Le type Datetime de SQL-Server

par Baptiste Wicht (home) Frédéric Brouard (home)

Date de publication : Le 4 Mars 2007

Avec cet article, vous allez découvrir comment fonctionne véritablement le type datetime de SQL-Server, son utilisation et ses mécanismes internes.

I - Introduction

DATETIME est le type SQL-Server pour stocker des valeurs composées d'une date et d'une heure (horodatage). Il correspond au type TIMESTAMP de la norme SQL. Disons-le tout de suite, SQL Server offre un type TIMESTAMP qui n'a rien à voir avec la norme SQL et constitue un simple marqueur de version de ligne. Ce type TIMESTAMP version SQL Server a d'ailleurs été rebaptisé ROWVERSION depuis la version 2000 de SQL Server. Le type DATETIME de SQL Server est donc destiné au stockage d'un combiné DATE + TEMPS dont la précision permet des expressions comme : 21/12/2006 22:16:43.666.
SQL Server ne dispose pas des types DATE et TIME séparés prévus par la norme. Pour l'anecdote, ces deux types étaient prévus dans la version 2005 de SQL Server. Ils ont même été montrés dans une version bêta à disposition de certains professionnels, puis rapidement retirés du fait de la mauvaise performance d'implémentation de ces deux nouveaux types.

Avant de s'aventurer à détailler ce type, commençons par un petit mystère avec l'utilisation du DATETIME :

```
CREATE TABLE T_TEST_DATE_TDT (
 TDT_ID       INT,
 TDT_DATE     DATETIME
)

INSERT INTO T_TEST_DATE_TDT VALUES (1, '20061223 23:59:59.99')
INSERT INTO T_TEST_DATE_TDT VALUES (2, '20061223 23:59:59.999')
INSERT INTO T_TEST_DATE_TDT VALUES (3, '20061224')
INSERT INTO T_TEST_DATE_TDT VALUES (4, '20061224 23:59:59')
INSERT INTO T_TEST_DATE_TDT VALUES (5, '20061224 23:59:59.9')
INSERT INTO T_TEST_DATE_TDT VALUES (6, '20061224 23:59:59.99')
INSERT INTO T_TEST_DATE_TDT VALUES (7, '20061224 23:59:59.999')
INSERT INTO T_TEST_DATE_TDT VALUES (8, '20061225')

-- selectionner les lignes pour la journée du 24 décembre...
SELECT *
FROM   T_TEST_DATE_TDT
WHERE  TDT_DATE BETWEEN '20061224 00:00:00.000'
                    AND '20061224 23:59:59.999'
```

Logiquement, à première vue, cette requête devrait nous renvoyer les lignes 3 à 7, mais ce n'est pas le cas, puisqu'elle nous retourne les lignes 2 à 8. Pourquoi ? C'est ce que vous allez comprendre à la lecture de cet article...

II - Stockage interne

Le type DATETIME est représenté en interne par deux entiers de 4 octets. Le premier nombre représente le nombre de jours depuis le 1er Janvier 1900 et le deuxième nombre équivaut au nombre de millisecondes écoulées depuis zéro heure.

Le type DATETIME permet donc de représenter une date du 1 Janvier 1753 à zéro heure jusqu'au 31 Décembre 9999 à 23:59:59 et 997 millisecondes.
Pourquoi avoir limité la date la plus basse à 1753 ? Ce n'est bien entendu pas par hasard.
Mais ce second mystère nécessite de nous plonger un peu dans l'Histoire...
En effet au cours du temps et compte tenu de la précision accrue des calculs astronomiques, on constate que le calendrier julien élaboré en 46 avant JC souffre d'un manque de précision : la durée de l'année julienne est trop longue d'un peu plus de 11 minutes (365,25 jours au lieu de 365,2422) et au fil du temps, le calendrier finit par retarder de plusieurs jours. C'est le pape Grégoire XIII qui entreprend de réformer le calendrier : Pour cela, 10 jours furent supprimés de l'année 1582, où le 4 octobre fut immédiatement suivi par le 15 octobre. Pour éviter de nouvelles dérives, la sur évaluation de l'année julienne fut corrigée par la suppression de 3 jours tous les 400 ans. On ignore donc la règle des années bissextiles les années séculaires, sauf pour celles qui sont divisibles par 400.
Il y a donc 97 années bissextiles par période de 400 ans et la durée moyenne d'une année grégorienne est 365 + 97/100, c'est-à-dire 365,2425 jours.
Ainsi naquit le calendrier grégorien toujours en vigueur de nos jours.

