Introducción a Transact-SQL

Contenido

Notas para el instructor

Transact-SQL es un lenguaje que sirve para la definición, tratamiento y control de datos. Este módulo proporciona una breve información general de Transact-SQL como lenguaje de programación. También describe los distintos tipos de instrucciones de Transact-SQL y los elementos de su sintaxis.

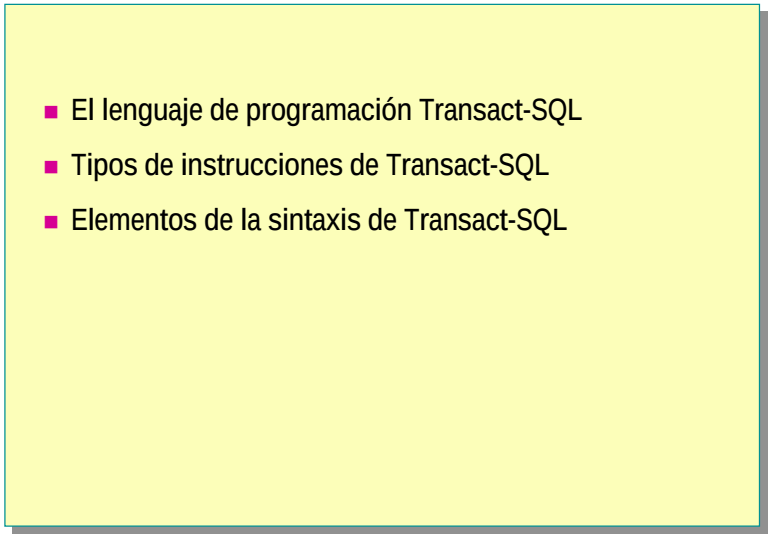
Al final de este módulo, los alumnos serán capaces de:

- Describir el lenguaje de programación Transact-SQL.
- Describir los tipos de instrucciones de Transact-SQL.
- Describir los elementos de la sintaxis de Transact-SQL.

Introducción

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

Explicación previa
En este módulo aprenderá acerca de Transact-SQL.



Transact-SQL es un lenguaje que sirve para la definición, tratamiento y control de datos. Este módulo proporciona una breve información general de Transact-SQL como lenguaje de programación. También describe los distintos tipos de instrucciones de Transact-SQL y los elementos de su sintaxis.

Al terminar este módulo, el alumno será capaz de:

- Describir el lenguaje de programación Transact-SQL.
- Describir los tipos de instrucciones de Transact-SQL.
- Describir los elementos de la sintaxis de Transact-SQL.

El lenguaje de programación Transact-SQL

Objetivos de la diapositiva
Presentar el hecho de que SQL Server tiene su propio lenguaje de programación.

Explicación previa
Transact-SQL es una versión del lenguaje de programación SQL que se utiliza exclusivamente en SQL Server.

- Implementa el estándar ISO del nivel básico de la especificación ANSI SQL-92
- Se pueden ejecutar desde cualquier producto que cumpla los requisitos básicos
- Incluye una funcionalidad ampliada

Los organismos ANSI (*American National Standards Institute*) e ISO (*International Standards Organization*) han definido estándares para SQL. Mediante Transact-SQL, Microsoft® SQL Server™ 2000 admite el nivel básico de implementación de SQL-92, el estándar SQL publicado por ANSI e ISO en 1992. Los elementos del lenguaje Transact-SQL que cumplen los requisitos de ANSI-SQL se pueden ejecutar desde cualquier producto que cumpla los requisitos básicos de ANSI-SQL. Transact-SQL incluye, además, varias extensiones que proporcionan una funcionalidad ampliada.

Tipos de instrucciones de Transact-SQL

Objetivos de la diapositiva
Describir los tipos básicos de instrucciones de Transact-SQL.

Explicación previa
Cuando escriba y ejecute instrucciones de Transact-SQL, utilizará algunos de estos tipos.

- Instrucciones del Lenguaje de definición de datos
- Instrucciones del Lenguaje de control de datos
- Instrucciones del Lenguaje de tratamiento de datos

Sugerencia
Resalte que este curso se centra especialmente en el uso de las instrucciones de Lenguaje de tratamiento de datos (DML)

Una *consulta* es una petición que se hace para obtener datos almacenados en SQL Server. Todas las consultas presentan al usuario el conjunto de resultados de una instrucción SELECT. Un *conjunto de resultados* es una tabla que muestra los datos obtenidos mediante la instrucción SELECT. La tabla tiene filas y columnas.

La escritura y ejecución de instrucciones de Transact-SQL es una de las formas en que se puede realizar una consulta en SQL Server. Cuando escriba y ejecute instrucciones de Transact-SQL, utilizará:

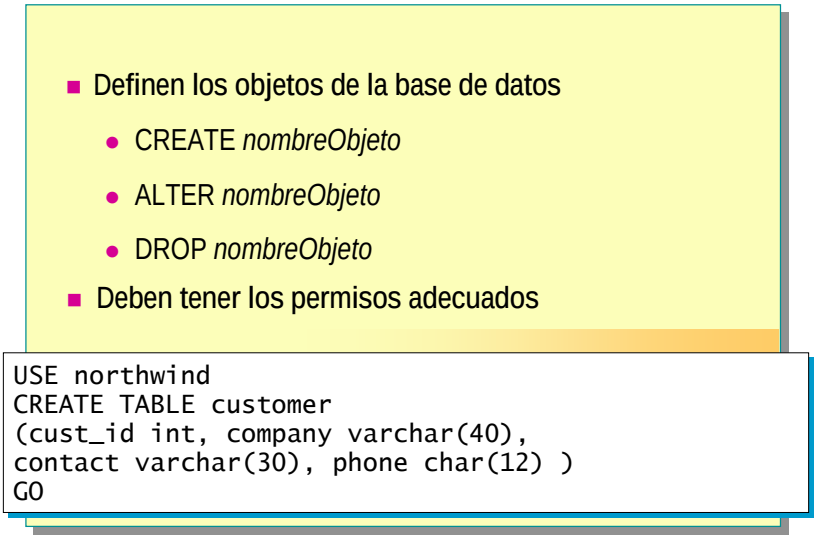
- Instrucciones del Lenguaje de definición de datos (DDL), que se utilizan para crear objetos en la base de datos.
- Instrucciones del Lenguaje de control de datos (DCL), que se utilizan para determinar quién puede ver o modificar los datos.
- Instrucciones del Lenguaje de tratamiento de datos (DML), que se utilizan para consultar y modificar los datos.

Nota Este curso se centra, principalmente, en el uso de instrucciones DML para consultar datos de SQL Server.

Instrucciones del Lenguaje de definición de datos

Objetivos de la diapositiva
Presentar a los alumnos las instrucciones de DDL.

Explicación previa
Las instrucciones de DDL definen una base de datos mediante la creación de bases de datos, tablas y tipos de datos definidos por el usuario.



Para su información
Para obtener más información acerca de las instrucciones DDL, consulte el curso 2329A, *Programación de una base de datos Microsoft SQL Server 2000*.

