。因此，日期和时间值能够指定为任意顺序或者单独运用。如果省略其中的一个值，系统就能用默认值（时间默认值为 12:00 AM）。

例 4.4 和 4.5 表示不同方式，使用不同格式书写日期和时间值：

例 4.4

能够使用户以下日期描述：

```
   '28/5/1959' (with SET DATEFORMAT dmy)
'May 28,  1959'

   '1959 MAY 28'
```

例 4.5

以下描述能够使用：

```
    '8:45 AM'
'4 pm'
```

混合型数据类型

T-SQL 还支持一些其他的数据类型，这些数据类型不属于先前描述的任何一种数据类型。

- Binary 数据类型
- BIT
- 大对象数据类型
- CURSOR（在第 8 章中讨论）
- UNIQUEIDENTIFIER
- SQL_VARIANT
- TABLE（在第 8 章中讨论）
- XML（在第 28 章中讨论）
- Spatial（如 GEOGRAPHY 和 GEOMETRY）数据类型（在第 29 章中讨论）
- HIERARCHYID
- TIMESTAMP 数据类型
- 自定义数据类型（在第五章中讨论）

以下部分主要描述每种数据类型（除了在其他章节中之指定的之外）。

Binary 和 BIT 数据类型

BINARY 和 VARBINARY 数据类型是两种二进制的数据类型。它们表示系统内部格式的数据对象。它们用来存储位串，因此这些值都是十六进制的数字。

BIT 数据类型

Binary 和 BIT 数据类型都用一位存储，因此，一个字节能够存储高达 8 位的列。以下是对这些数据类型的属性进行的总结：

数据类型	解释
BINARY[(n)]	指定有固定长度和n个字节的位串（0 < n ≤ 8000）。
VARBINARY[(n)]	指定一个位串的可变长度，最高值为n个字节（0 < n ≤ 8000）。
BIT	用于指定Boolean数据类型，它包含三个值：FALSE、TRUE以及NUL

（作者：SearchSQLServer.com 译者：April 来源：TT 中国）

在 SQL Server 2008 中使用 T-SQL 数据类型（三）

大对象数据库类型

大对象（LOBs）就是最大长度为 2GB 的数据对象。这些数据对象一般用来存储大型的文本数据、加载模块和音频/视频文件。T-SQL 支持两种不同的指定和访问方式：

1、使用 VARCHAR(MAX)， NVARCHAR(MAX)和 VARBINARY(MAX)的数据类型。

2、使用所谓的文本和图像数据类型。

MAX Specifier 是从 SQL Server 2005 开始的，你能用相同的程序模式访问标准数据类型和 LOBs 的值，换句话说，你能用简单的系统函数和字符串进行对 LOBs 操作。

Database Engine 用的是 MAX 分类符，包括以下数据类型：VARCHAR、NVARCHAR 和 VARBINARY 定义可变长度列。当你使用 MAX 默认值（不是确切值时），系统分析这一特殊字符的长度并决定是否将该值当作 LOB 来存储。MAX 分类符表示列值得大小能够达到目前系统的 LOB 最大值（在将来 SQL Server 版本中，MAX 将可能有一个更大值。

尽管数据库系同决定如何存储 LOB，你也可以通过使用 sp_tableoption 系统程序 LARGE_ VALUE_TYPES_OUT_OF_ROW 不考虑默认细则。如果选项值设为 1，那么列中的数据显然就是用的 MAX 分类符，就会与其他的数据区分开进行单独保存。如果选项设定为 0，Database Engine stores 就保存了所有表示行大小的值，一般行数据<8060 字节。

在 SQL Server 2008 中，你可以对 VARBINARY(MAX)列申请新的 FILESTREAM 属性将大型 binary 数据直接存储在 NTFS 文件系统中。这一属性的主要优点就是相应的 LOB 大小仅仅受限于文件系统的列大小。

TEXT、NTEXT 和 IMAGE 数据类型

TEXT、NTEXT 和 IMAGE 数据类型组成了所谓的文本/图像数据类型。IMAGE 类型的数据对象包含所有的数据（下载模块、音频/视频），而 TEXT 和 NTEXT 数据类型的数据对象包含在文本数据（也就是可打印的数据）。

文本/图像数据类型和数据库的其他值区别保存默认值，存储方式为使用指向该数据字段的 B-树型结构（A B-树型结构就是如同树的数据结构，所有远离树根的底端节点数字相同）。而对于含有文本/图像数据类型的列来说，Database Engine 在数据行中存储了 16 字节的指示器，指定能找到数据的地方。

如果文本/图像数据小于 32KB，指示器就指向了 B-树型结构中的根部节点，该节点长度约为 84 个字节。根部节点指向数据的物理模块。如果数据总量大于 32KB，Database Engine 就会在数据模块和根部节点建立中间节点。

由于每个表都包含一个以上有这种数据的列，列中的所有值都存储在一起。然而，一个物理页面能只包含单个表中的文本/图像数据。

虽然文本/图像数据是和所有其他数据分开保存的，你也可以用 sp_tableoption 系统程序的 TEXT_IN_ROW 选项进行修改。用这一选项你能指定最大字节数字，也可以让它和一般数据保存在一起。

文本/图像数据类型有很多局限性。你不能把它们当作本地变量（在存储程序或在 T-SQL 语句中）。同样，它们也不能作为索引的一部分、不能用在下列 SELECT 语句的子句中：WHERE、ORDER BY 和 GROUP BY。所有文本/图像数据类型最大的问题就是你必须使用特殊工具（READTEXT、WRITETEXT 和 UPDATETEXT）对它进行操作。

