Creación de tipos de datos y tablas

Contenido

Notas para el instructor

Este módulo proporciona a los alumnos una descripción de cómo crear tipos de datos y tablas, así como del modo de generar secuencias de comandos Transact-SQL que contienen instrucciones para crear las bases de datos y sus objetos.

Después de completar este módulo, los alumnos serán capaces de:

- Crear y eliminar tipos de datos definidos por el usuario.
- Crear y eliminar tablas del usuario.
- Generar valores de columnas
- Generar una secuencia de comandos.

Introducción

Objetivos de la diapositiva
Proporcionar una introducción a los temas y objetivos del módulo.

Explicación previa
En este módulo, aprenderá acerca de cómo crear tipos de datos y tablas y generar secuencias de comandos.

- Creación de los tipos de datos
- Creación de tablas
- Generación de valores de columnas
- Generación de secuencias de comandos

Este módulo describe cómo crear tipos de datos y tablas, así como el modo de generar secuencias de comandos Transact-SQL que contienen instrucciones para crear las bases de datos y sus objetos.

Después de completar este módulo, los alumnos serán capaces de:

- Crear y eliminar tipos de datos definidos por el usuario.
- Crear y eliminar tablas del usuario.
- Generar valores de columnas
- Generar secuencias de comandos.

Creación de los tipos de datos

Objetivo del tema
Proporcionar un resumen de este tema.

Explicación previa
En esta sección aprenderá cómo crear y eliminar tipos de datos definidos por el usuario.

- Tipos de datos del sistema
- Creación y eliminación de tipos de datos definidos por el usuario
- Directrices para especificar tipos de datos

Antes de crear una tabla, debe definir los tipos de datos de la tabla. Los tipos de datos especifican el tipo de información (caracteres, números o fechas) que se puede almacenar en una columna, así como la forma en que los datos están almacenados. Microsoft® SQL Server™ 2000 proporciona varios tipos de datos del sistema. SQL Server también permite tipos de datos definidos por el usuario que estén basados en los tipos de datos del sistema.

Tipos de datos del sistema

Objetivo del tema
Presentar los tipos de datos de SQL Server.

Explicación previa
SQL Server proporciona varios tipos de datos diferentes.

- Numérico
  - Entero
  - Numérico exacto
  - Numérico aproximado
  - Moneda
- Fecha y hora
- Carácter y caracteres Unicode
- Binario
- Otros

Los tipos de datos definen el valor de datos que se permite en cada columna. SQL Server proporciona varios tipos de datos diferentes. Ciertos tipos de datos comunes tienen varios tipos de datos de SQL Server asociados. Debe elegir los tipos de datos adecuados que le permitan optimizar el rendimiento y conservar espacio en el disco.

Categorías de tipos de datos del sistema

La siguiente tabla asocia los tipos de datos comunes con los tipos de datos del sistema proporcionados por SQL Server. La tabla incluye los sinónimos de los tipos de datos por compatibilidad con ANSI.

Tipos de datos comunes	Tipos de datos del sistema de SQL Server.	Sinónimo ANSI	Número de bytes
Entero	**int**	*integer*	4
	bigint	–	8
	smallint, **tinyint**	–	2, 1
Numérico exacto	**decimal[(p[, s])]**	*dec*	2–17
	numeric[(p[, s])]	–	
Numérico aproximado	**float[(n)]**	*double precision*, *float*[(*n*)] para *n=8-15*	8
	real	*float*[(*n*)] para *n=1-7*	4
Moneda	**money**, **smallmoney**	–	8, 4
Fecha y hora	**Datetime**, **smalldatetime**	–	8 4

Sugerencia
SQL Server puede aceptar varios idiomas al almacenar cadenas de texto en campos del tipo de datos Unicode (doble byte).

(*continuación*)

Tipos de datos comunes	Tipos de datos del sistema de SQL Server.	Sinónimo ANSI	Número de bytes
Carácter	**char[(n)]**	*character*[(*n*)]	0–8000
	varchar[(n)]	*char VARYING*[(*n*)] *character VARYING*[(*n*)]	
	text	–	0 a 2 GB
Caracteres Unicode	**nchar[(n)]**	–	0–8000
	nvarchar[(n)]		(4000 caracteres)
	ntext		0 a 2 GB
Binario	**binary[(n)]**	–	0–8000
	varbinary[(n)]	*binary VARYING*[(*n*)]	
Imagen	**image**	–	0 a 2 GB
Identificador global	**uniqueidentifier**	–	16
Especial	**bit**, **cursor**, **uniqueidentifier**	–	1, 0–8
	timestamp	**rowversion**	8
	sysname	–	256
	table	–	
	sql_variant	–	0–8016

Sugerencia
Destaque los tipos de datos cursor, table y sql_variant.

Tipos de datos numéricos exactos y aproximados

La elección del tipo de dato, numérico exacto o numérico aproximado, dependerá del uso que le vaya a dar.

