IES CHAN DO MONTE
C.S. de Desarrollo de Aplicaciones Multiplataforma
Módulo Base de datos

Anexo: Nombrar objetos y tipos de datos

Índice

1. Elementos de Transact-SQL

SQL es un lenguaje de consulta para los sistemas de bases de datos relaciónales, pero que no posee la potencia de los lenguajes de programación.

- Transact-SQL es el lenguaje que se utiliza para administrar instancias del SQL Server Database Engine (Motor de base de datos de SQL Server), para crear y administrar objetos de base de datos, y para insertar, recuperar, modificar y eliminar datos.
- **Transact-SQL es una extensión del lenguaje definido en los estándares de SQL** publicados por International Standards Organization (ISO) y American National Standards Institute (ANSI).
- Transact SQL es el lenguaje de programación que proporciona SQL Server para ampliar SQL con los elementos característicos de los lenguajes de programación: variables, sentencias de control de flujo, bucles, etc..

Cuando se desea realizar una aplicación completa para el manejo de una base de datos relacional, resulta necesario utilizar alguna herramienta que soporte la capacidad de consulta del SQL y la versatilidad de los lenguajes de programación tradicionales. Transact SQL es el lenguaje de programación que proporciona SQL Server para extender el SQL estándar con otro tipo de instrucciones.

Transact SQL es el lenguaje de programación que proporciona Microsoft SQL Server para extender el SQL estándar con otro tipo de instrucciones y elementos propios de los lenguajes de programación. Con **Transact SQL vamos a poder programar las unidades de programa de la base de datos SQL Server**, están son: Procedimientos almacenados, Funciones, Triggers

Transact SQL **no es CASE-SENSITIVE**, es decir, no diferencia mayúsculas de minúsculas como otros lenguajes de programación como C o Java.

2. Nombres de objetos en SQL Server

- El **nombre de un objeto de base de datos** se conoce como **su identificador**.
- Cualquier elemento puede tener un identificador: Servidores, bases de datos y objetos de bases de datos, como tablas, vistas, columnas, índices, desencadenadores, procedimientos, restricciones, reglas, etc.
- **Se requiere que la mayor parte de los objetos tengan identificadores**; pero para ciertos objetos, como las *restricciones*, son opcionales.
- El **identificador** de un objeto **se crea** cuando **se define el objeto**.
- SQL Server proporciona **una serie de reglas estándar** para identificadores de objetos y un método para usar delimitadores de los identificadores que no son estándar.

  Es recomendable que los nombres de objetos usen caracteres identificadores estándar cuando sea posible.

A continuación, el identificador se utiliza para hacer referencia al objeto. En este caso la restricción PRIMARY KEY no se le ha proporcionado un nombre explícito.

```
USE Empresa_Clase
CREATE TABLE Empleado(
  IDEmpleado int PRIMARY KEY,
  Nombre varchar(15),
  Apellidos varchar(40),
  Telefono char(9))
```

2.1 Clases de identificadores

Existen ***dos clases*** de identificadores:

- **Identificadores normales: Siguen las reglas de formato de los identificadores**. Los identificadores normales **no están delimitados** cuando se usan en instrucciones Transact-SQL.

```
SELECT * FROM TableX
WHERE KeyCol = 124
```

Reglas:

- El **primer carácter** debe ser alguno de los siguientes:
  - ✓ **Una letra**, como aparece definida por el estándar Unicode 3.2.
    La definición Unicode de letras incluye los caracteres latinos de la "a" a la "z" y de la "A" a la "Z", además de los caracteres de letras de otros idiomas.
  - ✓ **El signo de subrayado (_), arroba (@) o (#).**
- Los **caracteres subsiguientes** pueden ser:
  - ✓ **Letras**, tal como se definen en el estándar Unicode 3.2.
  - ✓ **Números** del alfabeto Latín básico u otros alfabetos de otros idiomas.
  - ✓ **El signo de arroba@, dólar ($), o subrayado**.
- **No debe ser una palabra reservada** de Transact-SQL.
- **No se permiten los caracteres especiales o los espacios incrustados**.