注：文本/图像数据类型具有 deprecated 特征，并且会迁移到将来的 Database Engine 版本中，用 VARCHAR(MAX)、NVARCHAR(MAX)和 VARBINARY(MAX)代替。

UNIQUEIDENTIFIER 数据类型

从它的名字来理解，UNIQUEIDENTIFIER 数据类型值是唯一辨别数字，它是用 16 个字节存储的二进制字符串。数据类型和 globally unique identifier（全球唯一识别码）密切相关，它保证了在世界范围内的唯一性。因此，通过运用数据类型，你能在分布式计算机系统中用非常独特的眼光识别数据和对象。

使用 NEWID 和 NEWSEQUENTIALID 函数可以进行列或 UNIQUEIDENTIFIER 类型的变量初始化，并且字符串常量是用十六进制数字和连字符这种特殊格式书写的。NEWID 和 NEWSEQUENTIALID 函数在本章以后的系统函数这部分会讲到。

在查询里用关键字 ROWGUIDCOL 能够查看 UNIQUEIDENTIFIER 数据类型列，并且指定包含 ID 值的列。这一关键字不能生成任何列。一个表中有多个 UNIQUEIDENTIFIER 列，但是只有一个列里包含 ROWGUIDCOL 关键字。

SQL_VARIANT 数据类型

SQL_VARIANT 数据类型用于同时存储各种数据类型，如数值、字符串以及日期值等等。只有 TIMESTAMP 类型值不能存储。SQL_VARIANT 列中的每个值都有两部分：日期值和描述值的信息。这些信息包括数据类型的所有属性，如长度、范围和精确度。

T-SQL 支持 SQL_VARIANT_PROPERTY 函数，它表示 SQL_VARIANT 列中每个值得附加信息。SQL_VARIANT 数据类型的用法，查看第五章例 5.5。

注：只有在非常必要的情况下才用 SQL_VARIANT 数据类型表中的列。如果这些值属于不同的数据类型或者在数据库设计过程中很难决定列中的数据类型，那么表中的列就应该包含这种数据类型。

HIERARCHYID 数据类型

HIERARCHYID 数据类型用于保存整个层面。它是作为 Common Language Runtime（CLR）自定义类型安装的，包括一些层级节点上创建和操作的系统函数。以下的一些函

数，属于这种数据类型：GetAncestor()， GetDescendant()， Read()， and Write()（有关这种数据类型的详细介绍不属于本章范围）。

TIMESTAMP 数据类型

TIMESTAMP 数据类型指定其中一栏为 VARBINARY(8)或 BINARY(8)，这是由列中的无效值决定的。系统包含每个数据库的当前值（不是日期或时间），这样在插入或更新含有 TIMESTAMP 列这样的行时就自动增加了。

注：该值不能自动保存在 TIMESTAMP 列中。因为 TIMESTAMP 列常用于查看是否一个具体的行已经在上次访问的时候发生了变化。

用 VARDECIMAL 格式存储的 DECIMAL 数据库

DECIMAL 数据类型一般以固定长度数据存储在磁盘上。自从 SQL Server 2005 SP2 以来，这种数据库就能用新的叫做 VARDECIMAL 存储格式以可变长度列存储。通过使用 VARDECIMAL，你能大大减少 DECIMAL 列的存储空间，在这些列里的值的长度也大不相同。

注：VARDECIMAL 是一种存储格式而不是一种数据类型。

当你需要指定最大可能值范围时 VARDECIMAL 存储格式就很有帮助，通常存储值要更小。 如表 4-1 表示：

精确度	字节个数: VARDECIMAL	字节个数: 固定长度
0 or NULL	2	5
1	4	5
20	12	13

30	16	17
38	20	17

注：VARDECIMAL 存储格式和 DECIMAL 数据类型工作原理一样，都是包括文字和数字的 VARCHAR 数据类型。

表 4-1 为存储 VARDECIMAL 和固定长度的字节个数。

要启用 VARDECIMAL 存储格式，你首先要为数据库启用这种格式，然后为数据库特殊的表启用这种格式。sp_db_ vardecimal_storage_format 系统程序用于第一步，如例 4.6 所示：

例 4.6

```
EXEC sp_db_vardecimal_storage_format 'sample', 'ON';
```

sp_table 选项系统程序中的 VARDECIMAL STORAGE FORMAT 选项常用于为该表开启这种存储格式。例 4.7 就为该项目表启用了 VARDECIMAL 存储格式。

例 4.7

```
EXEC sp_tableoption 'project', 'vardecimal storage format', 1
```

就像你已经了解到的一样，用 VARDECIMAL 存储格式最主要的原因就是减少数据的存储大小。如果你想测试它能够获得多少存储空间，你可以用 sys.sp_estimated_rowsize_reduction_for_vardecimal 动态管理视图。这种管理视图能对这种表进行详细概算。

(作者：SearchSQLServer.com 译者：April 来源：TT 中国)

SQL Server 2008 中新的 datetime 数据类型具有灵活性（一）

在这一技巧系列中我讲了 SQL Server 的 datetime 值的很多方面。我所举的这些例子都是用在 SQL Server 2005 中的，尽管我在那些文章中讨论的东西在 SQL Server 2008 中也同样适用。在这篇技巧中，我将进一步讨论 datetime 值，并特别强调能够用在 SQL Server2008 中新的 datetime 数据类型。它们包括 DATE、TIME、DATETIME2 以及 DATETIMEOFFSET 类型，因为这些类型 T-SQL 语言都支持。