Tipos de datos numéricos exactos

Los tipos de datos numéricos exactos permiten especificar *exactamente* la escala y la precisión que se va a utilizar. Por ejemplo, puede especificar tres cifras a la derecha de la coma decimal y cuatro a la izquierda. Las consultas devuelven siempre los datos introducidos. SQL Server acepta dos tipos de datos numéricos exactos por compatibilidad con ANSI: **decimal** y **numeric**.

En general, los tipos de datos numéricos exactos se deben utilizar en aplicaciones financieras en las que se quiere que los datos se reflejen de forma coherente (siempre con dos decimales) y en las consultas sobre dichas columnas (por ejemplo, para buscar todos los préstamos con un interés del 8,75 por ciento).

Tipos de datos numéricos aproximados

Los tipos de datos numéricos aproximados almacenan datos de la forma más precisa posible. Por ejemplo, la fracción 1/3 se representa en un sistema decimal como 0,33333 (periódico). El número no se puede almacenar con precisión, de modo que se almacena una aproximación. SQL Server acepta dos tipos de datos aproximados: **float** y **real**.

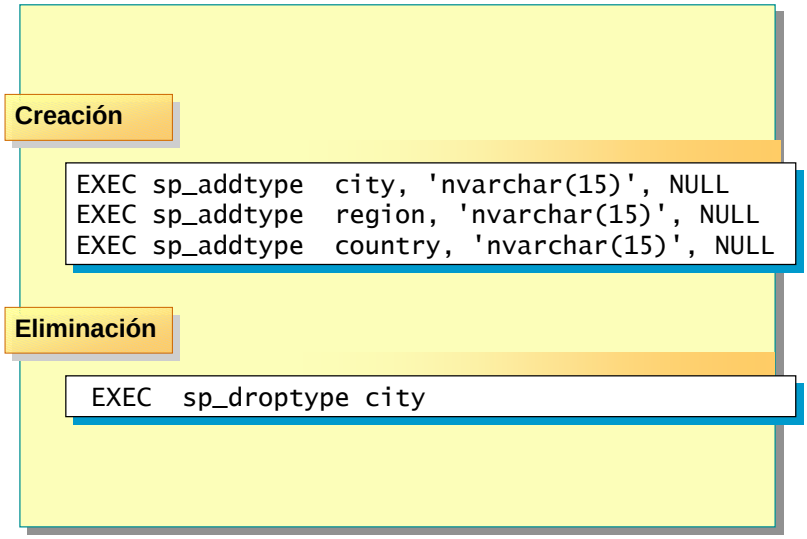
Si redondea números o realiza comprobaciones de calidad entre valores, debe evitar el uso de tipos de datos numéricos aproximados.

Nota Lo mejor es no hacer referencia a columnas del tipo de datos **float** o **real** en las cláusulas WHERE.

Creación y eliminación de tipos de datos definidos por el usuario

Objetivo del tema
Explicar cómo se agregan y se eliminan tipos de datos definidos por el usuario.

Explicación previa
Puede crear y eliminar tipos de datos definidos por el usuario mediante el Administrador corporativo de SQL o los procedimientos almacenados del sistema.



Sugerencia
Demuestre la creación de un tipo de datos con el Administrador corporativo de SQL Server.

Los tipos de datos definidos por el usuario están basados en los tipos de datos del sistema. Permiten afinar los tipos de datos para asegurar la coherencia cuando se trabaja con elementos de datos comunes en tablas o bases de datos diferentes. Un tipo de datos definido por el usuario se define para una base de datos específica.

Nota Los tipos de datos definidos por el usuario que se crean en la base de datos **model** se incluyen automáticamente en todas las bases de datos que se crean a continuación. Todos los tipos de datos definidos por el usuario se agregan como filas de la tabla **systypes**.

Puede crear y eliminar tipos de datos definidos por el usuario mediante el Administrador corporativo de SQL o procedimientos almacenados del sistema. Los nombres de los tipos de datos deben seguir las reglas de los nombres de identificadores y tienen que ser únicos en cada base de datos. Defina los tipos de datos definidos por el usuario en términos de tipos de datos del sistema, preferiblemente especificando NULL o NOT NULL.

Creación de un tipo de datos definido por el usuario

El procedimiento almacenado del sistema **sp_addtype** crea tipos de datos definidos por el usuario.

Sintaxis

sp_addtype {*tipo*}**,** [*tipoDeDatosDelSistema*] [, ['NULL' | 'NOT NULL']] [, '*nombrePropietario*']

Ejemplo

El siguiente ejemplo crea tres tipos de datos definidos por el usuario.

```
EXEC  sp_addtype  city, 'nvarchar(15)', NULL
EXEC  sp_addtype  region, 'nvarchar(15)', NULL
EXEC  sp_addtype  country, 'nvarchar(15)', NULL
```

Eliminación de un tipo de datos definido por el usuario

El procedimiento almacenado del sistema **sp_droptype** elimina tipos de datos definidos por el usuario en la tabla del sistema **systypes**. No se puede eliminar un tipo de datos definido por el usuario si hay referencias al mismo en tablas u otros objetos de la base de datos.