Cette réforme nécessaire n'a pas été adoptée à la même date dans tous les pays :

- L'Italie, Espagne, Portugal et Pologne sont passés du 4 octobre 1582 au 15 octobre 1582
- La France (sauf Alsace et Lorraine) est passée du 9 décembre 1582 au 20 décembre 1582
- Le Luxembourg est passé du 14 décembre 1582 au 25 décembre 1582
- La Belgique (alors province des Pays-Bas) est passée du 21 décembre 1582 au 1 janvier 1583
- Le Valais Suisse est passé du 28 février 1655 au 11 mars 1655
- L'Alsace est passée du 4 février 1682 au 16 février 1682
- Zurich, Berne, Bâle et Genève sont passés du 31 décembre 1700 au 12 janvier 1701
- L'Angleterre est passée du 2 septembre 1752 au 14 septembre 1752
- La Lorraine est passée du 16 février 1760 au 28 février 1760
- L'U.R.S.S est passée du 31 janvier 1917 au 14 février 1917

Voilà pourquoi dans les pays anglo-saxons, (les USA ayant adopté la culture anglaise) les dates antérieures à 1753 sont difficiles à exprimer sans un changement de calendrier !

III - Précision

Vous allez maintenant comprendre le petit mystère du chapitre I. La limitation de l'entier de 4 octets ne permettait pas des calculs justes et précis à une milliseconde près. Il a été donc choisi de limiter la précision à 3 millisecondes afin d'obtenir de bonnes performances de traitement. Mais cette imprécision relative entraîne quelques effets de bord dont celui présenté au début de cet article. Ainsi la limite de temps d'une journée dans SQL Server n'est pas située à 23h 59m 59s et 999ms car nous serions déjà au lendemain du fait de l'imprécision, mais plus exactement à 23h 59m 59s et 997ms dernière valeur de temps exprimable directement dans un type DATETIME de SQL Server...

IV - Conversion String vers DATETIME

Comment exprimer une date sous forme littérale afin qu'elle soit prise en compte de manière adéquate en tant que DATETIME ?
Autrement dit quel "format" d'encodage faut-il utiliser pour considérer qu'une chaîne de caractères représente une variable DATETIME ?
Tout le problème réside bien évidemment dans l'ordre des éléments : année, mois jour...
SQL Server permet différentes approches :

1) la sensibilité aux spécificités régionales

Sur un serveur SQL Français installé sur un OS Windows Français, utilisé par un poste client doté d'un OS Français, la correspondance date littérale vers DATETIME obéit aux principes de la langue. Autrement dit vous pouvez passer une date au format JJ/MM/AAAA ou encore JJ-MM-AAAA...
Mais qu'en est-il avec un serveur doté d'un OS anglais, d'un SQL français, utilisé par un Japonais ?

2) le forçage d'un format

Le flag de session DATEFORMAT permet de forcer l'ordre des parties de la date dans les correspondances littérales vers DATETIME. Il s'utilise comme suit :

```
SET DATEFORMAT MYD

SELECT CAST('03/1987/21' AS DATETIME)
```

Dans cet exemple on permet d'utiliser un format de date avec le mois en premier suivi de l'année et se terminant par le jour, d'où le paramètre MYD. Bien entendu on peut utiliser MDY, YMD, YDM, DYM, DMY...
Comme tout paramètre de session, DATEFORMAT est valable tant que vous ne modifiez pas son état et que la connexion existe. Aussi est-ce une bonne chose que de commencer toute session dans une application (exécutable comme ASP par exemple) par définir la valeur de ce paramètre.

3) le format universel

Mais il existe un format très particulier. Le format ISO court d'encodage de date passe outre le flag DATEFORMAT et les paramètres régionaux. Quel est ce format ? Tout simplement AAAAMMJJ sans aucun séparateur d'aucune sorte...