为了说明这些数据类型，我使用了下面的代码在 AdventureWorks2008 抽样数据库中创建了 Sales.OrderDates，并在表中插入了测试数据：

```
USE AdventureWorks2008
GO
IF EXISTS (SELECT table_name
FROM information_schema.tables
WHERE table_schema = 'Sales'
AND table_name = 'OrderDates')
DROP TABLE Sales.OrderDates
GO
CREATE TABLE Sales.OrderDates
(
OrderID INT NOT NULL,
Date_Type DATE NULL,
Time_Type TIME(7) NULL,
DateTime2_Type DATETIME2(7) NULL,
DateTimeOffset_Type DATETIMEOFFSET(7) NULL
```

```
)
GO
INSERT INTO Sales.OrderDates
VALUES(1001, '2008-09-22', '18:27:10.1234567',
'2008-09-22 18:27:10.1234567', '2008-09-22 18:27:10.1234567 -07:00')
```

如果你想测试这些文章中的例子，你也可以在 AdventureWorks2008 数据库或者在另外的数据库中创建这种表。若你决定用 AdventureWorks2008 数据库，你可以从 CodePlex 中获取一些必要的文档。

DATE 数据类型

在 SQL Server 2008 之前，主要 datetime 数据类型就是 DATETIME 和 SMALLDATETIME。每一种情况中存储在这一类型中的值包括日期和时间，也没有办法只存储一部分。然而，SQL Server 2008 改变了所有的 DATE 和 TIME 数据类型。

从它的名称我们也可以知道，DATE 数据类型包含了一个日期值。这一日期值可以从公元一世纪一月一日到公元 9999 年 12 月 31 日。这一日期值包括年、月和天。例如，以下 SELECT 语句就可以从 OrderID 和 Date_Type 列中找回数据。

```
SELECT OrderID, Date_Type FROM Sales.OrderDates
```

Date_Type 列是由 DATE 数据类型设置的，所以这些 retrieved value 中只包括了一个日期，其结果如下：

```
OrderID Date_Type
1001 2008-09-22
```

在这种情况下，DATE 值中就包括了年、然后是月和日期。但是，重新获取的日期格式是目前设置在 SQL Server 中的语言。

如果你想将 datetime 值返回到一个 DATE 值，你可以改变它，如下面的 SELECT 语句所示：

```
SELECT OrderID,  CAST(DateTimeOffset_Type AS DATE) AS ConvertedType
FROM Sales.OrderDates
```

这时，该语句从 DateTimeOffset_Type 列中找回了 DATE 值，它是用 DATETIMEOFFSET 数据类型设置的（在下文中会有详细解释）。尽管 retrieved value 中包含了日期和时间数据，这一语句返回的结果和之前的语句返回的结果一样。 换句话说，就是时间部分的值是可以忽略不计的。

将数据插入到 DATE 列中，你就能指定一个 date 值或者 datetime 值，如下所示：

```
INSERT INTO Sales.OrderDates (OrderID,  Date_Type)
VALUES (1002,  '2008-09-20');
INSERT INTO Sales.OrderDates (OrderID,  Date_Type)
VALUES (1003,  '2008-09-21 18:27:10.1234567');
```

指定了 datetime 值后，SQL Server 就自动将这些值转换成 DATE 数据类型，也就意味着只有日期部分被保存了。

TIME 数据类型

TIME 数据类型只保存在 time 值中。这些值自身就支持小数后 7 位的精确度。也就是说小数部分最多可支持 7 个小数位。这是在和 DATETIME 值相比较的情况下得出的结论，DATETIME 值只支持小数点后 3 位。

你 T-SQL 语句中指定 TIME 数据类型后，就能通过包括括号中的插入语指定存储值得精确度。例如，指定 7 个小数位，你就可以指定 TIME(7)。要指定 5 个小数位，你就可以指定成为 TIME(5)等等。如果你不想指定精确度，就假定是 7。

TIME 数据类型返回的数据格式为：时、分、秒和小数秒（fractional second）。例如，下面的 SELECT 语句就是从 Time_Type 列中重新获取数据。

```
  SELECT OrderID,  Time_Type FROM Sales.OrderDates
```

Time_Type 列是由 TIME(7)数据类型设置的，所以她返回的数据类型如下所示：

```
OrderID Time_Type
1001 18:27:10.1234567
```

注意这些列返回的数据只有时间和小数精确度为 7 位的数值。

看到这些 DATE 数据类型时，你还能够从 datetime 列中返回一个 TIME 值。例如：下面的语句是从 DateTimeOffset_Type 列中重新找回数据并将这些数据值变换为 TIME(7)：

```
  SELECT OrderID,  CAST(DateTimeOffset_Type AS TIME(7)) AS ConvertedType
FROM Sales.OrderDates
```

这一 SELECT 语句返回的结果和之前的 SELECT 语句返回的结果相同。

但是你也可以在转换数据时另外指定一个精确度，如下所示：

```
   SELECT OrderID,  CAST(DateTimeOffset_Type AS TIME(5)) AS ConvertedType
FROM Sales.OrderDates
```

小数秒部分现在包括五个小数位：

```
OrderID ConvertedType
1001 18:27:10.12346
```

注意小数秒现在已经舍弃取整数了。原先的小数秒为.1234567。但是如果你指定精确度为 5 位，SQL Server 就会精确舍弃小数部分的"67"，在结果中只剩下.12346，而不是.12345。将数据插入到 TIME 列中后，你就可以指定一个 time 值或者是 datetime 值了。如下列 INSERT 语句所示：

```
INSERT INTO Sales.OrderDates (OrderID,  Time_Type)
VALUES (1004,  '18:27:10.1234567');
INSERT INTO Sales.OrderDates (OrderID,  Time_Type)
VALUES (1005,  '2008-09-20 18:27:10.1234567');
```

在这两种情况下，只有时间值才能插入到 Time_Type 列中。

(作者：SearchSQLServer.com 译者：April 来源：TT 中国)