- **Identificadores delimitados**: Los identificadores que **no cumplen las reglas de los identificadores** deben **estar delimitados** en las instrucciones Transact-SQL. Se incluyen entre comillas dobles (") o corchetes ([ ]).

```
SELECT *FROM [Mi Tabla]      --Identificador contiene un espacio.
WHERE [order] = 10   --Identificador es una palabra reservada
```

Los identificadores entre comillas dobles **sólo son válidos** si la opción **QUOTED_IDENTIFIER es ON.**

3. Referencia a los objetos SQL Server.

Se pueden **usar distintas formas para referirnos a los objetos SQL Server**.

- ✓ Se puede especificar el **nombre completo del objeto**.
- ✓ especificar **sólo una parte del nombre del objeto** y dejar que SQL Server determine el resto del nombre según el contexto en el que se está trabajando.

- **Nombres completamente cualificados.**
  - ✓ El nombre completo de un objeto incluye **cuatro identificadores**: el **nombre del servidor, nombre de la base de datos, el esquema y el nombre del objeto** en el siguiente formato:

    Servidor.BaseDeDatos.Esquema.Objeto

- **Nombres parcialmente cualificados.**
  - ✓ Cuando se hace referencia a un objeto, **no siempre tiene que especificar** el servidor, base de datos y esquema.
  - ✓ **Los identificadores principales se pueden omitir**.
  - ✓ Los formatos válidos de los nombres de objetos son como se muestra a continuación:

```
Servidor.BaseDeDatos.Esquema.Objeto
BaseDeDatos.Esquema.Objeto
Esquema.Objeto
Objeto
BaseDeDatos..Objeto
```

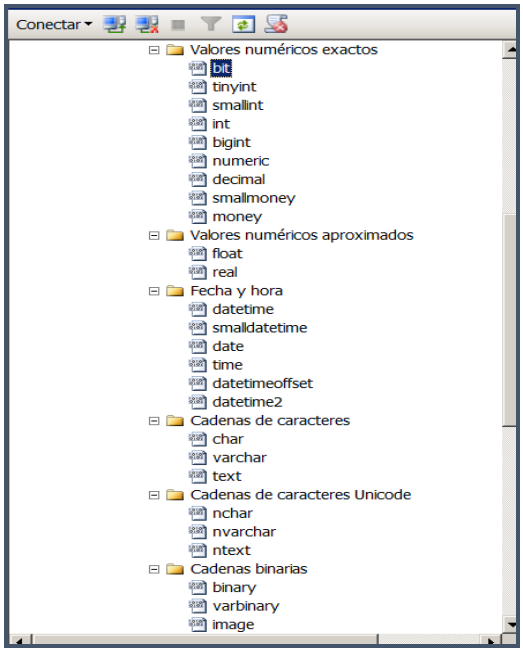
  - ✓ Se utiliza los seguintes **valores predeterminados** si diferentes partes del nombre no están especificadas:
    - El servidor está predeterminado al **servidor local**.
    - **Base de datos actual**.
    - El esquema **predeterminado del usuario**. No hace falta especificar el esquema cuando se accede a objetos en el esquema predeterminado.

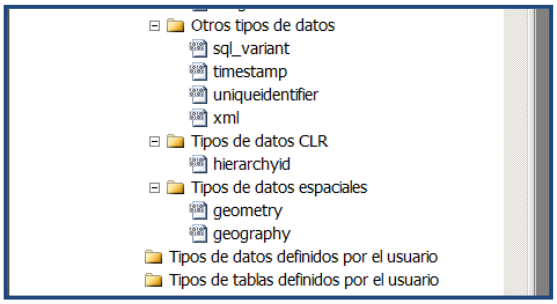
Ejemplo: El siguiente ejemplo crea una tabla Empleado en la base de datos Empresa con el **esquema predeterminado dbo** (es el esquema por defecto si no se le asigna otro)

```
CREATE TABLE Empresa..Empleado
(
 IDEmpleado int,
 Nombre varchar(15),
 Apellidos varchar(40),
 Telefono char(9)
)
```

