Anexo. Tipos de Datos e intercalaciones

1. Tipos de datos (Transact-SQL)

Referencia: **http://technet.microsoft.com/en-us/library/ms187752(SQL.90).aspx**

1.1 Introducción

En SQL Server 2005, cada columna, variable local, expresión y parámetro tiene un tipo de datos relacionado. Un tipo de datos es un atributo que especifica los valores que el objeto puede contener: datos enteros, de caracteres, de moneda, de fecha y hora, cadenas binarias, etc.

SQL Server proporciona un conjunto de tipos de datos del sistema que define todos los tipos de datos que pueden utilizarse con SQL Server. También podemos definir nuestros propios tipos de datos en Transact-SQL o Microsoft .NET Framework. Los tipos de datos de alias están basados en los tipos de datos proporcionados por el sistema. Para obtener más información acerca de los tipos de datos de alias, vea Trabajar con tipos de datos de alias. Los tipos definidos por el usuario obtienen sus características de los métodos y los operadores de una clase que se crean mediante uno de los lenguajes de programación compatibles con .NET Framework. Para obtener más información, vea Trabajar con tipos definidos por el usuario para CLR.

Cuando dos expresiones que tienen tipos de datos, intercalaciones, precisión, escala o longitud diferentes son combinadas por un operador, las características del resultado vienen determinadas por lo siguiente:

- El tipo de datos del resultado viene determinado por la aplicación de las reglas de precedencia de tipos de datos a los tipos de datos de las expresiones de entrada. Para obtener más información, vea Prioridad de tipo de datos (Transact-SQL).

- La intercalación del resultado viene determinada por las reglas de precedencia de intercalación cuando el tipo de datos del resultado es **char**, **varchar**, **text**, **nchar**, **nvarchar** o **ntext**. Para obtener más información, vea Prioridad de intercalación (Transact-SQL).

- La precisión, escala y longitud del resultado dependen de la precisión, escala y longitud de las expresiones de entrada. Para obtener más información, vea Precisión, escala y longitud (Transact-SQL).

SQL Server 2005 proporciona sinónimos de tipos de datos para la compatibilidad con SQL-92. Para obtener más información, vea Sinónimos de tipos de datos (Transact-SQL).

1.2 Categorías de tipos de datos

Los tipos de datos de SQL Server 2005 se organizan en las siguientes categorías:

Numéricos exactos	Cadenas de caracteres Unicode
Numéricos aproximados	Cadenas binarias
Fecha y hora	Otros tipos de datos
Cadenas de caracteres	

En SQL Server 2005, según las características de almacenamiento, algunos tipos de datos están designados como pertenecientes a los siguientes grupos:

- Tipos de datos de valores grandes: **varchar(max)**, **nvarchar(max)** y **varbinary(max)**

- Tipos de datos de objetos grandes: **text**, **ntext**, **image**, **varchar(max)**, **nvarchar(max)**, **varbinary(max)** y **xml**

1.2.1 Numéricos exactos

	Tipo	Almacenamiento	Valores aceptados	
Bit	bit	1 bit	0 o 1	
Enteros	tinyint	1 byte = 8 bits	De 0 a 255 (2^8 -1)	No tienen parte decimal
	smallint	2 bytes	De -32.768 (-2^{15}) a 32.767 (2^{15}-1)	
	int	4 bytes	De -2^{31} (-2.147.483.648) a 2^{31}-1 (2.147.483.647)	
	bigint	8 bytes	De -2^{63} (-9.223.372.036.854.775.808) a 2^{63}-1 (9.223.372.036.854.775.807)	
Coma fija	decimal[**(***p*[**,** *s*]**)**] numeric[**(***p*[**,** *s*]**)**]	Se almacena cada dígito del número con su equivalente en binario. Ocupa 5,9,13,17 bytes dependiendo de la precisión	Lo define la precisión y escala. Precisión máxima 38	**p**recisión: nº total de dígitos e**s**cala: nº de decimales ejemplo: decimal(16,4) admite nº negativos y positivos de 12 dígitos en la parte entera y 4 decimales.
Moneda	money	8 bytes	De -922,337,203,685.477,5808 a 922,337,203,685.477,5807	Tienen siempre 4 decimales Aparece el símbolo monetario correspondiente.
	smallmoney	4 bytes	De - 214.748,3648 a 214.748,3647	