Dernière petite remarque : si vous utilisez une année à deux chiffres, alors l'année pivot par défaut est 50. En dessous de 50 vous êtes au 21e siècle et vos années commencent par 20, à 50 et au dessus vous être au 20e siècle et vos années sont complétées par 19. Exemple :

```
SELECT CAST('21/04/49' AS DATETIME), CAST('21/04/50' AS DATETIME)
```

Vous pouvez d'ailleurs régler ce paramètre au niveau du serveur :

```
sp_configure 'two digit year cutoff', '2000'
```

V - Stocker seulement une date ou du temps

SQL Server ne permet pas de stocker seulement une date ou un temps, il ne fournit que le type DATETIME qui permet de stocker les deux. Néanmoins, vous pouvez utiliser indifféremment un DATETIME pour une date avec heure à zéro, un INTEGER pour une date sans heure, un FLOAT ou un DECIMAL pour ne stocker que le temps...Exemple :

```
SELECT CAST('1900-01-04' AS DATETIME), CAST('10:00' AS DATETIME)
```

La partie qui n'est pas renseignée sera tout simplement égale à zéro, c'est-à-dire pour la date 1/1/1900. Ne vous préoccupez donc que de la seule partie qui vous intéresse.

Les formats horaires reconnus par SQL Server sont les suivants :

- 14:30
- 14:30[:20:999]
- 14:30[:20.9]
- 4am
- 4 PM
- [0]4[:30:20:500]AM

Pour la date courante vous ne pouvez utiliser CURRENT_TIMESTAMP, la fonction qui renvoie la date/heure courante, car la date ne sera pas à zéro. Mais vous pouvez mettre la partie heure à zéro en faisant comme suit :

```
CAST(FLOOR(CAST(CURRENT_TIMESTAMP AS FLOAT)) AS DATETIME
```

VI - Transtypage vers FLOAT

Comme nous venons de le voir ci-dessus, le transtypage vers FLOAT est une chose qui va se révéler très utile avec la manipulation des DATETIME. En effet, en castant un DATETIME en FLOAT, on va obtenir un nombre dont la partie entière représente le nombre de jours écoulés depuis le 1er Janvier 1900 (date "zéro") et après la virgule, la fraction de jours représentée par le temps. Exemple :

```
SELECT CAST(CAST('1900-01-04 12:00' AS DATETIME) AS FLOAT)
```

Qui va nous retourner 3.5, donc 3 jours après le 1er janvier 1900 et 12 heures (soit une demi-journée).

On peut donc profiter de ce transtypage pour ajouter n jours, par exemple :

```
SELECT CURRENT_TIMESTAMP + 1
```

ajoute 1 jour

```
SELECT CURRENT_TIMESTAMP + 1.5
```

ajoute 1 jour et 12h (soit une demi-journée)

```
SELECT CURRENT_TIMESTAMP + 1.0 + 1/24.0 + 1/1440.0 + 1/86400.0
```

ajoute 1 jour, 1heure, 1 minute et 1 seconde

Bien entendu le transtypage inverse (FLOAT vers DATETIME) est possible.

VI - Recherche de dates

Prenons une table de dates :

```
CREATE TABLE T_DATES_TDT{
 TDT_ID INT,
 TDT_DATE DATETIME
}
INSERT INTO T_DATES (1, '2005-04-8 2:00:00.000')
INSERT INTO T_DATES (2, '2006-04-8 5:25:78.789')
INSERT INTO T_DATES (3, '2006-04-9 00:00:00.000')
INSERT INTO T_DATES (4, '2006-04-8 00:00:00.000')
```

Une idée malheureuse qui vient souvent à l'esprit pour rechercher toutes les dates du 8 avril 2006, consiste à faire :

```
SELECT * FROM T_DATES_TDT
WHERE TDT_DATE = '2006-04-8'
```

Hélas, vous n'allez récupérer que la ligne 4, car en l'absence de précision de l'heure, c'est à zéro heure que la partie temps opère. Une autre idée serait de faire :

```
SELECT * FROM T_DATES_TDT
WHERE TDT_DATE BETWEEN '2006-04-8' AND '2006-04-8 23:59:59.999'
```

Mais on retomberait dans la problématique de la précision et on obtiendrait aussi l'enregistrement 3.