4. TIPOS DE DATOS

- Todos los **objetos con datos tienen asociado un tipo de datos** en Transact-SQL.
- Los objetos que **tienen tipos de dato**s son:
  - ✓ Las **columnas de tablas** y **vistas**
  - ✓ Los **parámetros de procedimientos almacenados**,
  - ✓ **Las variables**
  - ✓ **Las funciones** de Transact-SQL que devuelven uno o más valores de datos de un tipo de datos específico





Cuando se asigna un **tipo de datos a un objeto** se definen **cuatro atributos** del objeto:

- ✓ El **tipo de datos** que contiene el objeto (carácter, entero, float o binario)
- ✓ La **longitud** del valor almacenado o su tamaño (es el **número de bytes necesarios** para contener el número de dígitos permitido para ese tipo de datos)
- ✓ La **precisión** del número para tipos de datos numéricos (la precisión es el **número de dígitos totales** que puede contener el número)
- ✓ La **escala** del número para tipos de datos numéricos (la escala es el máximo número de dígitos que se puede almacenar a la derecha del separador decimal)

Ejemplo:

- Si un objeto se define como **money,** puede tener un máximo de 19 dígitos y 4 de ellos pueden estar a la derecha del decimal. El objeto usa 8 bytes para almacenar los datos. Por tanto, el tipo de datos money tiene una precisión de 19, una escala de 4 y una longitud de 8.
- Por ejemplo, un tipo de **datos int** que puede contener 1**0** 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.

4.1 Categorías de tipos de datos

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

Numéricos exactos:

Tipo de datos	Intervalo	Almacenamiento
Bigint	De -2^63 **(-9.223.372.036.854.775.808)** a 2^63-1 **(9.223.372.036.854.775.807)**	8 bytes
Int	De -2^31 **(-2.147.483.648)** a 2^31-1 **(2.147.483.647)** **Es el principal tipo de datos de valores enteros** de SQL Server	4 bytes
Smallint	De -2^15 **(-32.768)** a 2^15-1 **(32.767)**	2 bytes
Tinyint	De **0 a 255**	1 byte
bit	Aceptar los valores **1(TRUE)**, 0 (FALSE) o NULL.	
numeric[(p[, s])] **o** **decimal[(p[, s])]**	Tipos de **datos numéricos** que tienen **precisión y escala fijas**. Cuando se utiliza la precisión máxima, los valores válidos se sitúan entre **- 10^38 +1 y 10^38 - 1.** numeric es funcionalmente equivalente a decimal. ▪ **p (precisión)**: El **número total máximo de dígitos** decimales que se puede almacenar, tanto a la izquierda como a la derecha del separador decimal.	Precisión → Bytes de almacenamiento 1 – 9 → 5 10-19 → 9 20-28 →13 29-38 →17

	La precisión debe ser un valor comprendido entre **1** y **la precisión máxima de 38**. **La precisión predeterminada es 18**. ▪ **s (escala)**:El número máximo de dígitos decimales que se puede almacenar a la derecha del separador decimal. La escala debe ser un valor comprendido entre 0 y p. Sólo es posible especificar la escala si se ha especificado la precisión. La escala predeterminada es 0; por lo tanto, 0 <= s <= p. Los tamaños de almacenamiento máximo varían, según la precisión.	
smallmoney	Tipos de datos que representan valores monetarios o de moneda.	4bytes
money	Tipos de datos que representan valores monetarios o de moneda.	8 bytes

Numéricos aproximados:

Tipos de datos numéricos y aproximados que se utilizan con datos **numéricos de coma flotante (números realiess)**. Los datos de coma flotante **son aproximados**; por tanto, **no todos los valores** del intervalo del tipo de datos se **pueden representar con exactitud**.