Las instrucciones de DDL definen la base de datos mediante la creación de bases de datos, tablas y tipos de datos definidos por el usuario. Las instrucciones de DDL se utilizan también para administrar los objetos de la base de datos. Algunas instrucciones de DDL son:

- CREATE *nombreObjeto*
- ALTER *nombreObjeto*
- DROP *nombreObjeto*

De forma predeterminada, sólo los miembros de la función **sysadmin**, **dbcreator**, **db_owner** o **db_ddladmin** pueden ejecutar instrucciones de DDL. En general, se recomienda no utilizar otras cuentas para crear objetos de base de datos. Si distintos usuarios crean sus propios objetos en una base de datos, se requiere que el propietario de cada objeto conceda los permisos adecuados a cada usuario de esos objetos. Esto genera trabajo administrativo y debe evitarse. Restringir los permisos de instrucciones a esas funciones también evita los problemas de propiedad de los objetos que se pueden producir cuando el propietario de un objeto se ha quitado de una base de datos o cuando el propietario de un procedimiento almacenado o vista no es propietario de las tablas subyacentes.

Ejemplo

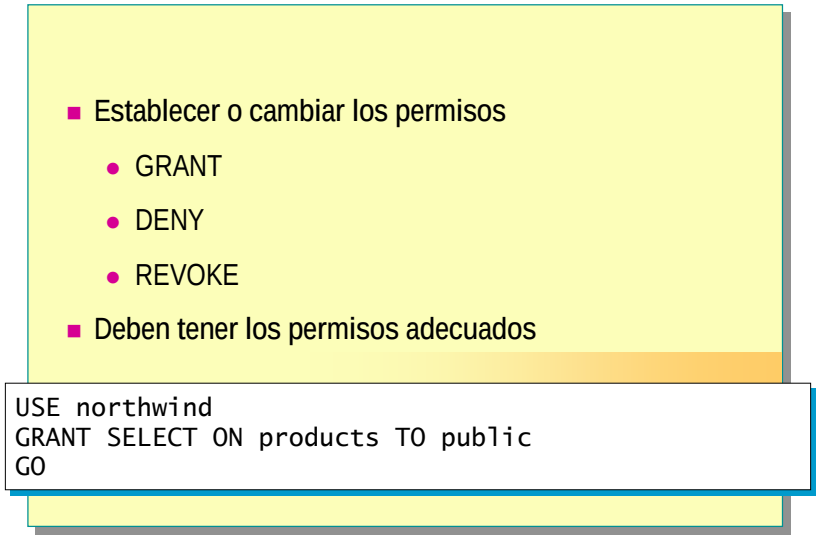
La secuencia de comandos siguiente crea una tabla llamada **customer** en la base de datos **Northwind**. Incluye las columnas **cust_id**, **company**, **contact** y **phone**.

```
USE northwind
CREATE TABLE customer
(cust_id int, company varchar(40),contact varchar(30),
phone char(12) )
GO
```

Instrucciones del Lenguaje de control de datos

Objetivos de la diapositiva
Presentar a los alumnos las instrucciones de DCL.

Explicación previa
Las instrucciones de DCL controlan el acceso a los objetos de la base de datos y a la capacidad de ejecutar ciertas instrucciones.



Sugerencia
Para obtener más información acerca de las instrucciones de DCL, consulte el curso 2323A, *Administración de una base de datos Microsoft SQL Server 2000*.

Las instrucciones de DCL se utilizan para cambiar los permisos asociados con un usuario o función de la base de datos. En la tabla siguiente se describen las instrucciones de DCL.

Instrucción	Descripción
GRANT	Crea una entrada en el sistema de seguridad que permite a un usuario trabajar con datos o ejecutar ciertas instrucciones de Transact-SQL.
DENY	Crea una entrada en el sistema de seguridad que deniega un permiso de una cuenta de seguridad e impide que el usuario, grupo o función herede el permiso a través de su pertenencia a grupos o funciones.
REVOKE	Quita un permiso concedido o denegado previamente.

De forma predeterminada, sólo los miembros de la función **sysadmin**, **dbcreator**, **db_owner** o **db_securityadmin** pueden ejecutar instrucciones DCL.

Ejemplo

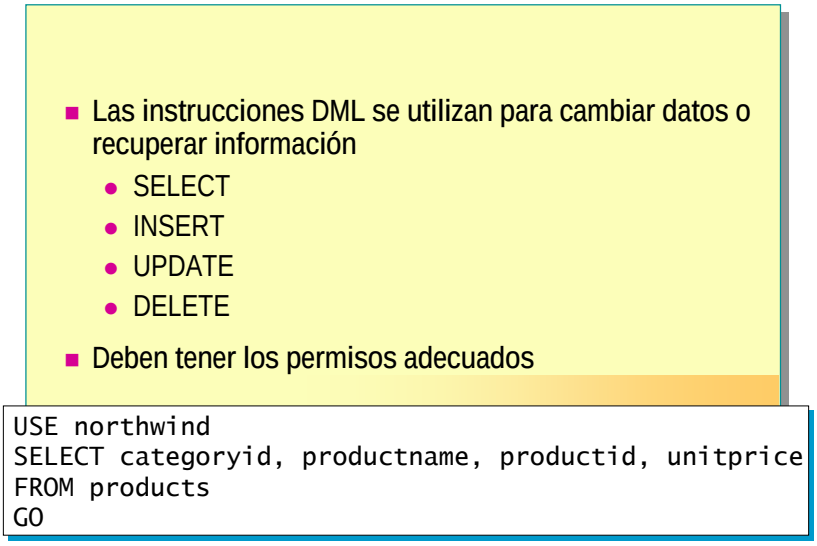
En este ejemplo se concede a la función **public** el permiso para consultar la tabla **products**.

```
USE northwind
GRANT SELECT ON products TO public
GO
```

Instrucciones del Lenguaje de tratamiento de datos

Objetivos de la diapositiva
Presentar a los alumnos las instrucciones de DML.

Explicación previa
Las instrucciones de DML funcionan con los datos de la base de datos.



Las instrucciones de DML funcionan con los datos de la base de datos. Mediante estas instrucciones puede cambiarlos o recuperar información. Las instrucciones de DML incluyen:

- SELECT
- INSERT
- UPDATE
- DELETE

De forma predeterminada, sólo los miembros de las funciones **sysadmin**, **dbcreator**, **db_owner** o **db_datawriter** pueden ejecutar instrucciones de DML.