Les bonnes solutions consistent à utiliser :
soit une fourchette non symétrique :

```
SELECT * FROM T_DATES_TDT
WHERE TDT_DATE >= '2006-04-8' AND TDT_DATE < '2006-04-9'
```

soit une précision à 3 ms de la fourchette BETWEEN :

```
SELECT * FROM T_DATES_TDT
WHERE TDT_DATE BETWEEN '2006-04-8'
               AND '2006-04-8 23:59:59.997'
```

Une dernière solution consiste à transtyper la colonne date en forçant l'heure à zéro, comme ceci :

```
SELECT * FROM T_DATES_TDT
WHERE CAST(FLOOR(CAST(TDT_DATE AS FLOAT)) AS DATETIME) = '2006-04-8'
```

Ce qui va aussi vous retourner les lignes 2 et 4. La fonction FLOOR permet de supprimer tout ce qui se trouve après la virgule pour n'avoir plus que la partie qui nous intéresse, c'est-à-dire la partie date.

VIII - Recherche sur une partie de date

Admettons que vous vouliez récupérer maintenant toutes les dates du 8 avril, mais de n'importe quelle année... SQL Server fournit des fonctions très pratiques d'extraction de parties de dates :

- MONTH : Permet de récupérer le mois d'un DATETIME
- YEAR : Permet de récupérer l'année d'un DATETIME
- DAY : Permet de récupérer le jour d'un DATETIME

On va donc pouvoir faire notre petite recherche :

```
SELECT * FROM T_DATES_TDT
WHERE MONTH(TDT_DATE) = 4 AND DAY(TDT_DATE) = 8
```

Ce qui va vous renvoyer les éléments 1, 2 et 4.

Plus généralement, SQL Server fournit une fonction d'extraction généraliste de partie de date nommée DATEPART. Elle doit prendre un argument supplémentaire parmi les suivants :

partie	argument
année	year, yy, yyyy
trimestre	quarter, qq, q
mois	month, mm, m
jour de l'année	dayofyear, dy, y
jour du mois	day, dd, d
semaine (*)	week, wk, ww
jour de semaine (**)	weekday, wd
heure	hour, hh
minute	minute, min
seconde	second, ss, s
milliseconde	millisecond, ms

(*) attention, ce numéro de semaine n'est pas celui de la définition ISO des semaines et ne convient pas à nos calendriers.
(**) attention car le premier jour de la semaine est conditionné par le flag SET DATEFIRST.

En ce qui concerne le jour de la semaine, si le flag FIRSTDATE est positionné à 7 (dimanche) alors les semaines sont considérées commençant à dimanche et l'application de la fonction DATEPART(wd, ...) donne respectivement 1 pour dimanche, 2 pour lundi, etc...
Pour pallier à cet inconvénient et être sûr que 1 = Lundi quelque soit le positionnement du flag SET DATEFIRST, vous pouvez utiliser l'UDF suivante :

```
CREATE FUNCTION F_ISO_WEEKDAY (@D DATETIME)
   RETURNS TINYINT
AS
BEGIN
   RETURN (((DATEPART(wd, @D) + @@DATEFIRST -1) -1) % 7) + 1
END
```

En ce qui concerne les numérotations ISO des semaines, vous pouvez utiliser l'UDF suivante :

```
CREATE FUNCTION F_ISO_WEEK  (@DATE DATETIME)
RETURNS INT
AS
```

```
BEGIN
   DECLARE @ISOWEEK INT
   SET @ISOWEEK = DATEPART(wk, @DATE) + 1
                - DATEPART(wk, CAST(DATEPART(yy,@DATE) as CHAR(4)) + '0104')

 -- si le premier jour de l'année est un vendredi, samedi ou dimanche,
 -- alors cette fin de semaine appartient à l'année passée.
   IF @ISOWEEK = 0
      SET @ISOWEEK = dbo.F_ISO_WEEK2(CAST(DATEPART(year, @DATE)-1 AS CHAR(4)) + '12' +
   CAST(24 + DATEPART(DAY,@DATE) AS CHAR(2))) + 1

 -- si le dernier jour de l'année est un lundi, mardi, mercredi ou jeudi, a
 -- lors ce début de semaine appartient à l'année suivante.
   IF DATEPART(month, @DATE) = 12 AND
      DATEPART(day,   @DATE) - dbo.F_ISO_WEEKDAY(@DATE) >= 28
      SET @ISOWEEK = 1

   RETURN @ISOWEEK
END
```