SQL Server 2008 中新的 datetime 数据类型具有灵活性（二）

DATETIME2 数据类型

DATETIME2 数据类型和 DATETIME 数据类型很相似。它们之间的不同之处就是 DATETIME2 支持范围更广的数据（和 DATE 一样）并且精确性也更高（和 TIME 一样）。如同 TIME 数据类型一样，你可以指定它的精确度。

换句话说，DATETIME2 主要是连接 DATE 值以及 TIME 值。例如，用下面 SELECT 语句就可以从 DateTime2_Type 列中返回数据，这些数据被设置成为 DATETIME2 列。

```
SELECT OrderID,  DateTime2_Type FROM Sales.OrderDates
```

语句返回数据的结果如下::

```
OrderID DateTime2_Type
1001 2008-09-22 18:27:10.1234567
```

正如你看到的一样，DATETIME2 首先提供的是日期值，然后是时间，精确度为 7 位。除了小数秒 7 位之外，这个值看起来就好像是 DATETIME 值或者是一个后面附着有 TIME 值的 DATE 值。

你还可以将其他的 datetime 转换为 DATETIME2 值，如下列语句所示：

```
```

```
SELECT OrderID,  CAST(DateTimeOffset_Type AS DATETIME2(7)) AS ConvertedType
FROM Sales.OrderDates
```

注意我在将 DATETIMEOFFSET 值转换到 DATETIME2 的过程中，制定的精确度是 7 位。这一语句返回的结果和先前的语句返回的结果一样。

但是如果你转换其他类型的数值，你得到的结果就不一样了。例如：下面的语句从 Date_Type 列中找回的数据就属于 DATE 数据类型，将这些数据转换成 DATETIME2(7)：

```
   SELECT OrderID,  CAST(Date_Type AS DATETIME2(7)) AS ConvertedType
FROM Sales.OrderDates
```

在 SQL Server 转换这些值时，它给新值增加一个默认时间，其结果显示如下：

```
OrderID ConvertedType
1001 2008-09-22 00:00:00.0000000
```

注意这些时间都是零（精确度为 7 位），也就是 24 小时的第一个值。无论你何时需要假定时间（如在这种情况下），所有这些零都用到了，也就是说默认时间值为 00:00:00.0000000。

然而日期的默认值很不一样。在下面的语句中，我将 Time_Type 列中属于 TIME 数据类型的数值转换到了 DATETIME2(7)中：

```
SELECT OrderID,  CAST(Time_Type AS DATETIME2(7)) AS ConvertedType
FROM Sales.OrderDates
```

因为 TIME 值没有包括这些日期，SQL Server 假定了一个日期，结果如下：

```
OrderID ConvertedType
1001 1900-01-01 18:27:10.1234567
```

所以，在没有日期默认值时日期显示的是 1900 年 1 月 1 日。

DATETIMEOFFSET 数据类型

DATETIMEOFFSET 数据类型和 DATETIME2 的数据类型一样，除此之外，DATETIMEOFFSET 值还增加了很重要的一项：时区偏移（time-zone offset）值。该偏移值代表一系列在 Coordinated Universal 时间（UTC）前后小时和分钟的数字。正数就代表在 UTC 基础之上增加的时间数，这样得到的就是本地时间。负数是从 UTC 时间的基础上减去这个数字就是当地时间。

我们看看下面的一个例子，这样更易于理解。下面的 SELECT 语句用于从 DateTimeOffset_Type 列中找回数据，这些数据就属于 DATETIMEOFFSET 数据类型：

```
SELECT OrderID,  DateTimeOffset_Type FROM Sales.OrderDates
```

在下面的结果中，你能看到 DATETIMEOFFSET 值包括时区偏移数值 07:00。

```
OrderID DateTimeOffset_Type
1001 2008-09-22 18:27:10.1234567 -07:00
```

返回值表示你必须从 datetime 值中减去七小时，这样才能得到当地时间。

在你将另外一种 datetime 值转换为 DATETIMEOFFSET 值时，时区偏移值就默认为+00:00。例如：以下 SELECT 语句就将 DATETIME2 转换为 DATETIMEOFFSET：

```
SELECT OrderID,  CAST(DateTime2_Type AS DATETIMEOFFSET(7)) AS ConvertedType
FROM Sales.OrderDates
```

结果如下：

```
OrderID ConvertedType
1001 2008-09-22 18:27:10.1234567 +00:00
```

现在的时区偏移值为+00:00，在这种情下，UTC 时间和当地时间相同。

如果你想将 DATE 值转换成为 DATETIMEOFFSET，那么该数值的时间部分就都是零。例如，以下语句就用于 Date_Type 值转换成为 DATETIMEOFFSET(7)：

```
SELECT OrderID,  CAST(Date_Type AS DATETIMEOFFSET(7)) AS ConvertedType
FROM Sales.OrderDates
```

返回结果如下：

```
OrderID ConvertedType
1001 2008-09-22 00:00:00.0000000 +00:00
```

注意时间和时区偏移值都设为零。

但是，如果你将 TIME 值、返回值的时间部分都设为 1900 年 1 月 1 日（默认值），如以下 SELECT 语句所示：

```
SELECT OrderID,  CAST(Time_Type AS DATETIMEOFFSET(7)) AS ConvertedType
FROM Sales.OrderDates
```

该结果就显示被转换了的数据：

```
   OrderID ConvertedType
1001 1900-01-01 18:27:10.1234567 +00:00
```

时间被设成默认值，时区偏移值设为+00:00。只有时间值自身才反映出原始值。

你可以看出，DATETIMEOFFSET 数据类型和 DATE、TIME 以及 DATETIME2 数据类型大大扩展了 SQL Server 的 datetime 容量。现在你可以将日期和时间值当作独立值来运用，运用范围更广的日期值并将 time 值定义得更加精确。想查询更多有关这些数据类型的信息，请查看数据 Microsoft SQL Server 2008 联机丛书。