Ejemplo

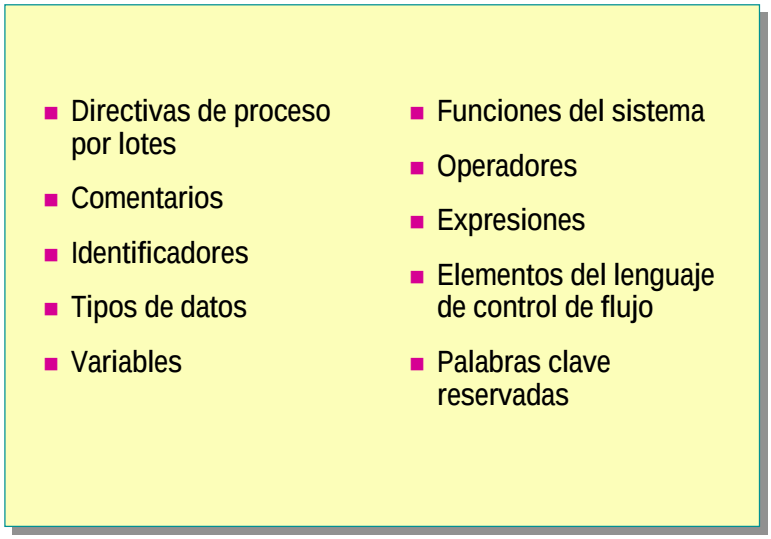
En este ejemplo se recupera el identificador de categoría, nombre de producto, identificador de producto y precio por unidad de los productos de la base de datos **Northwind**.

```
USE northwind
SELECT categoryid, productname, productid, unitprice
FROM products
GO
```

Elementos de la sintaxis de Transact-SQL

Objetivos de la diapositiva
Describir varios elementos de la sintaxis de Transact-SQL.

Explicación previa
Las instrucciones de DML se crean a partir de varios elementos de la sintaxis de Transact-SQL.



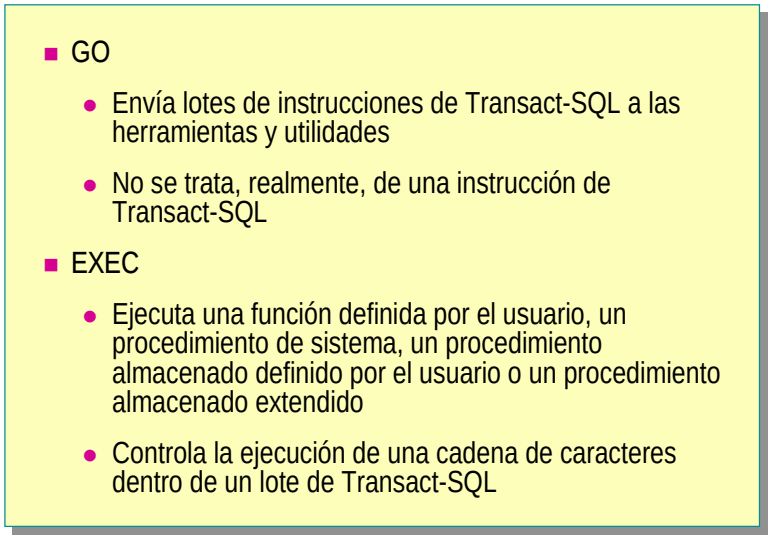
Las instrucciones de DML se crean a partir de varios elementos de la sintaxis de Transact-SQL. Entre estos elementos se encuentran los siguientes:

- Directivas de proceso por lotes
- Comentarios
- Identificadores
- Tipos de datos
- Variables
- Funciones del sistema
- Operadores
- Expresiones
- Elementos del lenguaje de control de flujo
- Palabras clave reservadas

Directivas de proceso por lotes

Objetivos de la diapositiva
Describir cómo se pueden ejecutar procesos por lotes de Transact-SQL.

Explicación previa
Sea cual sea la herramienta que use, necesita indicarle de alguna forma cómo debe procesar el código de Transact-SQL.



SQL Server procesa en lotes una o varias instrucciones de Transact-SQL. Una directiva de proceso por lotes indica a SQL Server que debe analizar y ejecutar todas las instrucciones que componen el lote. Hay dos métodos básicos de iniciar procesos por lotes en SQL Server.

GO

Las utilidades de SQL Server interpretan el comando GO como una señal para iniciar el envío del lote actual de instrucciones de Transact-SQL a SQL Server. Un comando GO envía lotes de instrucciones de Transact-SQL a las herramientas y utilidades. No se trata, realmente, de una instrucción de Transact-SQL.

Al usar el comando GO debe tener en cuenta estas cuestiones:

- El lote actual de instrucciones incluye todas las que se han escrito desde el comando GO anterior o desde el inicio de la sesión (o secuencia de comandos, si se trata del primer comando GO).
- Una instrucción de Transact-SQL no puede ocupar la misma línea que el comando GO, aunque esa línea sí puede contener comentarios.
- Los usuarios deben seguir las normas aplicables a los lotes.

  Por ejemplo, algunas instrucciones del Lenguaje de definición de datos (DDL) deben ejecutarse de forma independiente de otras instrucciones de Transact-SQL, por lo que se separan unas de otras por medio de un comando GO.

  El ámbito de las variables locales (definidas por el usuario) se limita a un lote, con lo que no se puede hacer referencia a ellas después de un comando GO.

Nota GO no es, realmente, una instrucción de Transact-SQL sino que se usa con el fin de determinar un lote para las herramientas y utilidades.

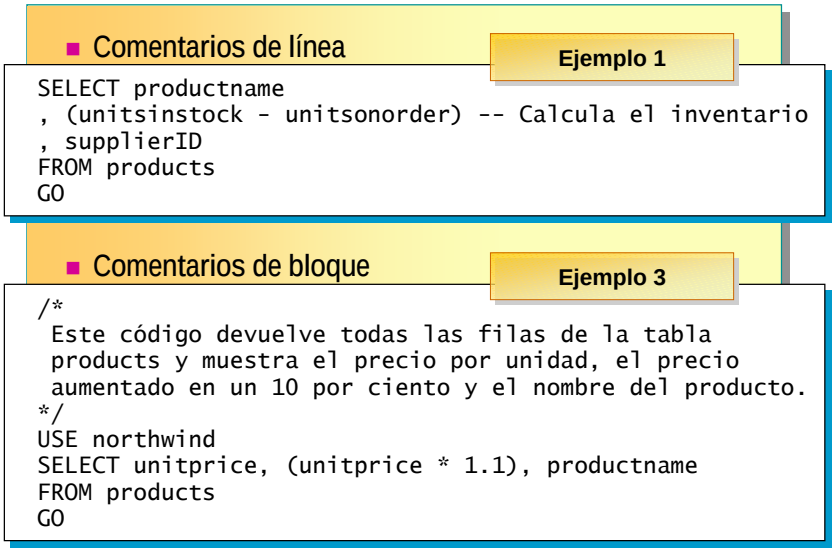
EXEC

La directiva EXEC se usa para ejecutar una función definida por el usuario, un procedimiento de sistema, un procedimiento almacenado definido por el usuario o un procedimiento almacenado extendido. También puede controlar la ejecución de una cadena de caracteres dentro de un lote de Transact-SQL. Se pueden pasar parámetros en forma de argumentos y también se puede asignar un estado de retorno.

Comentarios

Objetivos de la diapositiva
Presentar a los alumnos el uso común de los comentarios.

Explicación previa
Los comentarios son cadenas que no se ejecutan y que puede colocar en las instrucciones para realizar anotaciones o deshabilitar una parte de las mismas durante las pruebas.



Sugerencia
Los comentarios de línea que aparecen al principio de la línea pueden ayudar a solucionar problemas en las secuencias de comandos.