Sintaxis

sp_droptype {‘*tipo’*}

Ejemplo

El siguiente ejemplo elimina un tipo de datos definido por el usuario.

```
EXEC sp_droptype city
```

Nota Ejecute el procedimiento almacenado del sistema **sp_help** para obtener la lista de los tipos de datos actualmente definidos.

Directrices para especificar tipos de datos

Objetivo del tema
Presentar algunas directrices para seleccionar tipos de datos.

Explicación previa
Al seleccionar tipos de datos, haga un balance entre el tamaño de almacenamiento y los requisitos.

- Si la longitud de la columna varía, utilice uno de los tipos de datos variables
- Use tinyint adecuadamente
- Para tipos de datos numéricos, use los decimales más frecuentes
- Si el almacenamiento es superior a 8000 bytes, utilice text o image
- Para la moneda utilice el tipo de datos money
- No utilice float y real como claves principales

Para seleccionar tipos de datos y equilibrar el tamaño de almacenamiento con los requisitos, tenga en cuenta las directrices siguientes:

- Si la longitud de la columna varía, utilice uno de los tipos de datos variables. Por ejemplo, si tiene una lista de nombres, puede definirlos como **varchar** en lugar de **char** (fijo).
- Si es dueño de un próspero negocio de venta de libros con filiales en muchos lugares y ha especificado el tipo de datos **tinyint** para el identificador de cada librería en la base de datos, puede tener problemas cuando decida abrir la librería número 256.
- Para los tipos de datos numéricos, el tamaño y el nivel de precisión requeridos determinan su elección. En general, utilice **decimal**.
- Si el almacenamiento es superior a 8000 bytes, utilice **text** o **image**. Si es inferior a 8000, utilice **binary** o **char**. Cuando sea posible, lo mejor es utilizar **varchar** porque tiene mayor funcionalidad que **text** e **image**.
- Para la moneda utilice el tipo de datos **money**.
- No utilice los tipos de datos aproximados **float** y **real** como claves principales. Debido a que los valores de estos tipos de datos no son exactos, no es adecuado utilizarlos en comparaciones.

Creación de tablas

Objetivo del tema
Proporcionar un resumen de este tema.

Explicación previa
En esta sección, aprenderá cómo definir todos los tipos de datos de una tabla, así como el modo de crear tablas, crear y eliminar columnas y generar valores de columnas.

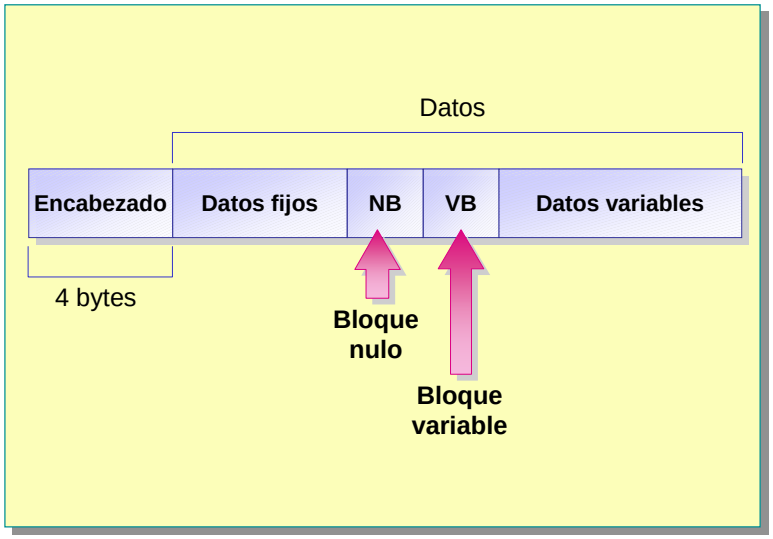
- Cómo SQL Server organiza los datos en filas
- Cómo SQL Server organiza los datos text, ntext e image
- Creación y eliminación de una tabla
- Agregar y quitar columnas

Después de definir todos los tipos de datos de la base de datos, puede crear tablas, agregar y quitar columnas, y generar los valores de las columnas.

Cómo SQL Server organiza los datos en filas

Objetivo del tema
Describir cómo los datos se organizan en filas.

Explicación previa
Para estimar con precisión el tamaño de una tabla es importante comprender los elementos de la parte de datos de cada fila.



Una fila de datos se compone de un encabezado de fila y una parte de datos. Para estimar con precisión el tamaño de una tabla es importante comprender los elementos de la parte de datos de cada fila.

Encabezado de fila

El encabezado de fila de 4 bytes contiene información acerca de las columnas de la fila de datos, por ejemplo, un puntero que indica la ubicación del final de la parte de datos fijos de la fila y si existen columnas de longitud variable en la fila.