（作者：SearchSQLServer.com 译者：April 来源：TT 中国）

在 SQL Server 2008 中用 CDC 提高 BI 报告的准确性

在大型商业智能系统或数据仓储上工作时，我们常常面临的问题就是数据及时性。本篇技巧向你介绍了 SQL Server 2008 中的 BI 特征：change data capture（CDC），CDC 能基于来源处理系统中的数据改变通知帮你在数据仓储或专用数据库实时更新数据。这在数据仓库和商业智能情况中是一个详细定义问题，但是是非常普遍的一个问题。

过去，SQL Server 管理员和开发员对解决方案特别依赖，如自定义代码或用 SQL 复制得到同样的结果。微软 SQL Server 2008 CDC 包含以下性能：在 ETL 过程中解决数据潜伏期问题。由于数据管理和数据仓储中的 ETL 许多特性，微软知道今天才进入 CDC 这个领域。今天市场上的其他产品包括 Attunity Integration Suite、Informatica PowerExchange CDC 和 DataMirror 最近都被 IBM 收购了。那些例子用 SQL Server 2008 CDC 提高数据源到数据库的数据及时性都是基于 SQL Server 2008 中的一些特征。

为什么要安装 CDC？

这里是个很常见的情况：你已经通过从 SSIS 数据源下载数据安装了数据库或者数据仓库（SQL Server Integration Services）。事物分析师已经对数据业务报告建立了 cube。每晚事物用户就会将数据填入数据仓库满足报告需要，你需要跟他们联系。 但是如果你已经开始预习给 beta 用户的解决方案，那么报告要求就有变化。现在的事务每小时运行一次报告。

所以，现在你不得不用 ETL 设计并且每小时更新数据。好的一面就是数据仓库、商业智能以及报告解决方案产生了实际效果并且还能收到。坏的一面就是也许你并没有为频繁更新设计或者架构系统。现在你面临的问题就是如何让数据更新越来越接近实际时间。这些问题涉及的范围从性能到可伸缩性然后再到流通问题。

CDC 主要观念就是基于数据源的 notification 将数据从数据源迁到数据仓库（典型的是事务数据库系统），即当数据在源处进行更新的时候。这就是说不是每一种数据源订阅 CDC notifications 都是合适的。股票贸易数据库或订单登陆系统中的高事务工作量就可能不是很理想。更加合适的可能就是数据源，如 CRM 客户数据、产品数据或者其他类型的查询数据。

怎样激活 CDC?

在 SQL Server 2008 中激活 CDC 的方法为：首先配置数据库激活该性能。第二，确定文件组保存更改数据，最后就是确定哪个表将出现在 CDC notifications 中。

你可以用新的 CDC 以下两个函数 cdc.fn_cdc_get_all_changes_ 和 cdc.fn_cdc_get_net_changes_查询更改数据。这两个函数之间主要的区别就是 get all changes 函数将所有变化，这些变化出现在日志序列码中的相应表的每个行里。而 get net changes 函数返回的却是反映该数据多个变化的一个行，所以你就不能收到包含上一个查询以后的每个变化这样一个截然不同的图。除了这些新函数之外，你就必须用 CDC 函数：sys.fn_cdc_map_time_to_lsn 将 LSNs 映射到时间参数解释合适的 LSN。

只有替换你为捕获源表变化配置的参数你才能用这些函数。用法如下：

```
SELECT * FROM
cdc.fn_cdc_get_all_changes_HR_Department(@from_lsn,  @to_lsn,
'all');
```

在上面的这个例子中，我选择返回所有记录，这样就可以包含所有 LSN 的变化。另外一个选择就是 all update old″，它可以返回所有的变化，还可以返回更新之前的包含列值的行以及更新后的这个行。如果你想从 LSN 中获得这些值或者是将这些值潜入 LSN 中，你应该首先从上面的 map 函数中找到合适的 LSN，如下：

```
    DECLARE @from_lsn binary(10),  @to_lsn binary(10);
SELECT @from_lsn = sys.fn_cdc_map_time_to_lsn('smallest greater than or equal',  GETDATE()-1);
SELECT @to_lsn = sys.fn_cdc_map_time_to_lsn('largest less than or equal',  GETDATE());
```

注意单引号中的参数。就是说 SQL Server 要比使用方法要大/小，包含/不包含和指定值相应的时间查询的记录。你可以选择：largest less than， largest less than or equal， smallest greater than or smallest greater than or equal。

最后要说的是，其实并没有能够在 SQL Server 2008 数据库激活 CDC 的方法以及在数据组中增加数据仓库这种性能的方法。事实上他们在《微软 SQL Server 联机从书》中总结得很好。

当你在优化数据仓库或 BI 时，为激活 CDC 选择一种合适的数据源是一个很重要的决策。其他的 CDC 工具在创建和配置时都有一些不同。但是在你将 SQL Server 源迁入到 SQL Server 2008 版本中时，这种性能就自然被加上了。然后你可以用 SQL Server 和 SSIS 中 built-in 函数获取你的 ETL 程序变化数据。在更短的间隙时间内只采集改变的数据变得越来越常见，这样做将会给你的报告方案事务用户增加数据可用性。

（作者：Mark Kromer 译者：April 来源：TT 中国）

在 SQL Server 2008 中安装安全审计

SQL Server 2008 引进了一种新的审计性能，它能使 DBA 追踪数据库的使用并进行详细审计。我们能在服务器和数据库上安装审计，在单独的数据库对象上激活并用各种不同的格式保存，如二进制文件或 Windows Application 日志。