Los comentarios son cadenas de texto que no se ejecutan, colocadas en las instrucciones para describir la acción que la instrucción está realizando o para deshabilitar una o varias líneas de la instrucción. Se pueden utilizar de una de estas dos formas: en una línea de una instrucción o como un bloque.

Comentarios de línea

Se pueden crear comentarios en una línea mediante dos guiones (--) para establecer un comentario aparte de una instrucción. Transact-SQL pasa por alto el texto situado a la derecha de los caracteres de comentario. Este carácter de comentario se puede utilizar también para deshabilitar líneas de una instrucción.

Ejemplo 1

En este ejemplo se utiliza un comentario de línea para explicar qué está haciendo el cálculo.

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

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
*/
```

◆ Identificadores

Objetivos de la diapositiva
Presentar las reglas de denominación de los objetos de SQL Server (reglas de identificadores).

Explicación previa
SQL Server proporciona varias reglas estándar de denominación para los identificadores de objetos y un método para utilizar delimitadores con los identificadores que no son estándar.

- Identificadores estándar
  - El primer carácter debe ser un carácter alfabético
  - Otros caracteres pueden incluir letras, números o símbolos
  - Los identificadores que comienzan con un símbolo tienen usos especiales
- Identificadores delimitados
  - Se utilizan cuando los nombres contienen espacios incrustados
  - Se utilizan cuando partes de los nombres incluyen palabras reservadas
  - Deben encerrarse entre corchetes ([ ]) o dobles comillas (" ")

Sugerencia
Demuestre cómo el Analizador de consultas de SQL codifica con colores los elementos de las instrucciones para mostrar palabras reservadas, cadenas, etc.

SQL Server proporciona varias reglas estándar de denominación para los identificadores de objetos y un método para utilizar delimitadores con los identificadores que no son estándar. Si es posible, se recomienda utilizar los caracteres estándar de los identificadores para asignar nombres a los objetos.

Identificadores estándar

Los identificadores estándar pueden contener de uno a 128 caracteres, incluidos letras, símbolos (_, @, o #) y números. En los identificadores estándar no se permite incluir espacios. Las reglas para utilizar identificadores son:

- El primer carácter debe ser un carácter alfabético de la “a” a la “z” o de la “A” a la “Z”.
- Después del primer carácter, los identificadores pueden incluir letras, números o los símbolos @, $, # o _.
- Los nombres de los identificadores que comienzan con un símbolo tienen usos especiales:
  - Un identificador que comience con el símbolo @ indica una variable o parámetro local.
  - Un identificador que comience con el símbolo de almohadilla (#) indica una tabla o procedimiento temporal.
  - Un identificador que comience con una almohadilla doble (##) indica un objeto global temporal.

Nota Los nombres de los objetos temporales no deben superar los 116 caracteres, con el símbolo de almohadilla (#) o la doble almohadilla (##) incluidos, porque SQL Server asigna a los objetos temporales un sufijo numérico interno.

Para su información
La capacidad de los nombres de objetos de contener espacios es una novedad de esta versión de SQL Server.

Identificadores delimitados

Si un identificador cumple todas las reglas de formato de los identificadores, se puede utilizar con o sin delimitadores. Si un identificador no cumple alguna de las reglas de formato de los identificadores, siempre debe estar delimitado.

Los identificadores delimitados se pueden utilizar en las situaciones siguientes:

- Cuando los nombres contienen espacios incrustados.
- Cuando se utilizan palabras reservadas en los nombres de los objetos o en partes de los nombres de los objetos.

Los identificadores delimitados deben encerrarse entre corchetes o dobles comillas cuando se utilizan en las instrucciones de Transact-SQL.

- Los identificadores entre corchetes se delimitan mediante corchetes ([ ]):

