Modificación de datos

Contenido

Notas para el instructor

Este módulo describe cómo funcionan las transacciones y cómo escribir instrucciones INSERT, DELETE y UPDATE para modificar datos de tablas.

Al terminar este módulo, el alumno será capaz de:

- Describir cómo funcionan las transacciones.
- Escribir instrucciones INSERT, DELETE y UPDATE para modificar datos de tablas.
- Describir las consideraciones acerca del rendimiento relacionadas con la modificación de datos.

Introducción

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

Explicación previa
En este módulo aprenderá acerca de la modificación de datos.

- Uso de transacciones
- Inserción de datos
- Eliminación de datos
- Actualización de datos
- Consideraciones acerca del rendimiento

Este módulo describe cómo funcionan las transacciones y explica cómo escribir instrucciones INSERT, DELETE y UPDATE para modificar los datos de tablas.

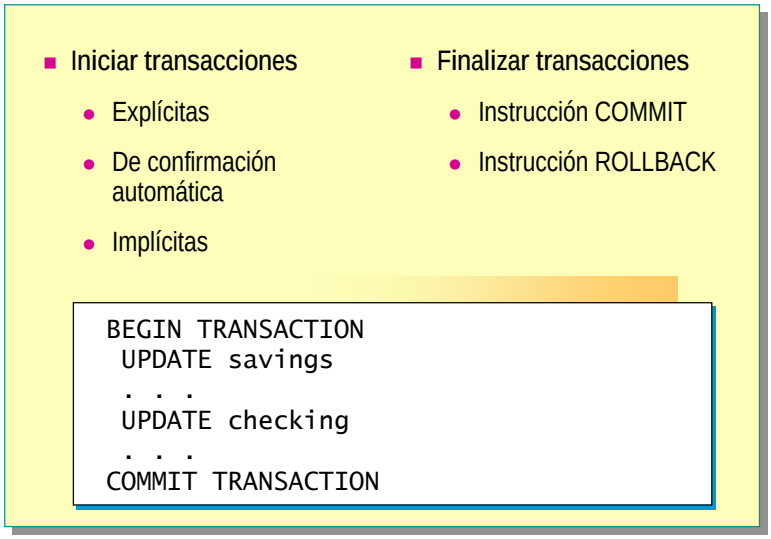
Al terminar este módulo, el alumno será capaz de:

- Describir cómo funcionan las transacciones.
- Escribir instrucciones INSERT, DELETE y UPDATE para modificar datos de tablas.
- Describir las consideraciones acerca del rendimiento relacionadas con la modificación de datos.

Uso de transacciones

Objetivos de la diapositiva
Presentar los temas que abarca esta sección.

Explicación previa
Las transacciones se utilizan para exigir la integridad de los datos.



Una *transacción* es una secuencia de operaciones realizadas como una sola unidad lógica de trabajo. Los programadores de SQL son responsables de iniciar y finalizar transacciones en puntos que hagan cumplir la coherencia lógica de los datos. El programador debe definir la secuencia de modificaciones de datos que los dejen en un estado coherente con respecto a las reglas empresariales de la organización.

Iniciar transacciones

Puede iniciar transacciones en Microsoft® SQL Server™ 2000 de uno de tres modos: *explícito*, *confirmación automática* o *implícito*.

- Las transacciones *explícitas* se inician con la emisión de una instrucción BEGIN TRANSACTION.
- Las transacciones de *confirmación automática* son el modo predeterminado de SQL Server. Cada instrucción de Transact-SQL se confirma una vez completada. No tiene que especificar ninguna instrucción para controlar transacciones.
- El modo de transacciones *implícito* se establece mediante una función de interfaz de programación de aplicaciones (API) o por medio de la instrucción SET IMPLICIT_TRANSACTIONS ON de Transact-SQL. Con este modo, la instrucción siguiente inicia automáticamente una nueva transacción. