SQL Server 中的日期/时间

数据类型

SQL Server 中的日期/时间数据类型

在 SQL Server 2005 中使用日期/时间值有时候会很模糊、会觉得很迷茫。因为日期/时间数据类型同时存储日期和时间值，而这些值的操作并不总是一个简单的过程，或者看起来不简单的。

SQL Server 2005 的 DATETIME 和 SMALLDATETIME 基础

理解 SQL Server 的日期/时间数据类型是有一定难度的，尤其是混合使用 TIMESTAMP 的时候。在关于日期/时间的这一系列的第一部分，你将了解到关于数据是如何存储在 DATETIME 和 SMALLDATETIME 的基础知识，以及大致地了解 TIMESTAMP 数据类型——它经常与两种主要的日期/时间数据类型相混淆。

- SQL Server 2005 的 DATETIME 和 SMALLDATETIME 基础（一）
- SQL Server 2005 的 DATETIME 和 SMALLDATETIME 基础（二）
- SQL Server 2005 的 DATETIME 和 SMALLDATETIME 基础（三）

SQL Server 中日期/时间值到字符类型的数据转换

在本文中，我将解释如何将 DATETIME 和 SMALLDATETIME 数据类型转换成字符数据，以及如何将字符数据转换成日期/时间数据。具体来说，本章将介绍 Transat-SQL 支持的两个内置 SQL Server 数据转换方法——隐式转换和显式转换。在前一篇的技巧《SQL Server 2005 的 DATETIME 和 SMALLDATETIME 数据基础》中，我已经阐述了 SQL Server 是如何使用 DATETIME 和 SMALLDATETIME 数据类型来存储日期/时间数据的。在本文中，我将解释日期/时间数据是如何转换成字符数据和字符数据是如何转换成日期/时间数据以及 Transact-SQL 是如何支持这两种执行这些数据转换的方法——隐式转换和显式转换。

- SQL Server 中日期/时间值到字符类型的数据转换（一）
- SQL Server 中日期/时间值到字符类型的数据转换（二）

正确使用 SQL Server 的 datetime 函数

Transact-SQL 包含一组函数可以用于检索当前的日期和时间或一个 DATETIME 或 SMALLDATETIME 值的某个部分。比如，你可以在 DATETIME 值中提取日、月或年以及季度、周、小时甚至毫秒。在本文中，我将对这些函数进行阐述并举例说明如何使用 SQL Server 中的这些函数来查询数据的。注意，本文假定你已经具备了一定的 T-SQL、DATETIME 和 SMALLDATETIME 数据类型的知识。想获得更多这些类型的信息，你可以阅读这一系列的第一部分“SQL Server 2005 的 DATETIME 和 SMALLDATETIME 基础”。

- 使用 SQL Server 的 datetime 函数：GETDATE、DATENAME 和 DATEPART（一）
- 使用 SQL Server 的 datetime 函数：GETDATE、DATENAME 和 DATEPART（二）

使用 DATEADD 和 DATEDIFF 计算 SQL Server 的 DATETIME 值

在 SQL Server 数据库中，DATETIME 和 SMALLDATETIME 值是以整数存储的。然而，与整数不同的是，它们不能直接地进行数学运算。尽管如此，有时候还是需要在日期/时间值中添加或减去一个时间间隔。比如，你可能想在一值上加一些月数或天数，或者甚至可能是小时数。你甚至可能想比较两个日期/时间值以便确定它们之间的时间间隔，如相差的天数或年数。为了简化这些类型的计算，Transact-SQL 支持两个重要的日期/时间方法：DATEADD 和 DATEDIFF。

- 使用 DATEADD 和 DATEDIFF 来计算 SQL Server 的 DATETIME 值（一）
- 使用 DATEADD 和 DATEDIFF 来计算 SQL Server 的 DATETIME 值（二）

SQL Server 2008 中新的 datetime 数据类型

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。

- SQL Server 2008 中新的 datetime 数据类型具有灵活性（一）
- SQL Server 2008 中新的 datetime 数据类型具有灵活性（二）

SQL Server 2005 的 DATETIME 和 SMALLDATETIME 基础（一）

理解 SQL Server 的日期/时间数据类型是有一定难度的，尤其是混合使用 TIMESTAMP 的时候。在关于日期/时间的这一系列的第一部分，你将了解到关于数据是如何存储在 DATETIME 和 SMALLDATETIME 的基础知识，以及大致地了解 TIMESTAMP 数据类型——它经常与两种主要的日期/时间数据类型相混淆。

在 SQL Server 2005 使用日期/时间值有时候会很混淆。因为日期/时间数据类型同时存储日期和时间值，而这些值的操作并不总是一个简单的过程，或者看起来不简单的。

当操作这类数据时，对于这些数据类型有一个基本的了解是很有重要的。在本文中，我将介绍 SQL Server 中的两种基本的日期/时间数据类型——DATETIME 和 SMALLDATETIME，以及数据是如何在这两种类型中存储的。另外，我还将概括介绍 TIMESTAMP 数据类型，这样你就可以了解到它与两种日期/时间数据类型的不同。

SQL Server 的 DATETIME 数据类型

毫无疑问，DATETIEM 数据类型对你的应用是最有用的。DATETIME 字段（或一个变量）中的值是以两个 4 位整型存储的。第一个整型表示日期，而第二个整型表示时间。

我们是从基准日期 1900 年 1 月 1 日开始的计算日期的整数的。整数表示该日期之前或之后的天数。因此，DATETIME 数据类型仅仅支持由 4 位范围整数表示的日期。这就意味着 DATETIME 字段的一个日期必须处于 1735 年 1 月 1 日到 9999 年 12 月 31 日之间。

第二个在 DATETIME 值的整数，即时间整数，存储了午夜后的 1/300-秒单位的数字。这就意味着时间是以毫秒为单位存储的，并精确到 3.33 毫秒。因此，当你在 DATETIME 字段中插入一个值，SQL Server 将把时间转换成.000、.003 或.007 秒。下面的表显示了 SQL Server 是圆整时间值的几个例子：