```
SELECT * FROM [Blanks In Table Name]
```

> Nota Los delimitadores entre corchetes se pueden utilizar siempre, sin importar el estado de la opción SET QUOTED_IDENTIFIER.

- Los identificadores entre comillas se delimitan mediante dobles comillas (""):

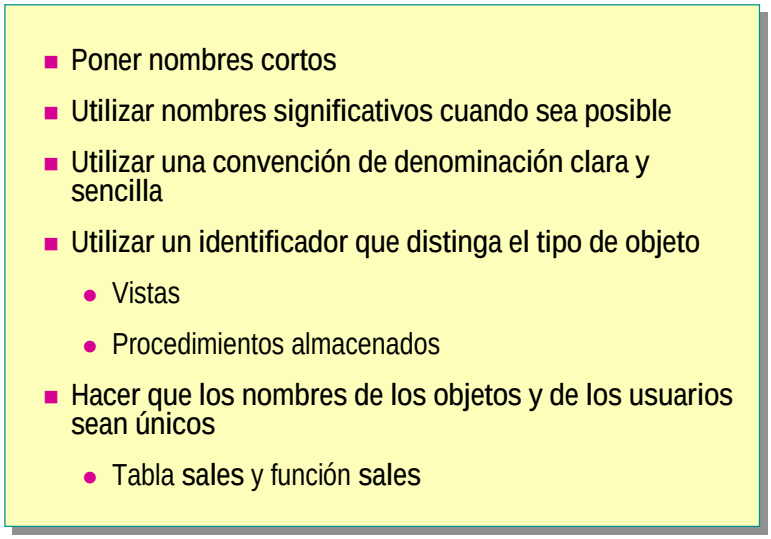
```
SELECT * FROM "Blanks In Table Name"
```

  Los identificadores entre comillas sólo se pueden utilizar si la opción SET QUOTED_IDENTIFIER está activada.

Directrices de denominación para los identificadores

Objetivos de la diapositiva
Presentar sugerencias de directrices de denominación.

Explicación previa
Utilice nombres cortos para los objetos de la base de datos.



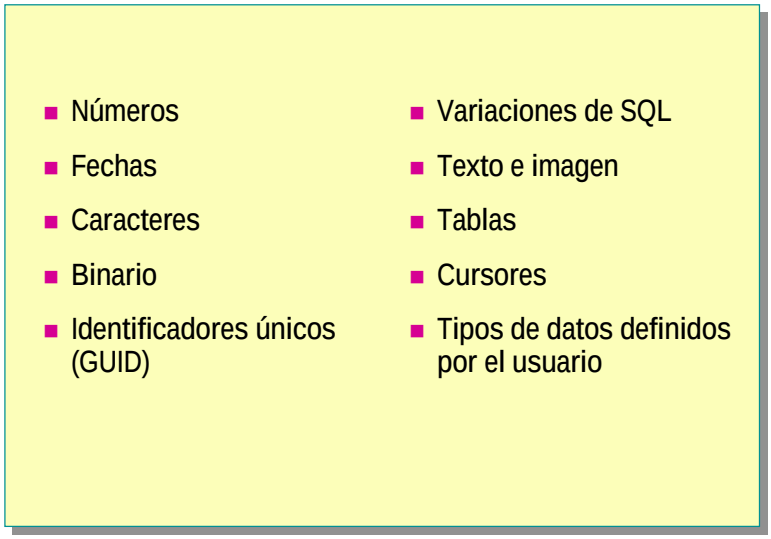
Cuando asigne nombres a los objetos de la base de datos, debe:

- Poner nombres cortos.
- Utilizar nombres significativos cuando sea posible.
- Utilizar una convención de denominación clara y sencilla. Decida qué funciona mejor para la situación y sea coherente. Intente que las convenciones de denominación no sean demasiado complejas, porque pueden resultar difíciles de seguir o de entender. Por ejemplo, puede quitar las vocales si el nombre de un objeto debe parecerse a una palabra clave (como un procedimiento almacenado de copia de seguridad llamado **bckup**).
- Utilizar un identificador que distinga el tipo de objeto, especialmente para las vistas y los procedimientos almacenados. A menudo, los administradores de sistemas confunden las vistas con las tablas, un descuido que puede causar problemas inesperados.
- Hacer que los nombres de los objetos y de los usuarios sean únicos. Por ejemplo, evite crear una tabla **sales** y una función **sales** en la misma base de datos.

Tipos de datos

Objetivos de la diapositiva
Describir los tipos de datos básicos de Transact-SQL.

Explicación previa
Los tipos de datos limitan los tipos de valores que se pueden almacenar en una base de datos.



Sugerencia
Los tipos de datos más sencillos y relativamente normales aparecen en la columna izquierda de la diapositiva, mientras que los más complejos están en la columna derecha.

Los tipos de datos limitan los tipos de valores que se pueden almacenar en una base de datos. Son atributos que especifican el tipo de información que se puede guardar en una columna, parámetro o variable. La mayor parte de las instrucciones de Transact-SQL no hacen referencia explícita a ningún tipo de datos, pero los resultados de la mayor parte de las instrucciones se ven influidos por las interacciones entre los tipos de datos de los objetos a los que se hace referencia en la instrucción.

SQL Server proporciona tipos de datos suministrados por el sistema (básicos), aunque puede crear otros tipos de datos adicionales. Éstos son algunos ejemplos de tipos de datos básicos:

Números

Este tipo de datos corresponde a los valores numéricos e incluye enteros como **int**, **tinyint**, **smallint** y **bigint**. También incluye valores decimales específicos como **numeric**, **decimal**, **money** y **smallmoney**. Y, finalmente, incluye valores de coma flotante como **float** y **real**.

Fechas

Este tipo de datos representa fechas o intervalos de tiempo. Los dos tipos de datos de fechas son **datetime**, que tiene una precisión de 0,333 milisegundos, y **smalldatetime**, que tiene una precisión de 1 minuto.

Caracteres

Este tipo de datos se usa para representar datos formados por caracteres o cadenas de caracteres. Incluye tipos de datos para cadenas de caracteres de ancho fijo, como **char** y **nchar**, así como tipos de datos para cadenas de caracteres de ancho variable, como **varchar** y **nvarchar**.

Para su información
El tipo de datos rowversion es el alias que usa SQL Server 2000 para el tipo de datos timestamp. El tipo de datos rowversion tiene las mismas funciones que timestamp. La definición de timestamp se cambiará en una versión futura de SQL Server para que coincida con la definición de timestamp en SQL-99.

Binario

Este tipo de datos es muy similar a los tipos de datos de caracteres en cuanto a almacenamiento y estructura, la diferencia es que los datos que contiene se tratan como si fueran una serie de valores de byte (octeto). Los tipos de datos binarios incluyen **binary** y **varbinary**. El tipo de datos **bit** indica un valor de un solo bit, es decir cero o uno. El tipo de datos **rowversion** indica un valor binario especial de 8 bytes que es único dentro de una base de datos.

Identificadores únicos

Este tipo especial de datos, **uniqueidentifier**, representa un identificador global único (GUID) que es un valor hexadecimal de 16 bytes que debe ser siempre único.

Variaciones de SQL

Este tipo de datos puede representar valores de diversos tipos de datos compatibles con SQL Server, exceptuando **text**, **ntext**, **rowversion** y otros valores **sql_variant**.

Texto e imagen

Estos tipos de datos corresponden a estructuras de objetos binarios grandes (BLOB, *Binary Large Object*) que representan tipos de datos de longitud fija y variable en los que se pueden guardar datos binarios y caracteres Unicode y no Unicode, como **image**, **text** y **ntext**.

Tablas

Este tipo de datos representa una estructura de tabla. En SQL Server 2000 es posible guardar una tabla dentro de un campo.

Cursores

Este tipo de datos se usa para programar dentro de procedimientos almacenados y con interfaces de cliente de bajo nivel. El tipo de datos cursor no se usa nunca en una instrucción DDL.

Tipos de datos definidos por el usuario

Este tipo de datos lo crea el administrador de la base de datos y está basado en tipos de datos del sistema. Los tipos de datos definidos por el usuario se usan cuando son varias las tablas que deben almacenar el mismo tipo de datos en una columna y se tiene que garantizar que todas ellas tengan el mismo tipo de datos, longitud y capacidad de admitir valores null.

Variables

Objetivos de la diapositiva
Definir una variable y describir cómo utilizarla.

Explicación previa
Las variables locales se declaran en el cuerpo de un programa por lotes o procedimiento mediante la instrucción DECLARE, y se les asignan valores con una instrucción SELECT.

- Variable definida por el usuario en una instrucción DECLARE @
- Valores asignados con una instrucción SET o SELECT @
- Las variables tienen el ámbito Local o Global

```
USE northwind
DECLARE  @EmpID  varchar(11)
        ,@vlName char(20)
SET @vlname = 'Dodsworth'
SELECT @EmpID = employeeid
 FROM  employees
 WHERE LastName = @vlname
SELECT @EmpID AS EmployeeID
GO
```

Las variables son elementos del lenguaje con valores asignados. En Transact-SQL se pueden utilizar variables locales.

Una variable local es una variable definida por el usuario en una instrucción DECLARE; se le asigna un valor inicial en una instrucción SET o SELECT y, después, se utiliza en la instrucción, programa por lotes o procedimiento en el que se declaró. Una variable local se identifica mediante un símbolo arroba (@) que precede a su nombre mientras que una variable global incluye dos símbolos arroba delante de su nombre.

Nota Las variables locales sólo duran el tiempo correspondiente a un proceso por lotes, mientras que las variables globales tiene la misma duración que una sesión.

Sintaxis

DECLARE {**@**variableLocal tipoDatos} [,...*n*]

SET *@nombreVariableLocal = expresión*

Ejemplo

En este ejemplo se crean las variables locales **@EmpID** y **@vlname**, se asigna un valor a **@vlname** y, a continuación, se asigna un valor a **@EmpID** al consultar en la base de datos **Northwind** para seleccionar el registro que contiene el valor de la variable local **@vlname**.