Parte de datos

La parte de datos de una fila puede contener los siguientes elementos:

- *Datos de longitud fija*. Los datos de longitud fija se introducen en la página antes que los datos de longitud variable. Una fila de datos de longitud fija que esté vacía ocupa tanto espacio como una que esté llena. Una tabla que sólo tenga columnas de longitud fija almacena el mismo número de filas en una página.
- *Bloque nulo*. Un bloque nulo es un conjunto de bytes de longitud variable. Se compone de dos bytes que almacenan el número de columnas seguido de un mapa de bits nulo que indica si cada columna individual es nula. El tamaño de un mapa de bits nulo es igual a un bit por columna, redondeado al byte más próximo. Las columnas de la una a la ocho requieren un mapa de bits de 1 byte. Las columnas de la nueve a la dieciséis requieren un mapa de bits de 2 bytes.

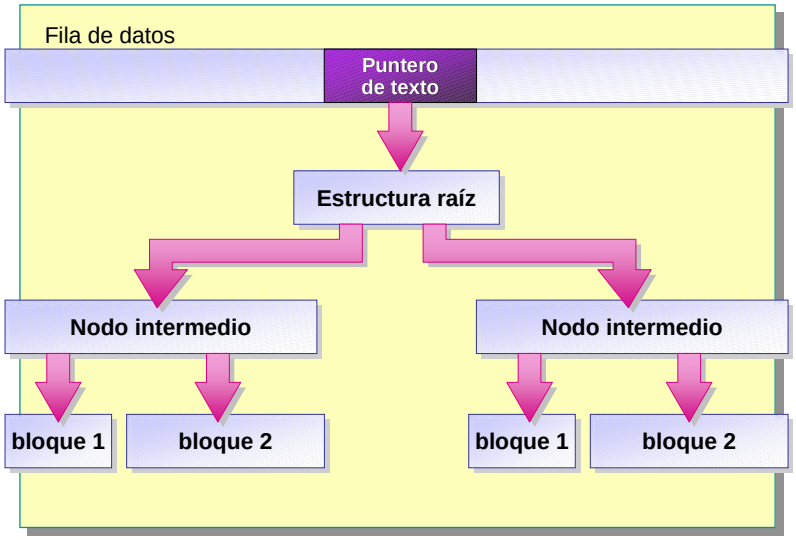
- *Bloque variable*. Un bloque variable se compone de dos bytes que describen el número de columnas de longitud variable que están presentes. Dos bytes adicionales por columna señalan el final de cada columna de longitud variable. Si no hay ninguna columna de longitud variable, el bloque variable se omite.
- *Datos de longitud variable*. Los datos de longitud variable se introducen en la página después del bloque variable. Una fila de datos de longitud variable que esté vacía no ocupa espacio. Una tabla con columnas de longitud variable puede tener unas cuantas filas largas o muchas filas cortas.

Sugerencia Cuando sea posible, mantenga la longitud de las filas compacta para permitir que quepan más filas en una página. Esto reduce la entrada/salida (E/S) y mejora la proporción de aciertos de caché del búfer.

Cómo SQL Server organiza los datos text, ntext e image

Objetivo del tema
Demostrar que existen tres tipos de datos que son una excepción a la organización en filas.

Explicación previa
Los tipos de datos de longitud variable se almacenan habitualmente como una colección de páginas y no en filas de datos.



Los tipos de datos de longitud variable se pueden almacenar como una colección de páginas o en filas de datos. Estos tipos de datos son los siguientes:

- **text**, que puede contener 2.147.483.647 caracteres. El tipo de datos **text** que no sea Unicode no se puede utilizar en variables o parámetros de procedimientos almacenados.
- **ntext**, que puede contener un máximo de 2^{30} -1 (1.073.741.823) caracteres o 2^{31} -1 bytes, que equivale a 2.147.483.647 bytes de datos Unicode de longitud variable. El sinónimo de SQL-92 para **ntext** es texto nacional.
- **image**, que puede contener de 0 a 2.147.483.647 bytes de datos binarios.

Debido a que los tipos de datos **text**, **ntext** e **image** suelen ser de gran tamaño, SQL Server los almacena fuera de las filas. Un puntero de 16 bytes en la fila de datos apunta a una estructura raíz que contiene los datos. La estructura raíz de text constituye el nodo raíz del árbol B, que apunta a los bloques de datos. Si hay más de 32 kilobytes (KB) de datos, los nodos intermedios del árbol B se agregan entre el nodo raíz y los bloques de datos. Esto permite una rápida navegación del árbol B que comienza en mitad de una cadena.

Con contenido **text**, **ntext** e **image** de tamaño pequeño a mediano, SQL Server ofrece la opción de almacenar valores en la fila de datos en lugar de en una estructura independiente del árbol B. Puede especificar esta opción **text in row**. También puede determinar el límite de la opción; el intervalo va de 24 a 7.000 bytes.