Attention à l'utilisation de fonctions dans des requêtes, car elles peuvent s'avérer peu performantes pour des requêtes lancées régulièrement. Il vaudrait mieux revoir le design de votre base de données et l'adapter de façon à ne plus avoir besoin de telles fonctions.
Voyez l'article suivant :
http://sqlpro.developpez.com/cours/gestiontemps/

IX - Recherche par heure

Prenons la table :

```
CREATE TABLE T_TIMES{
 id INT,
 hour DATETIME
}
INSERT INTO T_TIMES (1, '2006-02-28 2:00:00.000')
INSERT INTO T_TIMES (2, '1900-01-01 5:58:32.823')
INSERT INTO T_TIMES (3, '1900-01-01 1:59:59.997')
INSERT INTO T_TIMES (4, '1900-01-01 2:00:00.000')
```

Là encore, cette table a été remplie de valeurs de date et de temps mélangés, ce qui est à proscrire.

Si vous voulez rechercher tous les temps à 2h, vous feriez :

```
SELECT * FROM T_TIMES
WHERE hour = '2:00:00'
```

Malheureusement, cela va vous retourner seulement l'enregistrement numéro 4.

Pour récupérer les bons résultats de manière sûre, il va falloir à nouveau employer le transtypage FLOAT :

```
SELECT * FROM T_TIMES
WHERE hour - CAST(FLOOR(CAST(hour AS float)) AS datetime) = '10:00'
```

Cette fois, on va enlever la première partie de la date, pour ne conserver que l'heure et ainsi pouvoir la comparer exactement. Cela va nous retourner les lignes 1 et 4. Mais cela risque de se révéler lourd.

Une autre manière de faire, mais qui va nous retourner des résultats plus approximatifs et d'employer les opérateurs de comparaison :

```
SELECT * FROM T_TIMES
WHERE hour > '1:59' AND hour < '2:01'
```

Qui va nous retourner les enregistrements 3 et 4.

Si vous voulez vraiment récupérer toutes les heures à 2h, il va vous falloir faire quelque chose de plus lourd avec le transtypage vers FLOAT et les opérateurs de comparaison :

```
SELECT * FROM T_TIMES
 WHERE hour - CAST(FLOOR(CAST(hour AS float)) AS datetime) > '01:59'
   AND hour - CAST(FLOOR(CAST(hour AS float)) AS datetime) < '2:01'
```

Ce qui va cette fois vous retourner les enregistrements et 1,3 et 4.

X - Conversion DATETIME en String

Lorsque vous extrayez une date sous forme littérale, SQL Server se servira à nouveau des paramètres régionaux pour formater la chaîne de caractères. Vous risquez d'être surpris par le résultat...
Si vous voulez un format immuable et reproductible, vous devez utiliser la fonction CONVERT, qui, à l'aide d'un paramètre de style peut vous donner différentes présentations :

Style	Présentation
0 ou 100	mois jj aaaa hh:mmAM (ou PM)
101	mm/jj/aa
102	aa.mm.jj
103	jj/mm/aa
104	jj.mm.aa
105	jj-mm-aa
106	jj-mm-aa
107	mois jj, aa
108	hh:mm:ss
9 ou 109	mois jj aaaa hh:mm:ss:mmmAM (ou PM)
110	mm-jj-aa
111	aa/mm/jj
112	aammjj
13 ou 113	jj mois aaaa hh:mm:ss:mmm
114	hh:mi:ss:mmm
20 ou 120	aaaa-mm-jj hh:mm:ss
21 ou 121	aaaa-mm-jj hh:mm:ss.mmm
126	aaaa-mm-jj Thh:mm:ss.mmm
130	jj mon aaaa hh:mm:ss:mmmAM
131	jj/mm/aa hh:mm:ss:mmmAM