```
USE northwind
DECLARE  @EmpID  varchar(11)
        ,@vlName char(20)
SET @vlname = 'Dodsworth'
SELECT @EmpID = employeeid
 FROM employees
 WHERE LastName = @vlname
SELECT @EmpID AS EmployeeID
GO
```

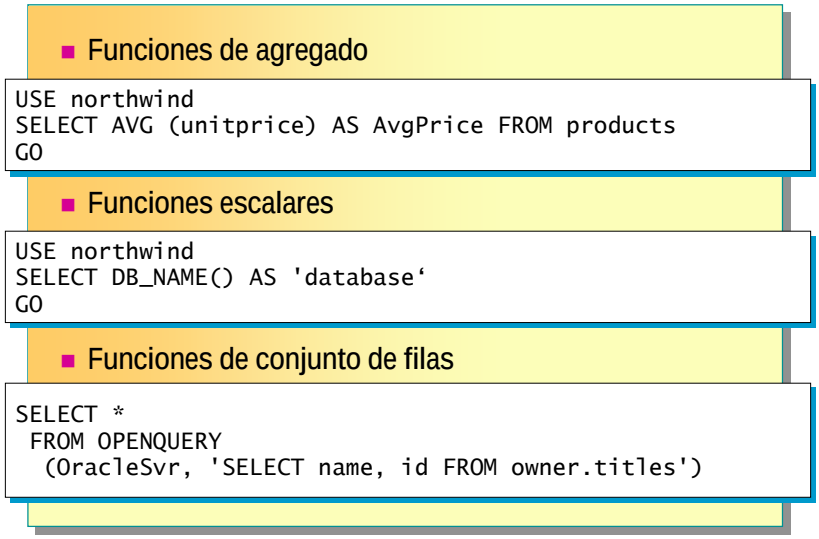
Resultado

```
EmployeeID
-----------
9
(1 fila afectada)
```

◆ Funciones del sistema

Objetivos de la diapositiva
Proporcionar información general acerca de las funciones disponibles en SQL Server.

Explicación previa
Transact-SQL proporciona muchas funciones del sistema que devuelven información.



Se pueden utilizar funciones del sistema en cualquier lugar en el que se permita una expresión en una instrucción SELECT. Transact-SQL proporciona muchas funciones que devuelven información.

Las funciones toman parámetros de entrada y devuelven valores que se pueden utilizar en expresiones. El lenguaje de programación Transact-SQL proporciona tres tipos de funciones:

Funciones de agregado Se ejecutan en una colección de valores, aunque devuelven un único valor de resumen.

Ejemplo 1

En este ejemplo se determina el promedio de la columna **unitprice** de todos los productos de la tabla **products**.

```
USE northwind
SELECT AVG(unitprice) AS AvgPrice
 FROM products
GO
```

Resultado

```
AvgPrice
---------------------
28.8663

(1 fila afectada)
```

Funciones escalares Se ejecutan con un único valor y devuelven también un valor único. Estas funciones se pueden utilizar en todos aquellos lugares en los que es válida una expresión. Las funciones escalares se pueden agrupar en las categorías siguientes.

Categoría de función	Descripción
Configuración	Devuelve información acerca de la configuración actual.
Cursor	Devuelve información acerca de los cursores.
Fecha y hora	Realiza una operación en un valor de entrada de fecha y hora, y devuelve un valor de cadena, numérico o de fecha y hora.
Matemática	Realiza un cálculo con los valores de entrada proporcionados como parámetros para la función y devuelve un valor numérico.
Metadatos	Devuelve información acerca de la base de datos y los objetos de la base de datos.
Seguridad	Devuelve información acerca de usuarios y funciones.
Cadena	Realiza una operación sobre un valor de entrada de cadena (**char** o **varchar**) y devuelve un valor de cadena o numérico.
Sistema	Realiza operaciones y devuelve información acerca de valores, objetos y configuraciones de SQL Server.
Estadísticas del sistema	Devuelve información estadística acerca del sistema.
Texto e imagen	Realiza una operación en un valor de entrada o columna de tipo texto o imagen, y devuelve información acerca del valor.

Ejemplo 2

En este ejemplo de función de metadatos se devuelve el nombre de la base de datos que se está utilizando actualmente.

```
USE northwind
SELECT DBNAME() AS 'database'
GO
```

Resultado

```
database
Northwind

(1 fila afectada)
```

Funciones de conjunto de filas Se pueden utilizar como referencias de tablas en una instrucción de Transact-SQL.

Ejemplo 3

En este ejemplo se ejecuta una consulta distribuida para recuperar información de la tabla **titles**.

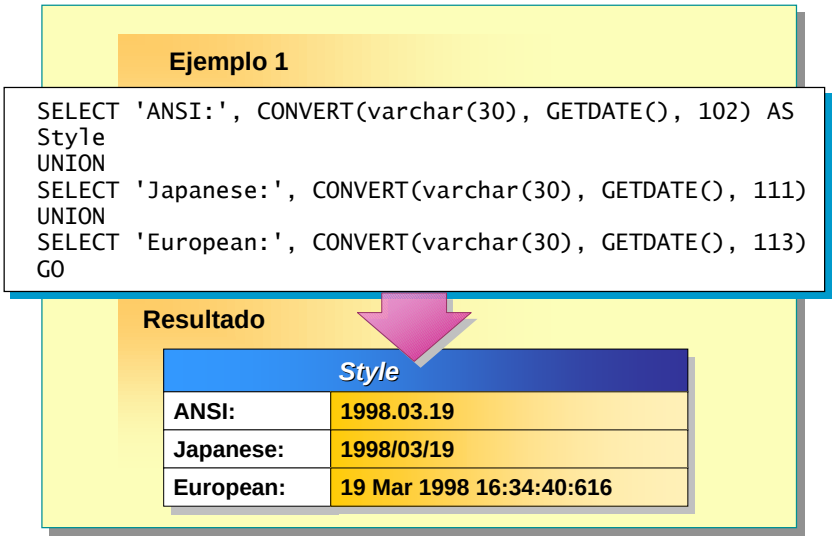
Sugerencia
Este ejemplo no se ejecutará correctamente si no tiene acceso a un servidor Oracle.

```
SELECT *
FROM OPENQUERY(OracleSvr, 'SELECT name, id FROM owner.titles')
GO
```

Ejemplos de función del sistema

Objetivos de la diapositiva
Mostrar algunos de los usos más comunes de las funciones.

Explicación previa
Las funciones del sistema se suelen utilizar para convertir datos de fecha del formato de un país al de otro país.



Sugerencia
Vaya al tema acerca de "CONVERT" de los Libros en pantalla de SQL Server y resalte las opciones de estilo.

Las funciones del sistema se suelen utilizar para convertir datos de fecha del formato de un país al de otro país.

Nota Para cambiar los formatos de fecha, debe utilizar la función CONVERT con la opción de estilo para determinar el formato de fecha que se devolverá.

Ejemplo 1

En este ejemplo se demuestra cómo puede convertir fechas a distintos estilos.

```
SELECT 'ANSI:', CONVERT (varchar(30), GETDATE(), 102) AS Style
UNION
SELECT 'Japanese:', CONVERT(varchar(30), GETDATE(), 111)
UNION
SELECT 'European:', CONVERT(varchar(30), GETDATE(), 113)
GO
```

Resultado

```
Style
European:    19 Mar 1998 16:34:40:616
Japanese:    1998/03/19
ANSI:        1998.03.19
```

Ejemplo 2

En este ejemplo se utiliza la opción DATEFORMAT de la instrucción SET para dar formato a las fechas de la duración de una conexión. Esta configuración sólo se utiliza en la interpretación de las cadenas de caracteres cuando se convierten a valores de fecha. No tiene efecto al mostrar los valores de fecha.