在 SQL Server 2008 里安装审计，步骤如下：

- 给每个 SQL Server 2008 具体实例创建一个 SQL Server 审计
- 创建服务器审计规范、数据库审计规范或者其中的一个
- 激活 SQL Server 审计
- 查看审计数据

在这一技巧中，我将复习其中的每一步并举例说名它们怎么进行的。注意大多数情况下这些例子是基于 T-SQL 语句的。但是，这些步骤也能够用 SQL Server Management Studio 界面来执行。如果想了解更多有关 SQL Server Management Studio 用法的信息，请查看 Microsoft SQL Server 联机丛书。

1、创建 SQL Server 审计

第一步你应该在 SQL Server 2008 的一个实例上创建审计，这样就能创建一个 SQL Server 审计。审计就是为与数据库引擎相关的具体事件集合配置的安全对象。你可以在 SQL Server 2008 里的一个实例上创建多个审计。

在创建审计时，你必须给它指定一个名称和事件输出的目标位置。目标文件可以是二进制的文件、Windows Security 日志或 Windows Application 日志。你还可以给审计对象指定一个或更多个可选参数。

你可以用 CREATE SERVER AUDIT 语句，如下所示：

```
    USE master
GO
CREATE SERVER AUDIT SrvAudit
TO FILE (FILEPATH='C:\Data',  MAXSIZE=5 MB)
WITH (QUEUE_DELAY = 3000)
```

注意，你必须在主数据库中创建一个审计。由于审计是和 SQL Server 实例相联系的，因此你不能在用户数据库中创建。

第一行的 CREATE SERVER AUDIT 语句仅规定审计名称（如 SrvAudit）。第二行的 TO 子句确定事件输出时的目标位置。例如，我想将输出结果保存在文件中，所以我必须指定 TO FILE 并规定 FILEPATH 值。注意这只是一个路径名。SQL Server 将自动命名输出文件，如下所示：

```
<audit_name>_<audit_GUID>_<partition_number>.sqlaudit
```

在上面的例子中，TO FILE 语句还包括了 MAXSIZE 参数，MAXSIZE 参数将文件大小限制为 5 MB。该参数是 TO FILE 子句可选参数之一。如果你将审计数据迁到 Application 日志或 Security 日志，你就只需要指定日志名选项，示例如下：

```
CREATE SERVER AUDIT SrvAudit2
TO APPLICATION_LOG
WITH (QUEUE_DELAY = 3000)
```

就像你看到的一样，TO FILE 子句已经被 TO APPLICATION_LOG 子句所替代并且还没有另外规定其他的参数。

最后一行在 CREATE SERVER AUDIT 语句中的就是一个 WITH 子句。该子句支持很多个选项，限制了创建审计的方法。在这种情况下，我使用的是 QUEUE_DELAY 参数并将它的值设为 3000。这个参数指定了在创建审计之前要耗费的毫秒数并且。默认数字为 1000 毫秒（即 1 秒）。

要了解所有 CREATE SERVER AUDIT 语句可选择项和本篇文章中的其他语句，请查看 Microsoft SQL Server 联机丛书。

(作者： Robert Sheldon 译者：April 来源：TT 中国)

在 SQL Server 2008 中安装安全审计（二）

2、创建服务器审计规范

你创建 SQL Server 审计之后，必须创建一个服务器审计规范或者是一个数据库审计规范或者是其中的每个。一个服务器审计规范就是和具体的 SQL Server 审计相关的一个或多个服务审计。活动组就数据库引擎暴露出来的一组相关的事件，例如，我们在进行安全审计操作时，SERVER_OPERATION_GROUP 行动组就出现了，如当用户在改变服务器设置时。你可以在每个审计上只创建一个服务器审计。但是，你可以对审计规范增加多个活动组。创建一个服务器审计规范，你需要在主数据库上运行 CREATE SERVER AUDIT SPECIFICATION，如下所示：

```
USE master
GO
CREATE SERVER AUDIT SPECIFICATION SrvAuditSpec
FOR SERVER AUDIT SrvAudit
ADD (SUCCESSFUL_LOGIN_GROUP),
ADD (FAILED_LOGIN_GROUP)
WITH (STATE=ON)
```

第一行 CREATE SERVER AUDIT SPECIFICATION 语句规定了审计规范名（SrvAuditSpec）。第二行 FOR SERVER AUDIT 子句指定了与审计规范相关的审计名（SrvAudit）。第三行和第四行为 增加了规范活动组的 ADD 子句。在这种情况下，我增加了 SUCCESSFUL_LOGIN_GROUP 和 FAILED_LOGIN_GROUP 活动组，跟踪试图登录到 SQL Server 实例的安全主管。

最后一行 CREATE SERVER AUDIT SPECIFICATION 语句为 WITH 子句。WITH 子句包括激活规范的在 STATE 参数。默认值不能激活审计规范（STATE=OFF）。如果你在创建时不能激活审计规范，你就必须过段时间再激活，在你能够审计活动组之前进行激活。

创建数据库审计规范

和服务器的审计规范不一样，数据库审计规范是具体针对数据库的。但是它和服务器审计规范相同的是，你可以增加审计活动组，但是它们仅仅针对数据库。此外，你可以给规范增加单独的审计活动。审计活动就是数据库具体的活动，如删除数据或运行存储程序。

创建数据库审计规范，在目标数据库中运行 CREATE DATABASE AUDIT SPECIFICATION 语句，例如：

```
USE AdventureWorks2008
GO
CREATE DATABASE AUDIT SPECIFICATION DbAuditSpec
FOR SERVER AUDIT SrvAudit
ADD (DATABASE_OBJECT_CHANGE_GROUP),
ADD (SELECT, INSERT, UPDATE, DELETE
ON Schema::HumanResources BY dbo)
WITH (STATE=ON)
```