1.2.2 Numéricos aproximados

	Tipo	Almacenamiento	Valores aceptados	
Coma flotante	real	4 bytes	De - 3,40E + 38 a -1,18E - 38, 0 y de 1,18E - 38 a 3,40E + 38	Se codifican en forma exponencial -> Son números aproximados
Bit	float[**(** *n* **)**] n: nº de bits que se utilizan para almacenar la mantisa del número en notación científica y, por tanto, dicta su precisión y el tamaño de almacenamiento.	Depende de n SQL Server 2005 trata *n* como uno de dos valores posibles. Si **1**<=n<=**24**, *n* se trata como **24**. Si **25**<=n<=**53**, *n* se trata como **53**. 53 es valor máximo y predeterminado	De - 1,79E+308 a -2,23E-308, 0 Y de 2,23E-308 a 1,79E+308	

1.2.3 Fecha y hora

Tipo	Valores aceptados	
datetime	Del 1 de enero de 1753 hasta el 31 de diciembre de 9999	Precisión 3,33 milisegundos
smallint	Del 1 de enero de 1900 hasta el 6 de junio de 2079	Precisión 1 minuto

En SQL Server 2008 se mantienen estos tipos y aparecen los tipos time (para sólo horas), date (para solo fechas) y nuevos datetime.

1.2.4 Cadenas de caracteres

	Tipo	Valores aceptados	Almacenamiento	
Longitud fija	char[(*n*)] *n* € [1, 8.000]	Cadenas de caracteres no Unicode	n bytes	con una longitud de *n* bytes aunque los valores almacenados sean más cortos. n predeterminado: 1 en definiciones, 30 en CAST() y CONVERT()
	nchar[(*n*)] *n* € [1, 4.000]	Cadenas de caracteres Unicode.	2n bytes	
Longitud variable	varchar[(*n*|MAX)] *n* € [1, 8.000]	Cadenas de caracteres no Unicode	1 byte/carácter +2 bytes (para la longitud)	con una longitud máxima de *n* bytes pero ocupa según el valor almacenado. n predeterminado: 1 en definiciones, 30 en CAST() y CONVERT() **max** indica que el tamaño de almacenamiento máximo es de 2^{31}-1 bytes.
	nvarchar[(*n*|MAX)] *n* € [1, 4.000]	Cadenas de caracteres Unicode.	2 bytes/carácter +2 bytes (para la longitud)	
	text	En vía de extinción no se recomienda su uso. Equivale a varchar(max)		
	ntext	En vía de extinción no se recomienda su uso. Equivale a nvarchar(max)		

1.2.5 Cadenas binarias

Tipo	Valores aceptados	Almacenamiento	
binary[(*n*)] *n* € [1, 8.000]	Valor binario de n bytes	n bytes	con una longitud fija de *n* bytes independientemente de los valores almacenados. n predeterminado: 1 en definiciones, 30 en CAST()
varbinary[(*n*)] *n* € [1, 8.000]	Valor binario de hasta n bytes	Según lo almacenado +2 bytes (para la longitud)	ocupa según el valor almacenado. n predeterminado: 1 en definiciones, 30 en CAST()
varbinary(max)	Valor binario de hasta 2^{31}-1 bytes	Según lo almacenado +2 bytes (para la longitud)	
image	En vía de extinción no se recomienda su uso. Equivale a varbinary(max)		

1.2.6 Otros tipos de datos

cursor	timestamp
sql_variant	uniqueidentifier
table	xml

No detallaremos estos tipos ya que no los utilizaremos en este curso. Sólo comentaremos, para evitar confusiones que t**imestamp** no sirve para fechas, es un tipo de datos que expone números binarios únicos generados automáticamente en una base de datos. **Timestamp** suele utilizarse como mecanismo para marcar la versión de las filas de la tabla. El tamaño de almacenamiento es de 8 bytes. El tipo de datos **timestamp** es simplemente un número que se incrementa y no conserva una fecha o una hora.

El tipo **uniqueidentifier** se utiliza para almacenar GUIDs. Un **G**lobally **U**nique **ID**entifier es un número pseudoaleatorio que pretende ser único en todos los equipos y redes, aunque no se puede garantizar que cada GUID generado sea único, la probabilidad de que se genere un mismo número dos veces es muy baja, casi nula. Los GUIDs se suelen generar mediante la función NEWID(). Un GUID es un entero de 128 bits (16 bytes).

1.3 Sinónimos de tipos de datos (Transact-SQL)

http://technet.microsoft.com/en-us/library/ms177566(SQL.90).aspx

Los sinónimos de tipos de datos se incluyen en SQL Server 2005 por compatibilidad con SQL-92. En la siguiente tabla se incluyen los sinónimos y los tipos de datos de sistema de SQL Server a los que se asignan.