Tipo de datos	Intervalo	Almacenamiento
float [(n)]	De **- 1,79E+308 a -2,23E-308**, 0 y de **2,23E-308 a 1,79E+308** Donde n es el número de bits que se utilizan para almacenar la mantisa del número float en notación científica y, por tanto, dicta su precisión y el tamaño de almacenamiento. Si se especifica n, debe ser un valor entre 1 y 53. El **valor predeterminado de n es 53**.	Depende de n n->Precisión-> Tamaño de almacenamiento 1-24→7 dígitos→4 bytes 25-53→15 dígitos→8 bytes SQL Server **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**.
real	De **- 3,40E + 38 a -1,18E - 38, 0** y de **1,18E - 38 a 3,40E + 38**	4 bytes

Cadenas de caracteres:

-Podemos definir **caractéres en formato ASCII o Unicode**

- **Datos de carácteres formato ASCII** que tienen **longitud fija o variable**.

Tipo de datos	Intervalo	Almacenamiento
char [(n)]	Datos de caracteres **ASCII de longitud fija**, con una longitud de n bytes. **n** debe ser un valor entre 1 y 8.000. **El tamaño de almacenamiento es n bytes**.	Cuando no se especifica n la **longitud predeterminada es 1.** .
varchar [(n | max)]	Datos de caracteres ASCII de **longitud variable**. **n** puede ser un valor entre 1 y 8.000. **max** indica que el tamaño de almacenamiento máximo es de 2^31-1 bytes.	
text	**Datos ASCII** de longitud variable de datos de **gran tamaño** con una longitud máxima de 2^31-1 (2.147.483.647) caracteres. Este tipo se quitará en versiones futuras. Utilizar en su lugar varchar(max)	

- **Datos de carácteres formato UNICODE** que tienen **longitud fija o variable**.

  Cada carácter ocupa 2 bytes. Es recomendable utilizar este tipo de datos cuando los valores que vayamos a almacenar puedan **pertenecer a diferente idiomas**.

Tipo de datos	Intervalo	Almacenamiento
nchar [(n)]	Datos de carácter **Unicode** de **longitud fija**, con n caracteres. n debe estar comprendido entre 1 y 4.000. El tamaño de almacenamiento **es dos veces n bytes**.	Cuando no se especifica n en una instrucción de definición de datos o de declaración de variable, la **longitud predeterminada es**
nvarchar [(n | max)]	Datos de **carácter Unicode de longitud variable**.	

	n puede ser un valor comprendido entre 1 y 4.000. max indica que el tamaño máximo de almacenamiento es 2^31-1bytes (1.073.741.823) caracteres.	**1.** .
ntext	**Datos Unicode de longitud variable** con una longitud máxima de 2^30 - 1 (1.073.741.823) caracteres. El tamaño del almacenamiento, **en bytes**, es dos veces el número de caracteres especificado. Este tipo se quitará en versiones futuras. Utilizar en su lugar varchar(max)	

Ejemplo 1:

```
DECLARE @DATO1 Char(3)
DECLARE @DATO2 varchar(6)
SET @DATO1= 'ab'
SET @DATO2= 'abcd'
SELECT @DATO1 AS DATO1,DATALENGTH(@DATO1)AS NumBytes
SELECT @DATO2 AS DATO2,DATALENGTH(@DATO2)AS NumBytes

   RESULTADO:
   DATO1 NumBytes
   ----- -----------
   ab          3
   DATO2  NumBytes
   ------ -----------
   abcd       4
```

Ejemplo 2:

```
DECLARE @DATO1 nchar(3)
DECLARE @DATO2 nvarchar(6)
SET @DATO1= 'ab'
SET @DATO2= 'abcd'
SELECT @DATO1 AS DATO1,DATALENGTH(@DATO1)AS NumBytes
SELECT @DATO2 AS DATO2,DATALENGTH(@DATO2)AS NumBytes

RESULTADO:
DATO1 NumByte
----- -----------
ab    6
DATO2  NumBytes
------ -----------
abcd   8
```

Fecha y hora:

Son tipos de datos que se utilizan para representar la **fecha y la hora del día**.