La fonction CONVERT prend trois paramètres : le type dans lequel vous voulez votre variable, par exemple un VARCHAR, l'expression de type DATETIME et ensuite l'identifiant du style. Par exemple :

```
CONVERT(VARCHAR, date, 101)
```

Vous pouvez aussi bien entendu employer cette méthode lors de l'insertion pour être dans le bon format, par exemple le format US :

```
INSERT INTO T_DATES SELECT CONVERT(DATETIME,'05/08/2004',101)
```

XI - Fonctions sur type DATETIME

Voici quelques unes des fonctions les plus courantes applicables au type DATETIME

DATEADD	Ajout d'une partie de date
DATEDIFF	Différence entre deux date pour une granularité de temps passée en paramètre
DATENAME	Nom d'une partie de date valuée. Accepte en paramètre : year, quarter, month, dayofyear, day, week, weekday, hour, minute, second, millisecond (*)
DATEPART	Renvoie une partie de date (voir chapitre VIII)
YEAR	Renvoie l'année d'une date
MONTH	Renvoie le mois d'une date
DAY	Renvoie le jour du mois d'une date
GETDATE	Renvoie la date/heure courante
CURRENT_TIMESTAMP	Renvoie la date/heure courante (fonction normative)
GETUTCDATE	Renvoie la date/heure UTC (solaire)
CAST	Transtype
COALESCE	Dénullifie

(*) en fonction des paramètres de langue pour month et day

Exemples :

```
SELECT DATEDIFF(second, '20060101', '200601 11:22:33.444') AS SECONDES
```

SECONDES ---------- 454936953

```
SELECT DATENAME(month, '20060101') AS MOIS,
       DATENAME(quarter, '20060101') AS TRIMESTRE,
       DATENAME(year, '20060101') AS AN,
       DATENAME(weekday, '20060101') AS JOUR_SEMAINE
```

MOIS TRIMESTRE AN JOUR_SEMAINE ---------- ---------- -------- ---------- janvier 1 2006 dimanche

En prime nous vous livrons une UDF pour calculer le dernier jour du mois :

```
CREATE FUNCTION F_LASTDAY_MONTH (@D DATETIME)
   RETURNS INT
AS
BEGIN

-- ajout d'un mois
SET @D = DATEADD(MONTH, 1, @D)

-- recherche du premier jour du mois
SET @D = CAST(YEAR(@D) AS CHAR(4))
       + CASE
            WHEN MONTH(@D) > 9 THEN CAST(MONTH(@D) AS CHAR(2))
            ELSE '0' + CAST(MONTH(@D) AS CHAR(1))
         END
       + '01'
```

```
-- retrait d'un jour
SET @D = DATEADD(day, -1, @D)

RETURN DAY(@D)

END
```

XII - Jointures sur des dates

Si vous avez des requêtes dont les jointures s'effectuent sur des dates et que ces colonnes de type DATETIME sont indexées (des clefs par exemple), les performances des requêtes d'extraction portant sur de tels critères seront améliorées si vous avez optimisé votre base de données pour ce faire.

```
ALTER DATABASE <MaBase>
SET DATE_CORRELATION_OPTIMIZATION ON
```

Attention, si cela vous apporte un gain dans le SELECT, la performance en mise à jour (INSERT, UPDATE, DELETE) décroit. Soyez sûr de ce que vous faîtes et mesurez-en l'impact avant une mise en production définitive !

Si vous voulez savoir si votre base est déjà optimisée pour la mise en relation des dates, vous pouvez lancer la requête suivante :

```
SELECT Name, is_date_correlation_on
FROM   sys.databases
```

Ceci n'est valable que pour la version 2005 de SQL Server.

XIII - Conclusion

Vous savez maintenant pratiquement tout sur la gestion des DATETIME dans SQL Server. La manipulation de données à base de date et d'heure est un des grands problèmes classiques dans les SGBDR et ne constitue à l'évidence pas une partie facile. Un bon gage de performance et de souplesse consiste à être très strict au niveau du design de la base, par exemple en ne mélangeant pas les dates, heures et dates/heures dans la même colonne.

XIII-A - Remerciements

Je tiens à remercier **Bisûnûrs** ainsi que **Xo** pour leurs corrections.