第一行 CREATE DATABASE AUDIT SPECIFICATION 语句指定了规范（DbAuditSpec），第二行为 FOR SERVER AUDIT 子句，用它可以判断和规范相关的审计。接下来，我增加了一个审计活动组，在这里就是 DATABASE_OBJECT_CHANGE_GROUP。在对 AdventureWorks2008 数据库执行 CREATE、ALTER 或 DROP 语句时就会出现这个活动组。

第二个 ADD 子句制定了单独审计活动，而不是一个活动组。这样，审计活动就是 SELECT、INSERT、UPDATE 和 DELETE。但是你要注意，下面一行包含一个 ON 子句指定的 HumanResources schema 和 dbo 安全主管。结果，只要 dbo 在 HumanResources schema 中查询一个对象或在 AdventureWorks2008 数据库中插入、更新或删除，SQL Server 就会将事件记入日志。

最后，CREATE DATABASE AUDIT SPECIFICATION 语句中的最后一个子句就是 WITH 子句。跟上次一样，你可以在操作完之后就激活审计规范或者过一段时间之后再进行激活。

3、激活 SQL Server 审计

和我们刚刚回顾的审计规范一样，CREATE SERVER AUDIT 语句中的 WITH 子句并不支持 STATE 参数。也就是说你必须在单独的一步中激活审计，如下面的语句中所示：

```
USE master
GO
ALTER SERVER AUDIT SrvAudit
WITH (STATE=ON)
```

你可以看到，我使用的是 ALTER SERVER AUDIT 语句更改我之前创建的 SQL Server 审计（SrvAudit）。ALTER SERVER AUDIT 语句中的 WITH 子句支持被我设置成 ON 的 STATE 参数，SQL Server 会审计这些具体事件。

4、查看审计数据

你可以在 SQL Server Management Studio 中用 Log File Viewer 查看审计数据。另外如果你创建 SQL Server 审计将事件保存到 Application 日志或者 Security 日志，这样你就可以用 Event Viewer 查看这些数据。我认为回顾事件信息最简单的方法就是将审计数据保存到一个二进制文件中，然后用 Log File Viewer 回顾这些数据。

要访问 Log File Viewer，就要打开 SQL Server Management Studio，扩展 Security 节点。接下来选择你要复习的审计，然后点击 View Audit Log 发送 Log File Viewer。图 1 表示以上的例题中 SQL Server Audit (SrvAudit)事件实例。

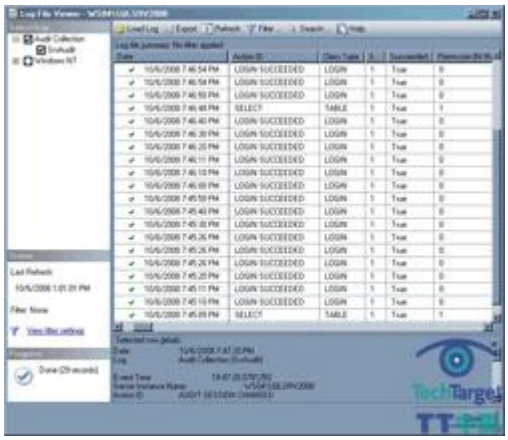


图 1：在 Log File Viewer 中查看审计数据

以上就是安装审计以及回顾审计数据的步骤。SQL Server 2008 让这些步骤比以前版本执行更加简单。你只需要简单创建 SQL Server 审计，并附上一到两个审计规范，然后激活。其余的工作都由 SQL Server 完成。要了解更多有关审计方面的信息，请查看微软 SQL Server 联机丛书。

文章内容，文章内容文章内容文章内容文章内容文章内容文章内容文章内容文章内容文章内容文章内容文章内容文章内容文章内容文章内容文章内容……文章内容文章内容文章内容文章内容文章内容文章内容文章内容文章内容文章内容。

（作者： 译者： 来源：TT 中国）（宋体 10，居右对齐）

你需要做的、有关 SQL Server 2008 新的安全特性工作

今年之前，我就写了一些 SQL Server 2008 安全特征如何帮你降低风险的文章。现在 SQL Server 2008 已经发布了一段时间了，那么应该如何保证呢？它是不是能够抵御住一些常见的攻击？

我们用一些攻击评估工具测量，它还是比较好的。这的确是我见过的微软最好的裸机数据库安全态势。但遗憾的是没有一个主流商业数据库攻击扫描仪是我们能够用来真正挖掘 SQL Server 弱点并且支持 SQL Server 2008 的。至少我知道的就是这样。所以从这个角度来说，仍有待观察。

除了攻击扫描之外，还有一点很重要：仅因为一个软件（如 SQL Server 2008 开始的时候是安全的）并不是说它在你的环境中是安全的。只要你能很好处理这些漏洞，你就不用担心 SQL Server 问题。

以下对微软不能为你解决的一些问题的总结：

- 缺少合理政策，如强烈需要 SQL Server 密码
- 数据库相关软件没有写入安全保障
- 缺省 OS 和 SQL Server 补丁，它们允许攻击者用 Metasploit 或其中的商业选项对你的数据库进行远程访问。
- 改变管理缺口和疏忽状况，如多重管理使得配置根据数据库环境而发生改变，这样就导致了 SQL Server 安全漏洞。
- 不严谨的网络设计让人们从内网和外网中获得攻击数据库环境的机会。
- 缺乏安全监控和事故管理就能帮你防止数据库受到攻击或者至少让它们免遭太大的打击。
- Windows 操作系统表明存在的调查结果。