Time example	Rounded to:
10:10:10.989 10:10:10.990 10:10:10.991	10:10:10.990
10:10:10.992 10:10:10.993 10:10:10.994	10:10:10.993
10:10:10.995 10:10:10.996 10:10:10.997 10:10:10.998	10:10:10.997
10:10:10.999	10:10:11.000

这个表的时间是以小时、分钟、秒钟顺序列出的，并且适当地将秒精确到毫秒。

正如你所看到的，DATETIME 字段中的两个整数一起表示一个特定日期的某一特定时间。当你从一个 DATETIME 字段中检索一个值时，它将日期和时间显示为一系列的数字。比如，下面的 Transact-SQL 语句就是从 DatabaseLog 表中的 PostTime 字段检索数据，这个表也是 AdventureWorks 示例数据库中的一部分：

```
SELECT PostTime FROM dbo.DatabaseLog
WHERE DatabaseLogID = 1
```

PostTime 字段是配置为 DATETIME 数据类型的。当你在 SQL Server Management Studio 中检索值时，所检索到的值在默认情况下以下面的格式返回：

```
PostTime
-----------------------
2005-10-14 01:58:27.567
(1 row(s) affected)
```

注意，日期显示首先是年（2005），接着是月（10），然后是日（14）。然后日期后面是时间，也就是 1 小时、58 分和 27.567 秒。值作为一个整体，它所指的是 2005 年 10 月 14 日，大约是凌晨 1：58 的日期和时间。

（作者：Robert Sheldon 译者：曾少宁／陈柳 来源：TT 中国）

SQL Server 2005 的 DATETIME 和 SMALLDATETIME 基础（二）

SQL Server 的 SMALLDATETIME 数据类型

SMALLDATETIME 数据类型类似于 DATETIEM，只是它的值长度有限。SMALLDATETIME 值是用 2 个 2 位整数存储的。第一个整数表示日期，而第二个整数表示时间。

与 DATETIME 数据类型一样，SMALLDATETIME 数据整数也是相对于基准日期 1900 年 1 月 1 日计算的。但是，整数仅仅表示在基准日期之后的天数，而不包括之前的日期。这就意味着 SMALLDATETIME 字段中的日期必须处于 1900 年 1 月 1 日到 2079 年 6 月 6 日之间。

SMALLDATETIME 值中的第二个整数——时间整数——存储午夜之后的分钟数。时间并不包括秒数，而且，如果值中包括秒，它们是以下面的方式圆整的：

- 小于或等于 29.998 秒的值向下圆整为前一分钟。
- 大于或等于 29.999 秒的值向上圆整为后一分钟。

下表显示了几个 SQL Server 是圆整时间值的例子：

Time example	Rounded to:
14:22:29.996 14:22:29.997 14:22:29.998	14:22
14:22:29.999 14:22:30.000 14:22:30.001	14:23

最初这个表中的时间是按照小时、分钟、秒钟、毫秒的顺序列出的。但是，正如你所看到的，这些时间值都被圆整为时和分，而不包括秒。当你查询 SMALLDATETIME 值时，事实上你是可以看到表示秒的数字的，而且这些数字总是显示为 00，但没有毫秒。比如，下面这个 SELECT 语句使用了 CAST 方法来将 DATETIME 值转换成 SMALLDATETIME 值：

```
SELECT CAST(PostTime AS SMALLDATETIME) AS [Date/Time]
FROM dbo.DatabaseLog
WHERE DatabaseLogID = 1
```

与先前的例子一样，这个语句在 DatabaseLog 表中的 PostTime 字段中查询到数据。只是这一次这个语句返回了稍微有些不同的结果：

```
Date/Time
-----------------------
2005-10-14 01:58:00
(1 row(s) affected)
```

正如你所看到的，时间被向下圆整为 01:58。虽然表示秒的两个数字也被包括在内，但是它们总是显示为 00。（注意，CAST 方法，与 CONVERT 方法相似，都是一个 T-SQL 显式地将一个日期类型转换成另外一个类型的方法。我将在这一系列的后面的文章中继续探讨 CAST 和 CONVERT 方法。你也可以在 Microsoft SQL Server Books Online 上找到关于这两个方法的资料）。

（作者：Robert Sheldon 译者：曾少宁/陈柳 来源：TT 中国）

SQL Server 2005 的 DATETIME 和 SMALLDATETIME 基础（三）

SQL Server 中的 TIMESTAMP 数据类型

另外一个很重要的数据类型是——TIMESTAMP。TIMESTAM 与 DATETIME 和 SMALLDATETIME 是非常不一样的。首先，它基本上与日期或时间无关。但是，它与行版本化却有着千丝万缕的关系。我只在本文中作阐述，是因为它经常与日期/时间数据类型相混淆。

TIMESTAMP 数据类型字段自动地生成二进制值，它表的每一行提供一个版本戳记。当你每插入一个记录行时，就会有一个版本戳记被插入了 TIMESTAMP 字段中。每次你更新行，TIMESTAMP 值也随之更新。

结果，TIMESTAMP 字段是一个确定一个行最近是否被修改过的很方便的方法——当你正在开发一个支持用户并发的应用时，这是一个非常有用的特性。比如，你可以使用 TIMESTAMP 值，通过比较 TIMESTAMP 的原始值与最近值来确定事务是结束还是回滚。如果值是一样的，那么你可以结束事务。否则你必须回滚该事务，因为你很快便知道有另一个用户已经修改了行。

TIMESTAMP 值使用一个数据库计数器，它随着每次行的插入或更新而递增。因为 TIMESTAMP 是一个二进制值，一个包含了 TIMESTAMP 字段的行将自动存储一个类似于下面的值：

```
0x00000000000007DD
```

如果行被更新，那么值也将改变。

当操作 TIMESTAMP 栏时，必须考虑以下几点：

- 这个数据类型是一个递增的数字而且不保存日期/时间数据。
- 这个数据类型不可以用作候选键，比如主键。
- 一个表只能包括一个 TIMESTAMP 字段。
- 这个数据类型的主要目的是支持行版本化。事实上，ROWVERSION 是 TIMESTAMP 数据类型的同义词。