Puede habilitar la opción **text in row** para una tabla mediante el procedimiento almacenado del sistema **sp_tableoption.**

Ejemplo

En este ejemplo se activa la opción **text in row** del procedimiento almacenado del sistema **sp_tableoption** y se especifica que hasta 1000 caracteres **text**, **ntext** o **image** se almacenarán en la página de datos.

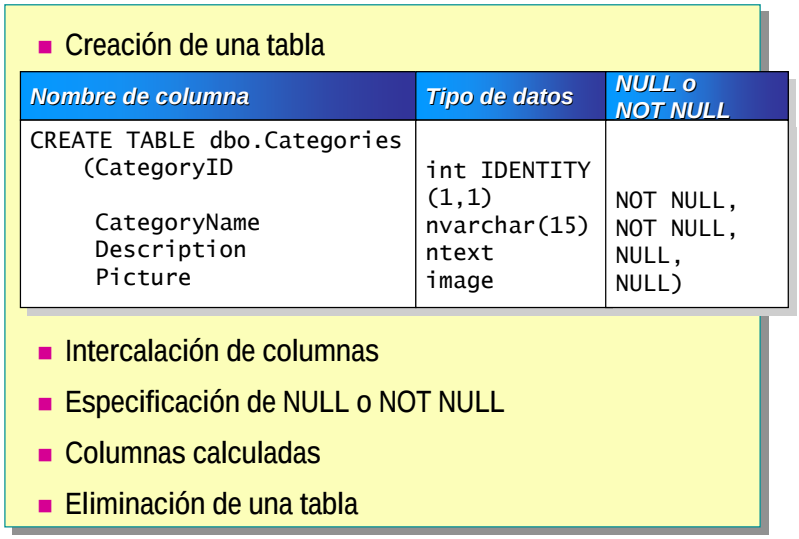
```
EXEC sp_tableoption N'Employees', 'text in row', '1000'
```

Nota Si se activa, pero no se especifica un valor, el valor predeterminado es 256 bytes. El valor predeterminado garantiza que los valores pequeños y los punteros de texto se puedan almacenar en filas de datos.

Creación y eliminación de una tabla

Objetivo del tema
Explicar cómo se crean y eliminan las tablas.

Explicación previa
Cuando se crea una tabla, hay que especificar el nombre de la tabla, los nombres de las columnas y sus tipos de datos.



Sugerencia
Demuestre la creación de una tabla con el Administrador corporativo de SQL Server.

Cuando se crea una tabla, hay que especificar el nombre de la tabla, los nombres de las columnas y los tipos de datos de las columnas. Los nombres de las columnas tienen que ser únicos en una tabla específica, pero se puede utilizar el mismo nombre de columna en tablas diferentes de la misma base de datos. Hay que especificar un tipo de datos para cada columna.

Creación de una tabla

Al crear tablas en SQL Server, debe tener en cuenta los hechos siguientes. Puede tener hasta:

- Dos mil millones de tablas en cada base de datos.
- 1.024 columnas por cada tabla.
- 8060 bytes por fila (esta longitud máxima aproximada no se aplica a los tipos de datos **image**, **text** y **ntext**).

Intercalación de columnas

SQL Server acepta el almacenamiento de objetos con diferentes intercalaciones en la misma base de datos. Es posible especificar intercalaciones de SQL Server independientes en el nivel de columna, de modo que a cada columna de una tabla se le puede asignar una intercalación distinta.

Especificación de NULL o NOT NULL

En la definición de una tabla, se puede especificar si se admiten valores NULL en cada una de sus columnas. Si no especifica NULL o NOT NULL, SQL Server proporciona la característica NULL o NOT NULL en función del nivel predeterminado de la sesión o la base de datos. Sin embargo, este nivel predeterminado puede variar, de modo que no confíe en él. NOT NULL es el valor predeterminado en SQL Server.

Sintaxis parcial

CREATE TABLE *nombreTabla*
nombreColumna tipoDatos [COLLATE<*nombreIntercalación*>]
[NULL | NOT NULL]
| *nombreColumna* AS *expresiónColumnaCalculada*
[,….*n*]

Ejemplo

El siguiente ejemplo crea la tabla **dbo.CategoriesNew** y se especifican las columnas de la tabla, el tipo de datos de cada columna y si cada una de las columnas admite valores NULL.

```
CREATE TABLE dbo.CategoriesNew
   (CategoryID     int IDENTITY
                   (1, 1)           NOT NULL,
   CategoryName    nvarchar(15)     NOT NULL,
   Descripción     ntext            NULL,
   Picture         image            NULL)
```

Nota Para ver las propiedades de una tabla, haga clic con el botón secundario del *mouse* (ratón) en una tabla en el Administrador corporativo de SQL Server o ejecute el procedimiento almacenado del sistema **sp_help** y, después, desplácese a la derecha.

Columnas calculadas

Una *columna calculada* es una columna virtual que no está físicamente almacenada en la tabla. SQL Server utiliza una fórmula que crea el usuario para calcular este valor de columna a partir de otras columnas de la misma tabla. El uso de un nombre de columna calculada en una consulta puede simplificar la sintaxis de la consulta.