- Web 应用输入允许 SQL 注入的验证缺陷。
- 包括书写得当、创建较好的 SQL Server 安全目标就是让数据库安全在事务范围内获的优先权。

即使 SQL Server 2008 包括所有最新、最好的安全特征，你仍然会面临数据库攻击问题。所以，我们还没有脱离危险，以前 SQL Server 旧版本也没有摆脱过。也就是说，我们用 SQL Server 2008 可能会更好。你唯一要知道的方法就是看看你现在的位置，保持可视性、把握并且控制你环境中新的安全漏洞。需要警惕的并不是一切，就是这一件事情。

(作者：Kevin Beaver, CISSP 译者：April 来源：TT 中国)

SQL Server 2008 Reporting Services 的新特性

随着 SQL Server 2008 的发布，Reporting Services （SSRS）就是他的第三版。而之前的版本包括许多新特征。本篇文章将讨论最重要的、而且是最有用的一些新特征，这些特征让 SQL Server 2008 Reporting Services 更值得你升级。

不再需要 IIS

之前的两个版本主要是 IIS 包括 RS Web 和 Report Manager Web 应用。出于安全考虑，许多公司已经制定出了不在和 SQL Server 机器上安装 IIS 的规定。对于一些潜在的用户来说，IIS 的依赖就成为使用 SSRS 最具吸引力的特征之一。 基于用户的回馈，微软重新构架了 SSRS 2008 并取消了这种依赖。

SSRS 现在是通过 HTTP.SYS 处理 HTTP，HTTP.SYS 是一种本地内核能够获取并处理 HTTP 请求如 IIS。如果这让你听起来很熟悉，那是因为 SQL Server 2005 Native XML Web Service 也用了相同的机制——这是一种很有意思的特征，但是在 SQL Server 2008 中没有。微软还表示这种安装提供了更好的性能和可预测性。我们升级 Reporting Services Configuration 工具来为 Report Manager 和 Reporting Server 服务提供管理性能。

更好的内存管理

在拥挤的环境中部署了 SSRS 的用户可能已经注意到当一些并发用户访问多页的报告时它的性能就会退化。这种放缓的原因就是 SSRS 将报告迁入内存中，如果内存需求太高就会造成问题。对于大型报告，它还会在用户查看时进行呈递。这种按需处理减少了在服务器付负荷，因为在用户没有查看这些页面时它就没有被提取。除增加内存外，你现在还能设置有关 SQL Server Reporting Services 占用内存空间大小的开端。

导出到 Microsoft Word

2004 年 SSRS 2000 发布时，对这种导入需求就特别强烈，这主要是因为有 Word 用户比 Excel 用户更多。由于某种原因，这一特性在 SQL Server 2005 中没有实现。但是最终在这里得到了实现，并且是除了所有其他的导出选项之外最受欢迎的。

绘图工具得到了改良

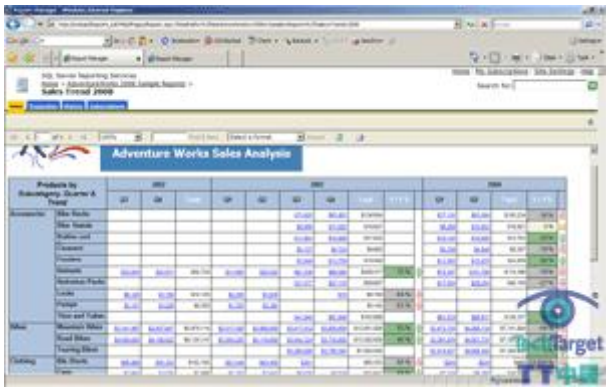
去年微软已经收购了 SSRS 的 Dundas Data Visualization 技术。Dundas 是一个专注于为研究人员开发强大的视图工具的公司，如图、测量、地图和日历装置。为它们购买代码基（ code base）会让微软提供更好的带有 SSRS 的集成工具，把握它们将来的方向。Dundas 工具从视觉的角度来说一直都是很好的工具，所以我们对于这一点并不感到奇怪：新的绘图工具和之前 SSRS 版本的绘图工具比较起来就像是白天和黑夜。

在我看来，如果你在报告中进行了大量的绘图，这就能证明你升级到 SQL Server Reporting Services 所做出的努力。这种新的绘图领域包括许多其他的图表类型，如圆柱体、金字塔、雷达和漏斗。其他一些有用之处就是会定时添加标签，将零碎地片组合在一起，达到 2-D 的视觉效果。

Tablix 数据区

这种新的数据区结合了表和矩阵数据区的功能。这个区域能允许你合并多个列组合和行组合，允许你创建有动态行和动态列结果的报告。你现在更加容易创建嵌入和递归式的组合并让它们彼此紧靠在一起，请阅读《微软联机丛书 Tablix Data Region 部分》。

图一表示 Tablix control 在嵌入集合和主要性能指示器的报告中运用。



图一：Tablix control 在嵌入集合和 KPIs 中的 SSRS 2008 报告中使用

CSV 导出增强

逗号分割值（Comma-separated value，CSV）到处对于自动进行报表操作很有益处。如果你对从报告中提取数据成分很感兴趣的话。SSRS CSV 导出到各种版本中，包括导出的数据和设计规划。但是 CSV 中包含的这些报告设计原理有带来了额外的工作和复杂因素。新的输出格式显得更加清晰明了，也更加易于进行自动操作。

对 SSRS 2008 进行认真考虑有很多重要原因。请记住，和集成服务一样，SSRS 是一种你能够独立安装在专门服务器上的工具。也就是说，如果公司不准备迁移到 SQL Server 2008，你就可以只需要升级你的报告服务器并使用新特征，把 2000 或者 2005 当作数据源。

（作者：Roman Rehak　译者：April 来源：TT 中国）