正如你所看到的，TIMESTAMP 字段的范围非常有限。或许除了使用 TIMESTAMP 数据类型来支持行版本化，其它的情况你都应该使用 DATETIME 和 SMALLDATETIME 数据类型，包括当你想要记录一个数据被修改的确切时间。同样的，我在此处提及 TIMESTAMP 数据类型仅仅是为了更好得区分实际的日期/时间数据类型。

后面的文章中，我将进一步探讨 DATETIME 和 SMALLDATETIME 类型值的操作。你将可以学到如何转换日期/时间值、从这些信息中如何获取特定的信息和对值进行差计算。我甚至还将探讨 SQL Server 2008 中的新的日期/时间数据类型。

(作者：Robert Sheldon 译者：曾少宁/陈柳 来源：TT 中国)

SQL Server 中日期/时间值到字符类型的数据转换（一）

在本文中，我将解释如何将 DATETIME 和 SMALLDATETIME 数据类型转换成字符数据，以及如何将字符数据转换成日期/时间数据。具体来说，本章将介绍 Transat-SQL 支持的两个内置 SQL Server 数据转换方法——隐式转换和显式转换。在前一篇的技巧《SQL Server 2005 的 DATETIME 和 SMALLDATETIME 数据基础》中，我已经阐述了 SQL Server 是如何使用 DATETIME 和 SMALLDATETIME 数据类型来存储日期/时间数据的。在本文中，我将解释日期/时间数据是如何转换成字符数据和字符数据是如何转换成日期/时间数据以及 Transact-SQL 是如何支持这两种执行这些数据转换的方法——隐式转换和显式转换。

本文的前提是假定你已具备 T-SQL 和 SQL Server 应用知识，并且该部分只涉及日期/时间数据与字符数据之间的相互转换。但是，你还可以转换其它类型的数值，如将 INT 转换为 DATETIME。虽然在大多数情况下，你的主要工作是字符到日期/时间的转换。

隐式转换数据

当你插入数据到 DATETIME 或 SMALLDATETIME 字段中时，SQL Server 会自动尝试将不同类型的数据进行转换。例如，如果你向 DATETIME 字段中插入 CHAR 值，SQL Server 将对数据作转换——如果该值是一个可以接受的格式。如果你在 CHAR 栏中插入 DATETIME 值，SQL Server 也将作自动转换。

让我们来看看几个隐式转换例子以便更好地理解它是如何工作的。为了说明这些转换，我使用了下面的代码在 AdventureWorks 示例数据库中创建 LogInfo 表：

```
USE AdventureWorks
GO
```

```
CREATE TABLE dbo.LogInfo
(
LogID INT PRIMARY KEY,
LogEvent NVARCHAR(30) NOT NULL,
Post_DateTime DATETIME NOT NULL,
Post_SmallDateTime SMALLDATETIME NOT NULL,
Post_NVarChar NVARCHAR(25) NOT NULL
)
```

在这个表中包含了三个用于保存日期/时间信息的字段：Post_DateTime、Post_SmallDateTime 和 Post_NVarChar。字段的名称反映了用于定义字段的数据类型。下面让我们在这些字段中插入数据：

```
INSERT INTO LogInfo
SELECT DatabaseLogID, [Event],
PostTime, PostTime, PostTime
FROM dbo.DatabaseLog
```

这个语句将从 DatabaseLog 表（在 AdventureWorks 数据库）中获取数据，然后插入到 LogInfo 表中。在源表的 PostTime 字段是 DATETIME 数据类型的。注意，这个字段是用于将数据插入到 LogInfo 表的每个日期/时间字段的。

在你填充好表的数据后，你可以使用下面的 SELECT 语句来获取 LogInfo 表的第一行记录：

```
SELECT * from dbo.LogInfo
```

```
WHERE LogID = 1
```

SELECT 语句返回 LogID 值为 1 的记录行的所有字段的值。下面的结果显示了数据是如何存储在表中的。

LogID	LogEvent	Post_DateTime	Post_SmallDateTime <? xml:namespace prefix = o /><? xml:namespace prefix = o />	Post_NVarChar
1	CREATE_TABLE	2005-10-14 01:58:27.567	2005-10-14 01:58:00	Oct 14 2005 1:58AM

你可以看到，每个日期/时间值都稍微有些不同。The Post_DateTime 字段存储的是完整日期和时间值。但是，正如我们所预期的，Post_SmallDateTime 字段存储一个缩短的时间（00 表示秒）。最后，Post_NVarChar 存储的是一个与其它两个非常不一样的字符串值。

默认情况下，当 SQL Server 将一个 DATETIME 或 SMALLDATETIME 值转换为一个字符值时，它使用上面显示的格式（Oct 14 2005 1:58AM)。在后面的文章中，你将会知道我们还可以将这种格式修改成其它的一些可用的格式。但是目前而言，我们要知道的重要一点是 SQL Server 是如何隐式转换日期/时间值的。现在让我们来看看 SQL Server 中的显式数据转换。

（作者：Robert Sheldon 译者：曾少宁／陈柳 来源：TT中国）

SQL Server 中日期/时间值到字符类型的数据转换（二）

显式转换数据

显式地转换日期/时间值，你必须使用 CAST 或 CONVERT Transact-SQL 方法。由于 CAST 方法是两者中相对简单的，因此我们从这个开始介绍。下面这个 SELECT 语句使用 CAST 方法将 Post_NVarChar 字段中的字符数据转换成一个 DATETIME 值。

```
SELECT LogID,  LogEvent,
CAST(Post_NVarChar AS DATETIME) AS Post_Converted
FROM dbo.LogInfo
WHERE LogID = 1
```

当你使用 CAST 方法时，你必须指定源字段名称（或其它一些表达式）、AS 关键字和值转换的数据类型——这里是 DATETIME。当你运行这个语句时，值就被转换了，如下面显示的结果：