Eliminación de una tabla

La eliminación de una tabla elimina la definición de la tabla y todos sus datos, así como las especificaciones de permisos sobre dicha tabla.

Antes de eliminar una tabla, debe quitar todas las dependencias entre la tabla y otros objetos. Para ver las dependencias existentes, ejecute el procedimiento almacenado del sistema **sp_depends**.

Sintaxis

DROP TABLE *nombreTabla* [,…*n*]

Ejemplo 2

En este ejemplo se utiliza un comentario de línea para impedir la ejecución de una sección de una instrucción.

```
USE northwind
SELECT productname
  , (unitsinstock - unitsonorder) -- Calcula el inventario
-- , supplierid
FROM products
GO
```

Comentarios de bloque

Para crear bloques de varias líneas de comentarios, coloque un carácter de comentario (/*) al comienzo del texto del comentario, escriba sus anotaciones y, después, concluya el comentario con un carácter de cierre de comentario (*/).

Utilice este indicador de carácter para crear una o varias líneas de comentarios o encabezados de comentarios (texto descriptivo que documenta las instrucciones que le siguen). A menudo, los encabezados de comentario incluyen el nombre del autor, la fecha de creación y de la última modificación de la secuencia de comandos, información de la versión y una descripción de la acción que realiza la instrucción.

Ejemplo 3

En este ejemplo se muestra un encabezado de comentario que abarca varias líneas.

```
/*
 Este código devuelve todas las filas de la tabla
 products y muestra el precio por unidad, el precio
 aumentado en un 10 por ciento y el nombre del producto.
*/
USE northwind
SELECT unitprice, (unitprice * 1.1), productname
 FROM products
GO
```

Nota Los comentarios deben colocarse en toda la secuencia de comandos para describir las acciones que están realizando las instrucciones. Esto es especialmente importante si otros usuarios deben revisar o implementar la secuencia de comandos.

Ejemplo 4

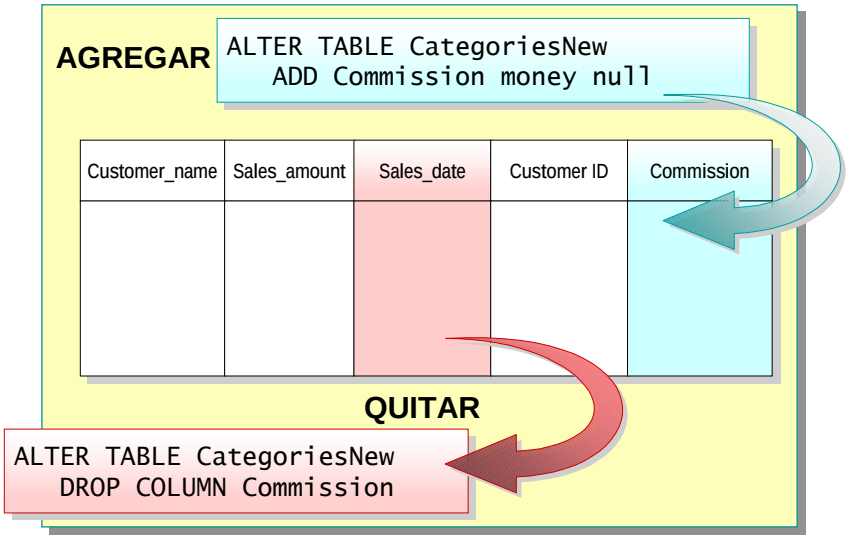
Esta sección de una secuencia de comandos está comentada para evitar que se ejecute. Esto puede resultar útil cuando se depura o se solucionan problemas de un archivo de comandos.

```
/*
DECLARE @v1 int
SET @v1 = 0
WHILE @v1 < 100
   BEGIN
    SELECT @v1 = (@v1 + 1)
    SELECT @v1
   END
*/
```

Agregar y quitar columnas

Objetivo del tema
Mostrar cómo se agregan y se quitan columnas.

Explicación previa
A medida que los requisitos cambian, puede que necesite modificar las tablas agregando o quitando columnas.



Agregar o quitar columnas son dos maneras de modificar las tablas.

Sintaxis parcial

```
ALTER TABLE tabla
    {| [ALTER COLUMN columna ]
        |{   ADD
                {    <definiciónColumna> ::=
                        columna tipoDeDatos
                        {    [NULL | NOT NULL]
            | DROP COLUMN columna } [,…n]
```

Agregar una columna

El tipo de información que se especifica cuando se agrega una columna es similar al que se proporciona cuando se crea una tabla.

Ejemplo

Este ejemplo agrega una columna que admite valores NULL.

```
ALTER TABLE CategoriesNew
 ADD Commission money null
```

Quitar una columna

Las columnas que se quitan son irrecuperables. Por lo tanto, asegúrese de que desea eliminar la columna antes de hacerlo.