Sinónimo	Tipo de datos de sistema de SQL Server
Binary varying	varbinary
char varying	varchar
character	char
character	char(1)
character(*n*)	char(n)
character varying(*n*)	varchar(n)
Dec	decimal
Double precision	float
float[(*n*)] para *n* = 1-7	real
float[(*n*)] para *n* = 8-15	float
integer	int
national character(*n*)	nchar(n)
national char(*n*)	nchar(n)
national character varying(*n*)	nvarchar(n)
national char varying(*n*)	nvarchar(n)
national text	ntext
rowversion	timestamp

Los sinónimos de tipos de datos pueden utilizarse en lugar del nombre del tipo de datos base correspondiente en las instrucciones del lenguaje de definición de datos (DDL), como CREATE TABLE, CREATE PROCEDURE o DECLARE *@variable*. Sin embargo, los sinónimos no tienen visibilidad después de crear el objeto. Una vez creado el objeto, se le asigna el tipo de datos base asociado al sinónimo. No hay ningún registro de que el sinónimo se haya especificado en la instrucción que ha creado el objeto.

A todos los objetos que proceden del objeto original, como las columnas del conjunto de resultados o las expresiones, se les asigna el tipo de datos base. Todas las funciones de metadatos subsiguientes ejecutadas en el objeto original y cualquier objeto derivado informarán del tipo de datos base y no del sinónimo. Este comportamiento se produce con las operaciones de metadatos como **sp_help** y otros procedimientos almacenados del sistema, las vistas del esquema de información o las diferentes operaciones de metadatos de la API de acceso a datos que informan de los tipos de datos de las columnas de tablas o conjuntos de resultados.

Por ejemplo, se puede crear una tabla utilizando el sinónimo national character varying:

```
CREATE TABLE ExampleTable (PriKey int PRIMARY KEY, VarCharCol national
character varying(10))
```

A VarCharCol se le asigna el tipo de datos nvarchar(10) y todas las funciones de metadatos posteriores informan de la columna como columna nvarchar(10). Las funciones de metadatos nunca informarán de la columna como national character varying(10).

1.4 Precisión, escala y longitud (Transact-SQL)

http://technet.microsoft.com/en-us/library/ms190476(SQL.90).aspx

La precisión es el número de dígitos de un número. La escala es el número de dígitos situados a la derecha de la coma decimal de un número. Por ejemplo, el número 123,45 tiene una precisión de 5 y una escala de 2.

En SQL Server 2005, la precisión máxima predeterminada de los tipos de datos **numeric** y **decimal** es 38. En versiones anteriores de SQL Server, el valor predeterminado máximo es 28.

La longitud de un tipo de datos numérico es el número de bytes utilizados para almacenar el número. La longitud para una cadena de caracteres o tipo de datos Unicode es el número de caracteres. La longitud para los tipos de datos **binary**, **varbinary** y **image** es el número de bytes. Por ejemplo, un tipo de datos **int** que puede contener 10 dígitos se almacena en 4 bytes y no acepta coma decimal. El tipo de datos **int** tiene una precisión de 10, una longitud de 4 y una escala de 0.

Cuando se concatenan dos expresiones **char**, **varchar**, **binary** o **varbinary**, la longitud de la expresión resultante es la suma de las longitudes de las dos expresiones de origen u 8.000 caracteres, lo que sea menor.

Cuando se concatenan dos expresiones **nchar** o **nvarchar**, la longitud de la expresión resultante es la suma de las longitudes de las dos expresiones de origen o 4.000 caracteres, lo que sea menor.

Cuando se comparan dos expresiones del mismo tipo de datos pero de distintas longitudes mediante UNION, EXCEPT o INTERSECT, la longitud resultante es la longitud máxima de las dos expresiones.

La precisión y la escala de los tipos de datos numéricos, excepto **decimal**, son fijas. Si un operador aritmético tiene dos expresiones del mismo tipo, el resultado tiene el mismo tipo de datos con la precisión y la escala definidas para ese tipo de datos. Si un operador tiene dos expresiones con tipos de datos numéricos diferentes, las reglas de prioridad de tipos de datos definen el tipo de datos del resultado. El resultado tiene la precisión y la escala definidas para el tipo de datos que le corresponde.