LogID	LogEvent	Post_Converted<? xml:namespace prefix = o /><? xml:namespace prefix = o /><? xml:namespace prefix = o /><? xml:namespace prefix = o />
1	CREATE_TABLE	2005-10-14 01:58:00.000

(1 row(s) affected)

注意，Post_Converted 字段（别名赋给 SELECT 子句中的字段）预期是包括完整日期时间值并精确到毫秒的 DATETIME 格式。但是，秒是表示为 00.000。这是因为当 SQL

Server 转换原始值时，它会去掉秒而只存储小时和分钟值。当你将值转换回 DATETIME 时，SQL Server 将秒设置为 00.000。

然而，如果日期/时间值是以作为字符串存储的并使用 DATETIME 数据所使用的格式，那么 SQL Server 就会保留秒。比如，下面的 SELECT 语句使用 CAST 方法将字符串值转换为 DATETIME：

```
SELECT CAST('2005-10-14 01:58:27.567' AS DATETIME) AS [Date/Time]
```

下面的结果显示秒和毫秒现在被保存了：

Date/Time
2005-10-14 01:58:27.567

(1 row(s) affected)

除了显式地将 DATETIME（或 SMALLDATETIME）值转换成字符数据，你也可以使用 CAST 方法将 DATETIME 数据转换成字符数据。下面的 SELECT 语句使用 CAST 功能从 Post_DateTime 字段中获取数据：

```
SELECT LogID,  LogEvent,
CAST(Post_DateTime AS VARCHAR(20)) AS Post_Converted
FROM dbo.LogInfo
WHERE LogID = 1
```

正如你所看到的下面显示的结果，句法将值转换成 VARCHAR：

LogID	LogEvent	Post_Converted
1	CREATE_TABLE	Oct 14 2005 1:58AM

(1 row(s) affected)

注意，当 SQL Server 隐式地将 DATETIMER 值转换成 NVARCHA 时，转换的值的格式现在就是你先前看到格式。

现在你了解了如何使用 CAST 方法，那么让我们接着看看 CONVERT 方法。最基本的，CONVERT 方法返回与 CAST 方法一样的结果。比如，与上面的例子一样，下面的语句将 Post_DateTime 值转换成 VARCHAR：

```
SELECT LogID,  LogEvent,
CONVERT(VARCHAR(20),  Post_DateTime) AS Post_Converted
FROM dbo.LogInfo
WHERE LogID = 1
```

然而，注意在 CONVERT 方法中的参数的顺序与 CAST 方法的是不一样的。当使用 CONVERT 时，你首先指定目标数据类型（VARCHAR），然后是源字段(Post_DateTime)的名称，它是由逗号分隔的两个参数，而不是 AS 关键字。当你运行语句时，你会得到下面的结果：

LogID	LogEvent	Post_Converted
1	CREATE_TABLE	Oct 14 2005 1:58AM

(1 row(s) affected)

结果跟前面的例子是一样的。但是，如果你想要以一定的格式显示你的日期/时间值，而不是使用目前我们所看到的格式(Oct 14 2005 1:58AM)。这时，你可以在 CONVERT 方法中添加第三个参数来指定新的格式，如下面的例子所显示的：

```
SELECT LogID,  LogEvent,
CONVERT(VARCHAR(20),  Post_DateTime,  101) AS Post_Converted
FROM dbo.LogInfo
WHERE LogID = 1
```

注意，101 已经作为第三个参数添加到方法中。当指定一个格式时，你必须使用由 T-SQL 支持的预定义代码来表示你想要使用的格式。在这种情况下，101 返回如下所显示格式的结果：

LogID	LogEvent	Post_Converted
1	CREATE_TABLE	10/14/2005

(1 row(s) affected)

Post_Converted 值现在的格式是 10/14/2005，这个也是代码 101 代表的格式。如果你想要你的结果显示为类似于 DATETIME 值所显示的格式，那么你可以指定代码 121，如下面的例子：

```
SELECT LogID,  LogEvent,
CONVERT(VARCHAR(25),  Post_DateTime,  121) AS Post_Converted
FROM dbo.LogInfo
WHERE LogID = 1
```

现在返回的结果是完整日期和时间值，精确到毫秒：

LogID	LogEvent	Post_Converted
1	CREATE_TABLE	2005-10-14 01:58:27.567

(1 row(s) affected)

T-SQL 支持多种预定义的格式。关于用于调用每个格式的格式命名和代码的完整清单，你可以在 Microsoft SQL Server Books Online 中阅读 CAST 和 CONVERT（Transact-SQL）专题。

现在让我们来看一个不同的例子。在下面的 SELECT 语句中，我们使用了 CONVERT 方法将 Post_SmallDateTime 字段栏转换成一个 VARCHAR 字段：

```
   SELECT LogID,  LogEvent,
CONVERT(VARCHAR(25),  Post_SmallDateTime,  121) AS Post_Converted
FROM dbo.LogInfo
WHERE LogID = 1
```

正如前面的例子，日期/时间值显示为 121 格式：

LogID	LogEvent	Post_Converted
1	CREATE_TABLE	2005-10-14 01:58:00.000

(1 row(s) affected)

注意，由于日期/时间值是从 SMALLADATETIME 字段中获取的，因此时间值中的秒是 00.000，这与 SMALLDATETIME 的是一样的。以下是如何以指定更短的长度截断 VARCHAR 数据类型的秒：

```
SELECT LogID,  LogEvent,
CONVERT(VARCHAR(16),  Post_SmallDateTime,  121) AS Post_Converted
FROM dbo.LogInfo
WHERE LogID = 1
```

目前 CONVERT 功能的数据类型参数显示为 VARCHAR（16）而非 VARCHAR（25），与前面的例子一样。下面的结果显示值是如何被截断以便秒不再显示：

LogID	LogEvent	Post_Converted
1	CREATE_TABLE	2005-10-14 01:58

(1 row(s) affected)

这就是所有关于日期/时间值的显式转换方法。当获取这些值时，CAST 和 CONVERT 方法都是方便的工具（注意，这些方法同样可以用于转换其它类型的值）。在接下来的文章中，我将阐述如何从日期/时间字段获取特定的信息，以及如何计算日期/时间值。同时，你现在也已经掌握了如何转换这些值以及以特定格式显示它们的基本用法，这对你是非常有用的。