Tipo de datos	Descripción	
datetime	Define una **fecha que se combina con una hora** del día con fracciones de segundos basada en un reloj de 24 horas	
	Intervalo de fechas	**Del 1 de enero de 1753 hasta el 31 de diciembre de 9999**
	Intervalo de horas	De 00:00:00 a 23:59:59.997
	Tamaño almacenamiento	**8bytes.**
	Especificación de la fecha en formato num	Formato numérico de fecha, se especifica el mes, el día y el año en una cadena con barras diagonales **(/)**, guiones **(-)** o puntos **(.)** como separadores. Se puede especificar los datos de la fecha con un mes especificado como el nombre completo del mes. Por ejemplo, abril o la abreviatura del mes Abr

```
DECLARE @MIFECHA DATETIME
SET @MIFECHA= '21/11/11 23:59:59.995'
DECLARE @MIFECHA2 DATETIME
SET @MIFECHA2= '21 NOV 11 23:59:59.995'
SELECT @MIFECHA AS FECHA_DATETIME
SELECT @MIFECHA2 AS FECHA_DATETIME
SELECT GETDATE()—Funcion que pbtiene la fecha del sistema
```

```
RESULTADO:
FECHA_DATETIME
-----------------------
2011-11-21 23:59:59.997   --Se redondea
FECHA_DATETIME
-----------------------
2011-11-21 23:59:59.997
2011-11-22 00:26:01.840 –fecha actual
```

Tipo de datos	Descripción	
smalldatetime	Define **una fecha que se combina con una hora del día**. La hora está en un formato de día de 24 horas, con segundos siempre a cero (: 00) y sin fracciones de segundo.	
	Intervalo de fechas	**Del 1 de enero de 1900 hasta el 6 de junio de 2079**
	Intervalo de horas	**De 00:00:00 a 23:59:59**
	Tamaño almacenamiento	**4bytes.**
	Especificación de la fecha en formato num	Formato numérico de fecha, se especifica el mes, el día y el año en una cadena con barras diagonales **(/)**, guiones **(-)** o puntos **(.)** como separadores. Se puede especificar los datos de la fecha con un mes especificado como el nombre completo del mes. Por ejemplo, abril o la abreviatura del mes Abr.

```
--El tiempo lo devuelve 12:35.
SELECT CAST('2003-05-08 12:35:29.998' AS smalldatetime);
--la function CAST comvierte la cadena a tipo de dato smalldatetime
GO
-- El tiempo lo devuelve as 12:36.
SELECT CAST('2003-05-08 12:35:29.999' AS smalldatetime);
GO
```

Resultado:
2003-05-08 12:35
2003-05-08 12:36

Tipo de datos		
date	Descripción	**Define una fecha**
	intervalo de fechas	**De 01-01-0001 a 31-12-9999**
	Tamaño almacenamiento	**8bytes.**
	Especificación de la fecha en formato numérico	Formato numérico de fecha, se especifica el mes, el día y el año en una cadena con barras diagonales **(/)**, guiones **(-)** o puntos **(.)** como separadores. Se puede especificar los datos de la fecha con un mes especificado como el nombre completo del mes. Por ejemplo, abril o la abreviatura del mes Abr

```
DECLARE @MIFECHA3 DATE
SET @MIFECHA3= '21/01/11'
SELECT @MIFECHA3
```

RESULTADO:
2011-01-21

Tipo de datos	Descripción	
Time	Define **una hora de un día**. La hora no distingue la zona horaria y está basada en un reloj de 24 horas	
	Intervalo de tiempo	**De 00:00:00.0000000 a 23:59:59.9999999**
	Tamaño almacenamiento	5 bytes (fijo) es el valor predeterminado con el valor predeterminado de 100 ns de precisión de fracciones de segundo.
	Formato	hh:mm:ss[.nnnnnnn]

```
DECLARE @MITIEMPO TIME
SET @MITIEMPO= '20:30'
SELECT @MITIEMPO
DECLARE @MITIEMPO2 TIME(3)
SET @MITIEMPO2= '20:30:2'
SELECT @MITIEMPO2
```