```
SET DATEFORMAT dmy
GO
DECLARE @vdate datetime
SET @vdate = '29/11/98'
SELECT @vdate
GO
```

Resultado

```
1998-11-29 00:00:00.000

(1 fila afectada)
```

Ejemplo 3

En este ejemplo se devuelve el nombre del usuario actual y la aplicación que está utilizando en la sesión o conexión actual. El usuario de este ejemplo es miembro de la función **sysadmin**.

```
USE library
SELECT user_name(), app_name()
GO
```

Resultado

```
dbo          Analizador de consultas SQL

(1 fila afectada)
```

Ejemplo 4

En este ejemplo se determina si la columna **firstname** de la tabla **member** de la base de datos **library** admite valores NULL.

Un resultado de cero (falso) significa que no se permiten valores NULL, mientras que un resultado de uno (verdadero) significa que se permiten valores NULL. Observe que la función OBJECT_ID está incrustada en la función COLUMNPROPERTY. Esto le permite obtener el **object id** de la tabla **member**.

```
USE library
SELECT COLUMNPROPERTY(OBJECT_ID('member'), 'firstname',
'AllowsNull')
GO
```

Resultado

```
0

(1 fila afectada)
```

Operadores

Objetivos de la diapositiva
Mostrar cómo se utilizan los operadores para tratar los conjuntos de resultados.

Explicación previa
Los operadores se pueden utilizar para realizar cálculos o comparar valores.

- Tipos de operadores
  - Aritmético
  - Comparación
  - Concatenación de cadenas
  - Lógico
- Niveles de precedencia de los operadores

Los operadores son símbolos que realizan cálculos matemáticos, concatenaciones de cadenas y comparaciones entre columnas, constantes y variables. Se pueden combinar y utilizar en las condiciones de búsqueda. Cuando se combinan, el orden en el que se procesan los operadores se basa en una precedencia predefinida.

Sintaxis parcial

```
{constante | nombreColumna | función | (subconsulta)}
  [{operadorAritmético | operadorCadena |
    AND | OR | NOT}
    {constante | nombreColumna | función | (subconsulta)}…]
```

Tipos de operadores

SQL Server admite cuatro tipos de operadores: aritmético, comparación, concatenación de cadenas y lógico.

Sugerencia
Transact-SQL admite además operadores binarios. Se omiten aquí porque se usan con poca frecuencia.

Aritmético

Los operadores aritméticos realizan cálculos con columnas numéricas o constantes. Transact-SQL admite operadores multiplicativos, que incluyen la multiplicación (*), división (/) y módulo (%) (el número entero que queda al dividir números enteros), y los operadores aditivos de adición (+) y sustracción (-).

Comparación

Los operadores de comparación comparan dos expresiones. Las comparaciones se pueden realizar entre variables, columnas y expresiones de tipo similar. Los operadores de comparación se incluyen en la tabla siguiente.

Operador	Significado
=	Igual que
>	Mayor que
<	Menor que
>=	Mayor o igual que
<=	Menor o igual que
<>	No igual que

Concatenación de cadenas

El operador de concatenación de cadenas (+) concatena valores de cadena. El resto de las operaciones con cadenas se controla mediante las funciones de cadena. Las cadenas vacías no se evalúan nunca como un valor NULL.

Lógico

Los operadores lógicos AND, OR y NOT conectan las condiciones de búsqueda de las cláusulas WHERE.

Sugerencia
Señale que los niveles de precedencia de los operadores lógicos en SQL Server son distintos a los que se usan en otros lenguajes de programación.

Niveles de precedencia de los operadores

Si utiliza varios operadores (lógicos o aritméticos) para combinar expresiones, SQL Server procesa los operadores en el orden de precedencia, lo que puede afectar al valor resultante. Los operadores tienen los siguientes niveles de precedencia (de mayor a menor).

Tipo	Operador	Símbolo
Grupo	Agrupación principal	( )
Aritmético	Multiplicativo	* / %
Aritmético	Aditivo	- +
Otros	Concatenación de cadenas	+
Lógico	NOT	NOT
Lógico	AND	AND
Lógico	OR	OR

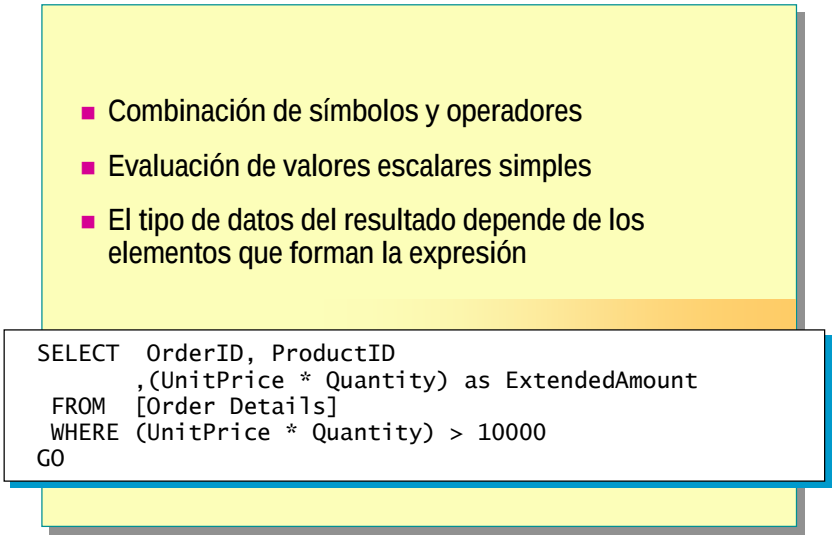
SQL Server resuelve primero las expresiones que se encuentran en la parte más interna de la anidación. Además, si todos los operadores aritméticos de una expresión comparten el mismo nivel de precedencia, el orden es de izquierda a derecha.

Nota Los niveles de precedencia de los operadores lógicos en SQL Server son distintos a los que se usan en otros lenguajes de programación.

Expresiones

Objetivos de la diapositiva
Presentar a los alumnos las expresiones.

Explicación previa
Las expresiones son combinaciones de símbolos y operadores que dan como resultado un único valor.



Las *expresiones* son combinaciones de símbolos y operadores que dan como resultado un único valor. Puede tratarse de expresiones simples, como una constante, variable, columna o valor escalar, o expresiones complejas que se crean mediante la conexión, por medio de operadores, de una o varias expresiones simples.

El tipo de datos del resultado depende de los elementos que forman la expresión. Durante la evaluación del resultado, se realizan con frecuencia conversiones implícitas de los tipos de datos de los elementos que componen la expresión.

Ejemplo

En el ejemplo siguiente se calcula el valor ampliado de un producto que forma parte de un pedido; para ello se multiplica el precio unitario por la cantidad pedida y, a continuación, se filtran los resultados de forma que sólo se devuelvan las filas correspondientes a pedidos con un valor ampliado mayor de 10000.