（作者：Robert Sheldon 译者：曾少宁/陈柳 来源：TT 中国）

正确使用 SQL Server 的 datetime 函数：GETDATE、DATENAME 和 DATEPART（一）

Transact-SQL 包含一组函数可以用于检索当前的日期和时间或一个 DATETIME 或 SMALLDATETIME 值的某个部分。比如，你可以在 DATETIME 值中提取日、月或年以及季度、周、小时甚至毫秒。在本文中，我将对这些函数进行阐述并举例说明如何使用 SQL Server 中的这些函数来查询数据的。注意，本文假定你已经具备了一定的 T-SQL、DATETIME 和 SMALLDATETIME 数据类型的知识。想获得更多这些类型的信息，你可以阅读这一系列的第一部分“SQL Server 2005 的 DATETIME 和 SMALLDATETIME 基础”。

检索当前日期和时间

在 T-SQL 中最便捷的一个函数就是 GETDATE，它根据本地系统的时钟设置检索当前日期和时间。使用 GETDATE，只需直接调用 T-SQL 语句中的函数而不用指定任何参数，如下例所示：

```
SELECT GETDATE() AS [Current Date/Time]
```

在这里，我在 SELECT 中使用 GETDATE 来检索日期/时间值。（注意，即使你不需要任何参数，你必须使用括号。）语句返回的结果如下：

Current Date/Time
2008-07-29 10:45:13.327

默认情况下，GETDATE 函数返回如下格式的时间值。然而，你可以使用 CONVERT 函数修改结果的格式。关于 CONVERT 的使用，可以参考这一技巧系列“从 date/time 值到字符类型的数据转换”。

Transact-SQL 的另外一个简单的函数是 GETUTCDATE，它用于检索当前的 Coordinated Universal Time (UTC)——也就是格林威治标准时间。检索的值是基于本地系统上的时钟和时区设置的。正如你在 GETDATE 中所看到的，你在 Transact-SQL 语句中调用 GETUTCDATE 是不需要包括任何参数的，如下例所示：

```
SELECT GETUTCDATE() AS [UTC Date/Time]
```

你运行这个语句时，你会得到如下的结果：

UTC Date/Time
2008-07-29 17:45:13.327

注意，这里返回的时间比前面显示的例子要晚 7 个小时。但我在一个配置为太平洋时区（在夏时制的白天）的系统上同时运行这些语句。

正如你在最后两个例子中所看到的，这些函数都是包含在 SELECT 列表中的。但是，当用它们来定义你定义表的默认值时，这些函数是特别有用的。比如，下面的三个语句所创建的 Orders 表格——包括一个 DATETIME 字段（OrderDate）——在表格中插入数据并检索该数据：

```
CREATE TABLE Orders
(
OrderID INT PRIMARY KEY IDENTITY,
```

```
Product VARCHAR(30) NOT NULL,
OrderAmt INT NOT NULL,
OrderDate DATETIME NOT NULL DEFAULT GETDATE()
)
GO
INSERT INTO Orders (Product, OrderAmt)
VALUES('Test Product', 12)
GO
SELECT * FROM Orders
```

OrderDate 字段定义包含一个指定 GETDATE 作为默认值的 DEFAULT 子句。因此，当你在表格中插入行时，当前日期和时间将自动插入列中，下面显示了 SELECT 语句返回的结果：

OrderID	Product	OrderAmt	OrderDate
1	Test Product	12	2008-07-29 10:46:47.420

你可以将这些信息作为一个时间戳来使用，以便在需要时跟踪加入的记录或协助数据审计。这也方便其它使用时间戳检索数据的操作。比如，当决定是否提取或更新数据时，抽取、转换和加载（ETL）程序可能参考时间戳。

检索年、月或日

在某些情况下，你可能想要在 DATETIME 或 SMALLDATETIME 值中检索年、月或日。其中一个函数是使用 YEAR、MONTH 或 DAY 函数来检索必要的数据（作为一个整数）。下面的 SELECT 语句就是一个很好的说明：

```
SELECT YEAR(PostTime) AS [Year],
```

```
MONTH(PostTime) AS [Month],
DAY(PostTime) AS [Day]
FROM DatabaseLog
WHERE DatabaseLogID = 1
```

SELECT 子句中包含了三个字段表达式。第一个使用的是 YEAR 函数来检索 DatabaseLog 表格（AdventureWorks 中的样本数据库）中的 PostTime 字段的年。当调用 YEAR 函数时，指定字段的名称（或其它表达式）作为函数的参数。MONTH 和 DAY 函数也是一样的运行方式。在 SELECT 子句中的第二个字段表达式使用 DAY 来检索日。下面的结果显示了语句返回的信息类型：

每一个值都是在 PostTime 字段中提取并作为一个整数返回的。（存储在表中的值是 2005-10-14 01:58:27.567。）

这些函数都是检索年、月或日的简单方式，但是，在某些情况下，你可能想更多的控制返回的值的类型以及这些值的格式。另外，你可能想从日期/时间值中提取时间。幸运的是，Transact-SQL 支持这些函数。

（作者：Robert Sheldon 译者：曾少宁/陈柳　来源：TT 中国）

正确使用 SQL Server 的 datetime 函数：GETDATE、DATENAME 和 DATEPART（二）

检索 date/time 值各部分

与 YEAR、MONTH 和 DAY 函数相似，DATEPART 函数返回一个代表 date/time 值的指定部分的整数值。比如，下面的 SELECT 语句返回与上面的例子一样的结果：

```
SELECT DATEPART(yy,  PostTime) AS [Year],
DATEPART(mm,  PostTime) AS [Month],
DATEPART(dd,  PostTime) AS [Day]
FROM DatabaseLog
WHERE DatabaseLogID = 1
```

首先要注意的是，当调用 DATEPART 时，需要指定两个参数。第一个参数确认检索的 date/time 值的组成部分，而第二个参数是一个源字段。对于第一个参数，你必须使用一个支持的缩写来指定时间部分。下面的表格列举了你可以检索的日期/时间部分以及你必须用来检索这些部分的缩写：