Ejemplo

Este ejemplo quita una columna de una tabla.

```
ALTER TABLE CategoriesNew
 DROP COLUMN Commission
```

Nota Antes de quitar una columna, debe eliminar todos los índices y las restricciones que se basen en esa columna.

Generación de valores de columnas

Objetivo del tema
Presentar los temas de esta sección.

Explicación previa
Esta sección describe cómo generar valores de columna.

- Uso de la propiedad Identity
- Uso de la función NEWID y el tipo de datos uniqueidentifier

Varias características le permiten generar valores de columnas: la propiedad **Identity**, la función NEWID y el tipo de datos **uniqueidentifier**.

Uso de la propiedad Identity

Objetivo del tema
Explicar cómo utilizar la propiedad Identity.

Explicación previa
Las columnas de identidad se suelen utilizar para los valores de las claves principales.

- Requisitos para utilizar la propiedad Identity
  - Sólo se permite una columna de identidad por tabla
  - Utilizar con tipos de datos integer, numeric y decimal
- Recuperar información acerca de la propiedad Identity
  - Utilizar IDENT_SEED e IDENT_INCR para información de definición
  - Utilizar @@identity para determinar el valor más reciente
- Administrar la propiedad Identity

Sugerencia
Si no utiliza el incremento automático, debe efectuar una consulta para buscar el valor siguiente.

Puede utilizar la propiedad **Identity** para crear columnas (denominadas columnas de identidad) que contengan valores secuenciales generados por el sistema que identifican cada fila insertada en una tabla. Las columnas de identidad se suelen utilizar para los valores de las claves principales.

Hacer que SQL Server proporcione automáticamente los valores de las claves puede reducir costos y aumentar el rendimiento. Simplifica la programación, mantiene valores bajos en las claves principales y reduce los cuellos de botella en las transacciones de los usuarios.

Sintaxis parcial

```
CREATE TABLE tabla
  (columna tipoDeDatos
   [ IDENTITY [(inicial, incremento)]] NOT NULL )
```

A la hora de utilizar la propiedad **Identity**, considere los siguientes requisitos:

- Sólo se permite una columna de identidad por tabla.
- Se debe utilizar con tipos de datos **integer** (**int**, **bigint**, **smallint** o **tinyint**), **numeric** o **decimal**. Los tipos de datos **numeric** y **decimal** se deben especificar con una escala de 0.
- No se puede actualizar.
- En una consulta, puede utilizar la palabra clave IDENTITYCOL en lugar del nombre de columna. Esto le permite hacer referencia a la columna de la tabla que tiene la propiedad **Identity** sin necesidad de saber el nombre de la columna.
- No permite valores NULL.

Cuando determine qué tipo de datos va a utilizar para definir una columna, mediante el uso de la propiedad Identity, intente calcular el número de filas que va a contener la tabla.

La información acerca de la propiedad **Identity** puede recuperarse de varias maneras:

- Dos funciones del sistema devuelven información acerca de la definición de las columnas de identidad: IDENT_SEED (devuelve el valor inicial) e IDENT_INCR (devuelve el valor del incremento).
- Puede obtener datos de las columnas de identidad mediante la variable global **@@identity**, que determina el valor de la última fila insertada en una columna de identidad durante una sesión.
- SCOPE_IDENTITY devuelve el último valor de IDENTITY insertado en una columna de identidad del mismo ámbito. Un ámbito es un procedimiento almacenado, un desencadenador, una función o un proceso por lotes.
- IDENT_CURRENT devuelve el último valor de identidad generado para una tabla específica en cualquier sesión y en cualquier ámbito.

La propiedad **Identity** puede administrarse de varias maneras:

- Puede permitir que se inserten valores explícitos en la columna de identidad de una tabla si establece la opción IDENTITY_INSERT en ON. Cuando IDENTITY_INSERT está activada, las instrucciones INSERT deben suministrar un valor.
- Para comprobar y, posiblemente, corregir el valor de identidad actual de una tabla, puede utilizar la instrucción DBCC CHECKIDENT. DBCC CHECKIDENT le permite comparar el valor de identidad actual con el valor máximo de la columna de identidad.

Nota La propiedad **Identity** no exige la unicidad. Para exigirla, cree un índice único.