RESULTADO:
20:30:00.0000000
20:30:02.**000**

Tipo de datos	Descripción	
Datetime2	Define una **fecha** que se combina con una **hora** de día basada en el reloj de 24 horas. **datetime2 puede considerarse una extensión del tipo datetime existente que tiene un intervalo de fechas mayor**.	
	Intervalo de fechas	De **01-01-0001 a 31-12-99999**
	Intervalo de horas	**De 00:00:00 a 23:59:59.9999999**
	Tamaño almacenamiento	6 bytes para precisiones inferiores a 3; 7 bytes para precisiones 3 y 4. Todas las demás precisiones requieren 8 bytes.
	Formato	AAAA-MM-DD hh:mm:ss[.fracciones de segundos]

```
DECLARE @MIFECHA DATETIME2
SET @MIFECHA= '21/01/11 23:59:59.995'
SELECT @MIFECHA AS FECHA_DATETIME2
```

RESULTADO:
FECHA_DATETIME2

2011-01-21 23:59:59.99

Tipo de datos	Descripción	
datetimeoffset	Define una **fecha que se combina con una hora del día** con reconocimiento de zona horaria **(husos horarios**) y basado en un reloj de 24 horas.	
	Intervalo de fechas	De **01-01-0001 a 31-12-99999**
	Intervalo de horas	**De 00:00:00 a 23:59:59.9999999**
	Tamaño almacenamiento	10 bytes, fijo es el valor predeterminado con el valor predeterminado de 100 ns de precisión de fracciones de segundo.
	Longitud de los caracteres	De 26 posiciones como mínimo (AAAA-MM-DD hh:mm:ss {+|-}hh:mm) a 34 como máximo (AAAA-MM-DD hh:mm:ss.nnnnnnn {+|-}hh
	Formato	**AAAA-MM-DD hh:mm:ss[.nnnnnnn] [{+|-}hh:mm]**

```
DECLARE @MIFECHAOFFSET DATETIMEOFFSET
SET @MIFECHAOFFSET='2007-05-08 12:35:29.1234567+01:00'
DECLARE @MIFECHAOFFSET2 DATETIMEOFFSET(3)
SET @MIFECHAOFFSET2='2007-05-08 12:35:29.1234567+01:00'
SELECT @MIFECHAOFFSET AS FECHA_DATETIMEOFFSET
SELECT @MIFECHAOFFSET2 AS FECHA_DATETIMEOFFSET
select SYSDATETIMEOFFSET ( ) --Devuelve la fecha y hora del equipo
                             --con ajuste zona horaria
```

RESULTADO:
FECHA_DATETIMEOFFSET

2007-05-08 12:35:29.**1234567** +01:00
FECHA_DATETIMEOFFSET

2007-05-08 12:35:29.**123** +01:00
2011-11-22 00:18:35.8783445 +01:00

Cadenas binarias:

Los tipos de datos binary y varbinary almacenan **cadenas de bits**. Aunque los datos de caracteres se interpretan según la página de códigos de SQL Server, los datos binary y varbinary son simplemente una **secuencia de bits**.

Tipo de datos	Intervalo	Almacenamiento
binary [(n)]	**Datos binarios de longitud fija** con una longitud de n bytes, donde n es un valor que oscila entre 1 y 8.000. El tamaño de almacenamiento es de n bytes.	Cuando no se especifica n en una instrucción de definición de datos o de declaración de variable, la **longitud predeterminada es 1.**
varbinary [(n | max)]	**Datos binarios de longitud variable**. n puede ser un valor que oscila entre 1 y 8.000. **max** indica que el tamaño de almacenamiento máximo es de 2^31-1 bytes. .	
image	**Datos binarios de longitud variable** desde 0 hasta 2^31-1 (2.147.483.647) bytes. Este tipo se quitará en versiones futuras. Utilizar en su lugar varbinary(max)	

Otros tipos de datos:

Tipo de datos	función
cursor	Una referencia a un cursor.
uniqueidentifier	Se trata de un número hexadecimal **de 16 bytes** que hace referencia a un **identificador exclusivo global (GUID)**. El GUID es especialmente útil cuando una fila debe ser única entre otras muchas. Por ejemplo, utilice el tipo de datos uniqueidentifier en una columna con números de identificación de los clientes para compilar una lista de clientes de una compañía en varios países. Para generar identificadores únicos debemos utilizar la función **NEWID()**.
sql_variant	El tipo de datos sql_variant permite que **una única columna**, parámetro o variable **almacene valores de datos de distintos tipos**. Cada instancia de una columna sql_variant registra el valor de los datos y los metadatos que describen el valor: su tipo de datos base, tamaño máximo, escala, precisión e intercalación.
timestamp	Un número único para toda la base de datos que se actualiza cada vez que se actualiza una fila. El tipo de datos timestamp de SQL Server no tiene nada que ver con horas o fechas. Los valores de tipo timestamp de SQL Server son números binarios que indican la secuencia relativa en la que se realizaron las modificaciones en una base de datos. El tipo de datos timestamp se desarrolló originalmente para admitir los algoritmos de recuperación de SQL Server. No use nunca columnas timestamp en claves, especialmente claves principales, porque el valor timestamp cambia cada vez que se modifica la fila.
table	Es un **tipo de datos especial** que se puede utilizar para almacenar un conjunto de resultados para su procesamiento posterior. Table se utiliza principalmente para el **almacenamiento temporal** de un conjunto de filas devuelto como el conjunto de resultados de una función con Para declarar variables de tipo table, se utiliza DECLARE @local_variable. un tipo de datos especial, que se utiliza para almacenar un conjunto de resultados para un proceso posterior (es como las tablas temporales) Este tipo de datos sólo se puede utilizar para definir variables locales de tipo table.
xml	El tipo de datos xml permite almacenar documentos y fragmentos XML en una base de datos de SQL Server. La representación almacenada de las instancias de tipo de datos xml no puede superar los 2 GB.

Ejemplo 1: Muestra el nombre del servidor en formato XML

```
DECLARE @myxml XML
set @myxml = (SELECT @@SERVERNAME NOMBRE FOR XML RAW, TYPE)
print cast(@myxml as varchar(max))
```

la salida es:

<row NOMBRE="EQ-DAI2-PROFE"/>

Ejemplo 2: de generación de un valor de identificación único para la variable myuniqueid.

```
DECLARE @myuniqueid UNIQUEIDENTIFIER
 set @myuniqueid = NEWID()
 print cast(@myuniqueid as varchar(36))
```

La salida es:
7812EE92-A66A-48F1-AAE0-963D82BA4DA4

5. CONSTANTES

- **Cadenas de caracteres:**

Van entre **comillas simples.** Si una cadena de caracteres entre comillas simples contiene una comilla, se representa la comilla interna con dos comillas simples (no es necesario en las cadenas incluidas entre comillas dobles). Ej: 'Luis'

Las **cadenas vacías** se representan como dos comillas simples sin nada entre ellas.

- **Cadenas Unicode**:

  Las cadenas Unicode tienen un formato similar al de las cadenas de caracteres, pero están precedidas por el **identificador N** que tiene que ir en mayúscula. Por ejemplo: N'Andrés'

- **Constantes de cadenas binarias**:

Las constantes de tipo binary tienen el **prefijo 0x** y son cadenas de números hexadecimales que no se incluyen entre comillas. Por ejemplo 0x12Ef.

- **Constantes bit:**

Se representan con los números 0 o 1, y si se utiliza un número mayor que uno, se convierte en uno. No se introducen entre comillas.

- **Constantes datetime:**

Se representan mediante valores de fecha en cadenas de caracteres en formatos concretos, incluidos entre comillas simples. Por ejemplo: '04/15/98', '04:24 PM'

- **Constantes integer:**

Se representan con una cadena de números no incluida entre comillas y que no contiene decimales. Por ejemplo 1894

- **Constantes decimal:**

Se representan con una cadena de números no enmarcados entre comillas y contienen un separador decimal. Por ejemplo: 1894.1204

- **Constantes float y real:**

  Se representan con notación científica. Por ejemplo: 101.5E5

- **Constantes money:**

Se representan con una cadena de números con un separador decimal y un símbolo de moneda opcionales como prefijo que no se incluyen entre comillas. Por ejemplo: $12

- **Constantes uniqueidentifier**:

Cadenas que representan un valor de identificador exclusivo global (GUID). Se pueden especificar en formato de cadena de caracteres o binario. Por ejemplo: 0xff19966f868b11d0b42d00c04fc964ff