Date/time part	Abbreviations
year	yy, yyyy
quarter	qq, q
month	mm, m
day of year	dy, y
day	dd, d

week	wk， ww
weekday	dw
hour	hh
minute	mi， n
second	ss， s
millisecond	ms

有些 datetime 部分有多个缩写支持。比如，你可以使用“YY”或“YYY”作为第一个 DATEPART 参数来检索 date/time 值中的年。注意，表格不仅仅包含年、月或日的缩写。换言之，你还可以检索季度、一年中特定的一天、一年中特定的一周以及工作日，如下面的 SELECT 语句所示：

```
SELECT DATEPART(qq， PostTime) AS [Quarter]，
DATEPART(dy， PostTime) AS [DayOfYear]，
DATEPART(wk， PostTime) AS [Week]，
DATEPART(dw， PostTime) AS [Weekday]
FROM DatabaseLog
WHERE DatabaseLogID = 1
```

正如前面的例子所显示的，每一个 DATEPART 实例都包含两个参数：date/time 部分的缩写和源字段。语句返回如下的结果：

Quarter	DayOfYear	Week	Weekday
4	287	42	6

注意，工作日显示为 6。默认情况下，SQL Server 的每周是从星期天开始的，因此，工作日 6 等同于星期五。

上面的两个例子仅仅检索与日期相关的值。但是，正如下面的表格所显示的，你也可以检索其它与时间相关的数据：

```
SELECT DATEPART(hh,  PostTime) AS [Hour],
DATEPART(mi,  PostTime) AS [Minute],
DATEPART(ss,  PostTime) AS [Second],
DATEPART(ms,  PostTime) AS [Millisecond]
FROM DatabaseLog
WHERE DatabaseLogID = 1
```

在这种情况下，该语句是检索小时、分、秒和毫秒，结果显示如下：

DATEPART 函数的主要的局限是它返回的只是整数，这就是为什么星期五是以 6 显示的。然而，如果你想要显示实际的日和月名称，你可以使用 DATENAME 函数。DATENAME 函数与 DATEPART 函数的运作完全一样。DATENAME 使用同样数目的参数并支持同样的缩写。比如，如果你检索年、月和日，那么你只需简单地用 DATENAME 取代 DATEPART 就可以得到正如你在前面的例子中所看到的结果：

```
SELECT DATENAME(yy,  PostTime) AS [Year],
DATENAME(mm,  PostTime) AS [Month],
DATENAME(dd,  PostTime) AS [Day]
FROM DatabaseLog
WHERE DatabaseLogID = 1
```

这样，你将得到下面的结果：

Year	Month	Day

2005	October	14

月的值现在是 October 而不是 10。但是，由于只有一种方式来代表它们，因此，年和日依然是整数。你也可以在其它的日期/时间组件上使用 DATENAME 函数，如下所示：

```
SELECT DATENAME(qq,  PostTime) AS [Quarter],
DATENAME(dy,  PostTime) AS [DayOfYear],
DATENAME(wk,  PostTime) AS [Week],
DATENAME(dw,  PostTime) AS [Weekday]
FROM DatabaseLog
WHERE DatabaseLogID = 1
```

再次，我用 DATENAME 取代 DATEPART，但是保留其它的不改变。语句返回了以下的结果。

Quarter	DayOfYear	Week	Weekday
4	287	42	Friday

注意，季度、一年中第几天和第几周仍然是整数，但是工作日现在是 Friday 而不是 6。你也可以使用 DATENAME 来检索一个 date/time 值的时间组件，但是结果将如你所想的那样，仍然是整数。

这就是 DATEPART 和 DATENAME 函数以及其它用来检索日期/时间值的函数。你可以单独或者混合着使用这些函数来关联值。我建议你尝试着使用这些函数以便更好地了解它们是如何工作的。在以后的技巧中（第四部分），我将阐述如何在这些值上执行运算以便添加数据和确定日期范围。同时，你也可以在 Microsoft SQL Server Books Online 上面获得各个函数的更多的信息和例子。

（作者：Robert Sheldon 译者：曾少宁/陈柳 来源：TT 中国）

使用 DATEADD 和 DATEDIFF 来计算 SQL Server 的 DATETIME 值（一）

在 SQL Server 数据库中，DATETIME 和 SMALLDATETIME 值是以整数存储的。然而，与整数不同的是，它们不能直接地进行数学运算。尽管如此，有时候还是需要在日期/时间值中添加或减去一个时间间隔。比如，你可能想在一值上加一些月数或天数，或者甚至可能是小时数。你甚至可能想比较两个日期/时间值以便确定它们之间的时间间隔，如相差的天数或年数。为了简化这些类型的计算，Transact-SQL 支持两个重要的日期/时间方法：DATEADD 和 DATEDIFF。

在关于 DATETIME 值这一系列文章的第四部分，我阐述了如何使用这两个方法并举例说明它们是如何工作的。为了演示这些方法，我使用了下面的 Transact-SQL 代码在 AdventureWorks 示例数据库中创建了一个 Sales.Orders 表：

```
USE AdventureWorks
GO
IF EXISTS (SELECT table_name
FROM information_schema.tables
WHERE table_schema = 'Sales'
AND table_name = 'Orders')
DROP TABLE Sales.Orders
GO
CREATE TABLE Sales.Orders
(
OrderID INT NOT NULL,
OrderDate DATETIME NOT NULL,
```

```
DelivDate DATETIME NOT NULL
)
GO
INSERT INTO Sales.Orders
VALUES(1001, GETDATE(), '2008-09-08 18:27:10.750')
```

表的定义包含了 OrderDate 和 DelivDate 字段，两者都是 DATETIME 数据类型。在我创建了表之后，我在表中插入了一行用于测试 DATEADD 和 DATEDIFF 方法的数据。