Ejemplo

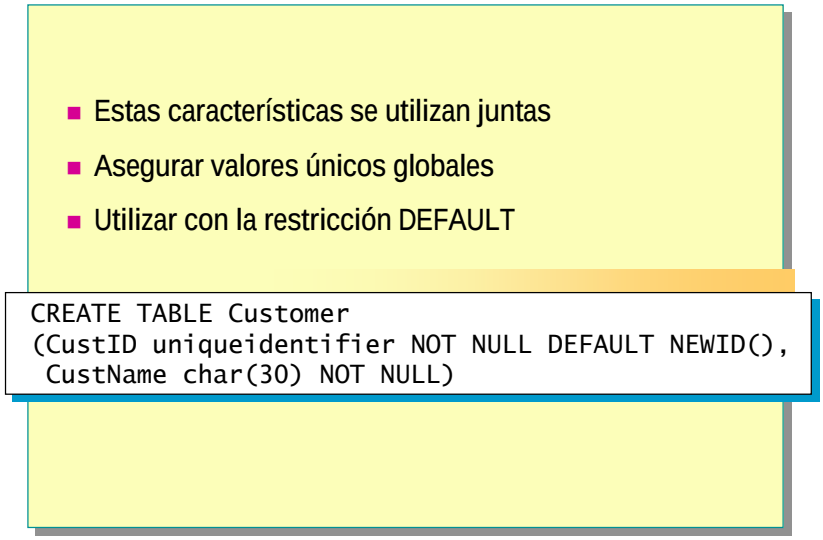
Este ejemplo crea una tabla con dos columnas, **StudentId** y **Name**. La propiedad **Identity** se utiliza para incrementar automáticamente el valor de la columna **StudentId** por cada fila agregada. El valor inicial se establece en 100, y el valor de incremento es 5. Los valores de la columna serían 100, 105, 110, 115, etc. El uso de 5 como valor de incremento permite insertar registros entre los valores posteriormente.

```
CREATE TABLE Class
  (StudentID int  IDENTITY(100, 5) NOT NULL,
  Name varchar(16))
```

Uso de la función NEWID y el tipo de datos uniqueidentifier

Objetivo del tema
Describir cómo utilizar estas características.

Explicación previa
El tipo de datos **uniqueidentifier** y la función NEWID son dos características que se utilizan juntas.



El tipo de datos **uniqueidentifier** y la función NEWID son dos características que se utilizan juntas. Utilícelas cuando los datos provengan de muchas tablas para crear una tabla mayor y sea necesario mantener la unicidad entre todos los registros:

- El tipo de datos **uniqueidentifier** almacena un número de identificación único como cadena binaria de 16 bytes. Este tipo de datos se utiliza para almacenar un identificador global único (GUID, *Globally Unique Identifier*).
- La función NEWID crea un número identificador único que puede almacenar un GUID mediante el tipo de datos **uniqueidentifier**.
- El tipo de dato **uniqueidentifier** no genera automáticamente nuevos ID para las filas insertadas en la forma en que lo hace la propiedad **Identity**. Para obtener nuevos valores de **uniqueidentifier**, debe definir una tabla con una restricción DEFAULT que especifique la función NEWID. Si utiliza una instrucción INSERT, debe especificar también la función NEWID.

Ejemplo

En este ejemplo, la columna Id. de cliente (**custID**) de la tabla **Customer** se crea con el tipo de datos **uniqueidentifier**, con un valor predeterminado generado por la función NEWID. Se generará un valor único para la columna **CustID** en cada nueva fila y en las filas existentes.

```
CREATE TABLE Customer
(CustID uniqueidentifier NOT NULL DEFAULT NEWID(),
  CustName char(30) NOT NULL)
```

Generación de secuencias de comandos

Objetivo del tema
Describir cómo se genera un archivo de comandos.

Explicación previa
Cuando se crean objetos en una base de datos, es importante guardar todas las definiciones de los objetos en un archivo de comandos.

- Generación del esquema como una secuencia de comandos de Transact-SQL
  - Mantener una copia de seguridad
  - Crear o actualizar una secuencia de comandos de desarrollo de la base de datos
  - Crear una prueba o un entorno de desarrollo
  - Formar empleados recién contratados
- Qué se genera
  - Toda la base de datos en un sólo archivo de comandos
  - Sólo el esquema de las tablas
  - El esquema de las tablas y los índices

Sugerencia
Demuestre la generación de una secuencia de comandos mediante la base de datos Northwind.

Cuando se crean objetos en una base de datos, es importante guardar todas las definiciones de los objetos en un archivo de comandos.

Generación del esquema como una secuencia de comandos Transact-SQL

Puede utilizar el Administrador corporativo de SQL Server para documentar la estructura de una base de datos existente (esquema); para ello, genérelo como una o varias secuencias de comandos Transact-SQL. Estas secuencias de comandos Transact-SQL contienen las descripciones de las instrucciones que se han utilizado para crear una base de datos y sus objetos.

El esquema generado como secuencia de comandos Transact-SQL se puede utilizar para:

- Mantener una copia de seguridad que permita que el usuario pueda volver a crear todos los usuarios, grupos, inicios de sesión y permisos.
- Crear o actualizar una secuencia de comandos de desarrollo de la base de datos.
- Crear una prueba o un entorno de desarrollo a partir de un esquema existente.
- Formar empleados recién contratados.

Qué se genera

Puede generar:

- Toda la base de datos en un solo archivo de comandos.
- Sólo el esquema de las tablas para una, varias o todas las tablas de la base de datos en uno o varios archivos de comandos.
- El esquema de las tablas y los índices en un archivo de comandos, los procedimientos almacenados en otro archivo y los valores predeterminados y las reglas en otro.