```
SELECT  OrderID, ProductID
       ,(UnitPrice * Quantity) as ExtendedAmount
 FROM  [Order Details]
 WHERE (UnitPrice * Quantity) > 10000
GO
```

Resultado

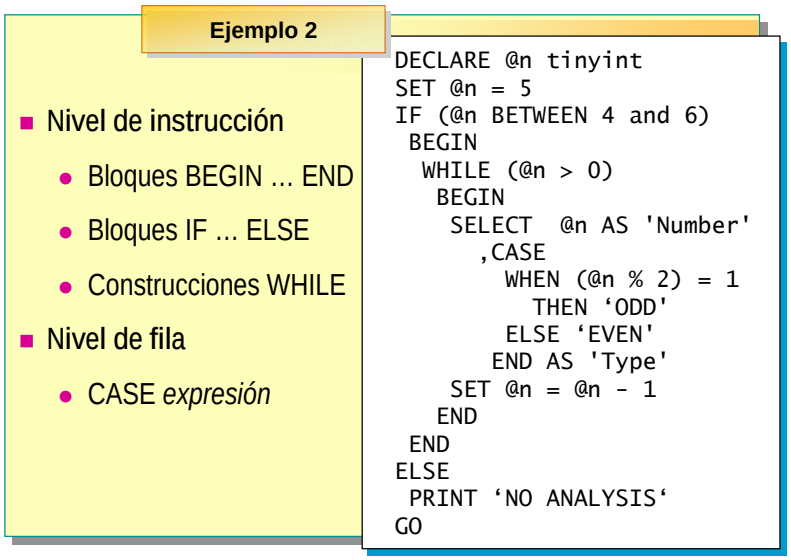
OrderID	ProductID	ExtendedAmount
10353	38	10540.0000
10417	38	10540.0000
10424	38	10329.2000
10865	38	15810.0000
10889	38	10540.0000
10981	38	15810.0000

(6 filas afectadas)

Elementos del lenguaje de control de flujo

Objetivos de la diapositiva
Presentar los elementos del lenguaje Transact-SQL que controlan el procesamiento de las instrucciones.

Explicación previa
Transact-SQL proporciona un lenguaje que controla el flujo de la lógica en las instrucciones.



Transact-SQL contiene varios elementos de lenguaje que controlan el flujo de la lógica de una instrucción. También contiene la expresión CASE, que permite utilizar la lógica condicional en una única fila dentro de una instrucción SELECT o UPDATE.

Nivel de instrucción

Los siguientes elementos del lenguaje permiten controlar el flujo de la lógica en una secuencia de comandos:

Bloques BEGIN ... END Estos elementos encierran varias instrucciones de Transact-SQL para que se traten como una unidad.

Bloques IF ... ELSE Estos elementos especifican que SQL Server debe ejecutar la primera alternativa si una condición es verdadera. En caso contrario, SQL Server debe ejecutar la segunda alternativa.

Construcciones WHILE Estos elementos ejecutan varias veces una instrucción siempre y cuando la condición que se especifica sea cierta. Las instrucciones BREAK y CONTINUE controlan la operación de las instrucciones incluidas en el bucle WHILE.

Sugerencia Para mejorar la legibilidad puede aplicar sangrías a las instrucciones de Transact-SQL que componen un bloque de control de flujo.

Ejemplo 1

En este ejemplo se determina si un cliente tiene algún pedido antes de eliminarlo de la lista de clientes.

```
USE northwind
IF EXISTS (SELECT * FROM orders
           WHERE customerid = 'frank')
  PRINT '*** Customer cannot be deleted ***'
ELSE
  BEGIN
    DELETE customers WHERE customerid = 'frank'
    PRINT '*** Customer deleted ***'
  END
GO
```

Nivel de fila

Una expresión CASE enumera predicados, asigna un valor a cada uno y, a continuación, prueba cada uno de ellos. Si la expresión devuelve un valor verdadero, la expresión CASE devuelve el valor de la cláusula WHEN. Si la expresión es falsa y ha especificado una cláusula ELSE, SQL Server devuelve el valor de la cláusula ELSE. Puede utilizar una expresión CASE en cualquier lugar en el que pueda utilizar una expresión.

Sintaxis

CASE *expresión*
  {WHEN *expresión* THEN *resultado*} [,…*n*]
 [ELSE *resultado*]
 END

Ejemplo 2

En el ejemplo siguiente se declara una variable local, se comprueba si su valor es 4, 5 ó 6, y si lo es, se pasa ese número de veces por un bucle WHILE para determinar si el valor actual es un número par o impar.

Sugerencia
Señale que las sangrías de bloque que se usan en el ejemplo 2 mejoran la legibilidad.

```
DECLARE  @n tinyint
SET   @n = 5
IF (@n BETWEEN 4 and 6)
 BEGIN
  WHILE (@n > 0)
   BEGIN
    SELECT  @n AS 'Number'
      ,CASE
        WHEN (@n % 2) = 1
          THEN 'ODD'
        ELSE 'EVEN'
       END AS 'Type'
    SET @n = @n - 1
   END
 END
ELSE
 PRINT 'NO ANALYSIS'
GO
```

Resultado

Number	Type
5	ODD

(1 fila afectada)

Number	Type
4	EVEN

(1 fila afectada)

Number	Type
3	ODD

(1 fila afectada)

Number	Type
2	EVEN

(1 fila afectada)

Number	Type
1	ODD

(1 fila afectada)

Palabras clave reservadas

Objetivos de la diapositiva
Presentar a los alumnos el concepto de palabra clave reservada.

Explicación previa
SQL Server tiene reservadas determinadas palabras para su uso.
No debe usarlas en los nombres de identificadores.

- Nombres de identificadores que tienen un significado especial
  - Palabras clave de Transact-SQL
  - Palabras clave ANSI SQL-92
  - Palabras clave reservadas de ODBC
- No utilice palabras clave reservadas para nombres de identificadores

Sugerencia
Muestre la amplia lista de palabras clave que aparece en los Libros en pantalla de SQL Server.

SQL Server tiene reservadas determinadas palabras clave para su uso exclusivo. Por ejemplo, si se usan las palabras clave DUMP o BACKUP en una sesión de **osql** o del Analizador de consultas de SQL, se estará indicando a SQL Server que debe realizar una copia de seguridad total o parcial de la base de datos, o una copia de seguridad del registro.

No se pueden incluir palabras clave reservadas en ningún lugar dentro de una instrucción de Transact-SQL, excepto en los casos que define SQL Server. Debe evitar usar una palabra clave reservada como nombre de un objeto. Si el nombre de un objeto coincide con una palabra clave, cada vez que se haga referencia al objeto debe aparecer entre identificadores delimitadores, como dobles comillas (“ ”) o corchetes ([ ]).

La persona que asume las funciones de administrador de base de datos o del sistema, o el creador de la base de datos, suele ser responsable de comprobar que no se usan palabras clave reservadas en las instrucciones de Transact-SQL ni en los nombres de las bases de datos.

Advertencia Es posible construir instrucciones de Transact-SQL sintácticamente correctas que se analicen y compilen correctamente, y que devuelvan un error en tiempo de ejecución cuando se ejecutan. Es preferible no utilizar nunca palabras clave reservadas.