使用 DATEADD 方法

在一些情况下，你可能想添加一个时间间隔到 DATETIME 或 SMADDDATETIME 值中——或者减去一个时间间隔。比如，你可能需要在一个指定的日期中增加或减去一个月。你可以使用 DATEADD 方法来执行这个计算。这个方法运用了下面的语法：

```
DATEADD(<date/time_part>, <number>, <date>)
```

<date/time_part>占位符指的是日期/时间值中增加或减少的增量/余差（如日或月）。下表列出了可以使用的日期/时间部分，以及代表这些部分的缩写：

Date/time part	Abbreviations
year	yy, yyyy
quarter	qq, q
month	mm, m
day of year	dy, y
day	dd, d
week	wk, ww

weekday	dw
hour	hh
minute	mi， n
second	ss， s
millisecond	ms

比如，如果你想在一个日期/时间值中增加一小时，可以使用 hh 缩写。在某些情况下，日期/时间部分支持两个缩写，如周可以用 wk 或 ww 支持。

<number>占位符指的是所增加的数值（一个整数）。比如，如果在日期中增加 10 天，就是 10。但是，注意，如果是减去时间间隔，它必须是一个负整数。比如，从天数中减去 10，就必须是-10。

<date>占位符指的是增加或减少的指定间隔的日期/时间值。它可能是一个日期/时间格式的字符串值，或者是方法返回的一个日期/时间值，又或者是常见的 DATETIME 或 SMALLDATETIME 字段。

让我们举例来说明它是如何工作的。在下面的 SELECT 语句中，我增加三个月到 Sales.Orders 表中 OrderDate 值：

```
SELECT OrderDate， DATEADD(mm， 3， OrderDate) AS NewDate
FROM Sales.Orders
WHERE OrderID = 1001
```

注意，SELECT 列表使用了 DATEADD 方法。这个方法有三个参数：mm 指月，3 指月数，而 OrderDate 是一个 DATETIME 值。因此，当查询返回值时，每个 OrderDate 值都会增加三个月时间，如下的结果所示：

OrderDate	NewDate
2008-08-27 13:36:16.280	2008-11-27 13:36:16.280

如上所示，日期 August 27 已经被改为 November 27。而且，这样的运算还不仅限于日期。下面我在 OrderDate 值中增加三个小时：

```
SELECT OrderDate,  DATEADD(hh,  3,  OrderDate) AS NewTime
FROM Sales.Orders
WHERE OrderID = 1001
```

DATEADD 的第一个参数现在是 hh，而不是 mm，因此，只有小时被改变了，如下结果所示：

OrderDate	NewTime
2008-08-27 13:36:16.280	2008-08-27 16:36:16.280

日期/时间值也可以减去一定的日期或时间间隔。在下例中，我从 OrderDate 值中减去了三天：

```
SELECT OrderDate,  DATEADD(dd,  -3,  OrderDate) AS PastDate
FROM Sales.Orders
WHERE OrderID = 1001
```

注意，DATEADD 的第一个参数现在是 dd。同时，注意，第二个参数是一个负数，这意味着将有三天被减去，如下所示：

OrderDate	PastDate

2008-08-27 13:36:16.280	2008-08-24 13:36:16.280

这样，新的日期是 August 24 而不是 August 27。

这样，上面的例子演示如何在从数据库查询到日期/时间值后再对它进行修改。而 DATEADD 方法同样也可以用来插入日期/时间数据。因为 DATEADD 方法返回一个 DATETIME 值。（如果所提供的日期对应的方法是 SMALLDATETIME，那么它将返回一个 SMALLDATETIME 值。）在下面的例子中，我添加了一行数据到 Sales.Orders 表中，然后使用 SELECT 语句来检索这个行：

```
INSERT INTO Sales.Orders
VALUES(1002, GETDATE(), DATEADD(dd, 10, GETDATE()))
GO
SELECT * FROM Sales.Orders
WHERE OrderID = 1002
```

注意，VALUES 子句包含了表中每个字段的值。对于 OrderDate 值，我使用 GETDATE() 方法来获取当前的日期和时间。对于 DelivDate 字段，我使用 DATEADD 方法以及相应的三个参数。第一个参数 dd 表示将要添加到日期中的是天数。第二个参数 10 意味着将添加 10 天到日期中。最后，第三个参数是 GETDATE 方法。因此，10 天将添加到目前的日期和时间中并插入到 DelivDate 字段。这就是 SELECT 语句生成的结果：

OrderID	OrderDate	DelivDate
1002	2008-08-27 13:40:22.357	2008-09-06 13:40:22.357

正如所期待的，DelivDate 值比 OrderDate 晚 10 天。

现在让我们来检测一个使用了 DATEADD 方法的 UPDATE 语句。在下面的语句中，我从 DelivDate 值中减去了三天，然后显示了结果：

```
UPDATE Sales.Orders
SET DelivDate = DATEADD(dd,  -3,  DelivDate)
WHERE OrderID = 1002
GO
SELECT * FROM Sales.Orders
WHERE OrderID = 1002
```

这次我在 SET 子句中使用了 DATEADD——我将 DelivDate 值设为 DATEADD 方法返回的结果。这个方法指定天数（dd）为第一个参数，-3 为第二个参数，而 DelivDate 字段为第三个参数。这就意味着该方法将返回一个比原始日期早三天的日期，并将 DelivDate 设置为新的日期，如下结果显示：

OrderID	OrderDate	DelivDate
1002	2008-08-27 13:40:22.357	2008-09-03 13:40:22.357

你应该记得，INSERT 语句（在前一个例子）添加了一个 DelivDate 值为 September 6 的行。但是，这个值现在是 September 3，比原来早了三天。

（作者：Robert Sheldon 译者：曾少宁/陈柳 来源：TT 中国）

使用 DATEADD 和 DATEDIFF 来计算 SQL Server 的 DATETIME 值（二）

使用 DATEDIFF 方法

DATEDIFF 方法可以计算两个日期之间的时间间隔，并返回一个代表间隔的整数。这个方法使用下面的语法：

```
DATEDIFF(<date/time_part>, <start_date>, <end_date>)
```