Esta tabla define cómo se calculan la precisión y la escala del resultado de la operación cuando éste es de tipo **decimal**. El resultado es **decimal** cuando se cumple alguna de las siguientes condiciones:

Ambas expresiones son de tipo **decimal**.

Una expresión es **decimal** y la otra es de un tipo de datos con una prioridad menor que **decimal**.

Las expresiones de operando se denotan como expresión e1, con precisión p1 y escala s1, y expresión e2, con precisión p2 y escala s2. La precisión y la escala de cualquier expresión que no sea **decimal** es la precisión y la escala definidas para el tipo de datos de la expresión.

Operación	Precisión del resultado	Escala del resultado *
e1 + e2	máx(s1, s2) + máx(p1-s1, p2-s2) + 1	máx(s1, s2)
e1 - e2	máx(s1, s2) + máx(p1-s1, p2-s2) + 1	máx(s1, s2)
e1 * e2	p1 + p2 + 1	s1 + s2
e1 / e2	p1 - s1 + s2 + máx(6, s1 + p2 + 1)	máx(6, s1 + p2 + 1)
e1 { UNION | EXCEPT | INTERSECT } e2	máx(s1, s2) + máx(p1-s1, p2-s2)	máx(s1, s2)

* La precisión y la escala del resultado tienen un valor máximo absoluto igual a 38. Cuando la precisión de un resultado es mayor que 38, la escala correspondiente se reduce para evitar que la parte entera del resultado se trunque.

1.5 Prioridad de tipo de datos (Transact-SQL)

http://technet.microsoft.com/en-us/library/ms190309(SQL.90).aspx

Cuando un operador combina dos expresiones de tipos de datos distintos, las reglas de prioridad de tipo de datos especifican que el tipo de datos con la prioridad más baja se convierta al tipo de datos con la prioridad más alta. Si la conversión no es una conversión implícita admitida, se devuelve un error. Cuando ambas expresiones de operandos tienen el mismo tipo de datos, el resultado de la operación tiene ese tipo de datos.

SQL Server 2005 utiliza el siguiente orden de prioridad para los tipos de datos:

1. **tipos de datos definidos por el usuario** (el + alto)
2. **sql_variant**
3. **xml**
4. **datetime**
5. **smalldatetime**
6. **float**
7. **real**
8. **decimal**
9. **money**
10. **smallmoney**
11. **bigint**
12. **int**
13. **smallint**
14. **tinyint**
15. **bit**
16. **ntext**
17. **text**
18. **image**
19. **timestamp**
20. **uniqueidentifier**
21. **nvarchar**
22. **nchar**
23. **varchar**
24. **char**
25. **varbinary**
26. **binary** (el + bajo)

2. Intercalaciones

http://technet.microsoft.com/en-us/library/ms187582(SQL.90).aspx

Un nombre de intercalación *COLLATION* define el juego de caracteres que se podrá utilizar, las reglas de ordenación y comparación incluso el comportamiento frente a minúsculas, mayúsculas y acentos que seguirá el servidor.

Un nombre de intercalación se compone de un designador de intercalación y los estilos de comparación.

Los estilos de comparación permiten definir el comportamiento frente a mayúsculas/minúsculas, acentos, y otras opciones. Por ejemplo:

_CS Case-Sensitive indica que se distingue entre mayúsculas y minúsculas

_CI Case-Insensitive no se diferencian las mayúsculas de las minúsculas

_AS Accent-Sensitive sensible a los acentos

_AI Accent-Sensitive no se diferencian las letras acentuadas de las no acentuadas.

El designador de intercalación define la página de códigos (los caracteres) y la forma en que se comparan y ordenan esos caracteres.

Por ejemplo:

Modern_Spanish y Latin1_General utilizan los dos la página de códigos 1252 (para caracteres latinos) pero en Modern_Spanish la ñ y la n son letras diferentes mientras que en Latin1_General la ñ es una n acentuada.

Latin1_General_BIN utiliza la página de códigos 1252 y las reglas de orden binario. Se pasan por alto las reglas de orden del diccionario Latín-1 general. Las intercalaciones binarias ordenan y comparan datos de SQL Server según el patrón de bits de cada carácter. Cada intercalación binaria de SQL Server se asigna a una página de códigos ANSI y a una configuración regional de idioma específicos y cada una de ellas realiza ordenaciones de datos que distinguen mayúsculas de minúsculas y acentos. En este caso _BIN indica el estilo de comparación.

Ejemplos de nombres de intercalación:

Modern_Spanish_CS_AI

Modern_Spanish_CI_AI

Modern_Spanish_CI_AS