<date/time_part>占位符指的是两个日期中需要比较的部分。比如，你想确认开始日期和结束日期之间的小时数或天数。

除了工作日(dw, w)缩写之外，<date/time_part>占位符使用的缩写与 DATEADD 方法一样。DATEDIFF 不支持工作日比较。

<start_date>占位符指的是比较的开始日期，而<end_date>占位符指的是结束日期。换言之，方法将返回开始日期和结束日期之间的具体时间或日期间隔。

让我们举例来说明它是如何工作的。下面的 SELECT 语句计算了 Sales.Orders 表中 OrderDate 和 DelivDate 值之间的时间间隔：

```
SELECT OrderDate, DelivDate,
DATEDIFF(dd, OrderDate, DelivDate) AS DaysDiff
FROM Sales.Orders
WHERE OrderID = 1002
```

在这个语句中，我在 SELECT 列表中使用了 DATEDIFF。这个方法的第一参数指定间隔必须是天数（dd），而第二个参数指定 OrderDate 作为开始日期，然后第三个参数指定 DelivDate 作为结束日期。因此，DATEDIFF 将计算 OrderDate 和 DelivDate 之间的天数，在此例子中，它是 7 天，如下结果所示：

OrderDate	DelivDate	DaysDiff
2008-08-27 13:40:22.357	2008-09-03 13:40:22.357	7

当然，它也可以用来计算各种时间间隔，如下例的语句所示：

```
SELECT OrderDate,  DelivDate,
DATEDIFF(hh,  OrderDate,  DelivDate) AS HoursDiff
FROM Sales.Orders
WHERE OrderID = 1002
```

在这种情况下，方法的第一个参数是小时（hh）而非天数。因此，方法将返回 OrderDate 和 DelivDate 值之间相差的小时数，如下结果所示：

OrderDate	DelivDate	HoursDiff
2008-08-27 13:40:22.357	2008-09-03 13:40:22.357	168

两个值之间相差 168 个小时。

与 DATEADD 方法一样，DATEDIFF 方法并不仅限于用在 SELECT 语句中。比如，DATEDIFF 可以用在 UPDATE 语句的 WHERE 子句中，以确定哪一行需要更新。在下例中，我使用了 DATEDIFF 来指定这些在 OrderDate 和 DelivDate 值之间天数少于 8 的行。

```

```

```
UPDATE Sales.Orders
SET DelivDate = DATEADD(dd,  3,  DelivDate)
WHERE DATEDIFF(dd,  OrderDate,  DelivDate) < 8
GO
SELECT OrderID,  OrderDate,  DelivDate,
DATEDIFF(dd,  OrderDate,  DelivDate) AS DaysDiff
FROM Sales.Orders
```

在前面的例子中，DATEDIFF 方法返回了 OrderDate 和 DelivDate 值之间的天数。然后这个数目将与 8 作比较。如果天数少于 8，那么这一行将被更新；否则，该行将不改变。对于这些需要更新的行，我使用 DATEADD 方法来增加三天到 DelivDate 值中。然后，我运行一个 SELECT 语句来返回 Sales.Orders 表的数据以及计算每个行中两个日期的不同，如下结果所示：

OrderID	OrderDate	DelivDate	DaysDiff
1001	2008-08-27 13:36:16.280	2008-09-08 18:27:10.750	12
1002	2008-08-27 13:40:22.357	2008-09-06 13:40:22.357	10

结果显示现在在两个日期（第二行中）之间是相差 10 天，而非原来的 7 天。

在表定义中使用 DATEADD 和 DATEDIFF

DATEADD 和 DATEDIFF 方法也可以用在表定义中。例如，字段定义的 DEFAULT 子句中可以用 DATEADD 方法或使用 DATEDIFF 方法来创建一个计算得来的字段。在下面的 Transact-SQL 代码中，我首先创建了使用 DATEADD 和 DATEDIFF 的表，然后添加一行数据到表中，最后检索表的数据：

```
USE AdventureWorks
```

```
GO
IF EXISTS (SELECT table_name
FROM information_schema.tables
WHERE table_schema = 'Sales'
AND table_name = 'Orders')
DROP TABLE Sales.Orders
GO
CREATE TABLE Sales.Orders
(
OrderID INT NOT NULL,
OrderDate DATETIME NOT NULL DEFAULT GETDATE(),
DelivDate DATETIME NOT NULL DEFAULT DATEADD(dd, 10, GETDATE()),
DaysDiff AS DATEDIFF(dd, OrderDate, DelivDate)
)
GO
INSERT INTO Sales.Orders(OrderID)
VALUES(1001)
GO
SELECT OrderID, OrderDate, DelivDate, DaysDiff
FROM Sales.Orders
```

在 CREATE TABLE 语句中，我创建了四个字段，其中三个存储日期/时间数据。OrderDate 字段直接使用 GETDATE 来生成默认值。DelivDate 字段也有一个默认值。然而，这个默认值是基于 DATEADD 返回的结果的，同时，在这种情况下，我使用方法增加 10 天到 GETDATE 返回的值存储到 DelivDate 字段中。最后，DaysDiff 字段是一个计算得来的字段，它的值是使用 DATADIFF 来计算 OrderDate 和 DelivDate 值之间的天数差。

在表定义之后，我插入一数据行到表中。因为所有的日期/时间值都是自动生成的，因此我仅仅需要插入 OrderID 值，如下所示：

OrderID	OrderDate	DelivDate	DaysDiff
1001	2008-08-27 13:42:50.433	2008-09-06 13:42:50.433	10

DATEADD 和 DATEDIFF 方法不仅仅在表定义中非常有用，同样也适用于查询和数据修改语句。通过 DATEADD，我们可以将日期/时间值增加和减少一定值，而通过 DATEDIFF，我们可以计算日期/时间值之间的时间间隔。更多详细的关于这些方法的信息，可以阅读 Microsoft SQL Server Books Online。

（作者：Robert Sheldon 译者：曾少宁/陈柳 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：Robert Sheldon 译者：曾少宁/陈柳 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：Robert Sheldon 译者：曾少宁/陈柳 来源：TT 中国)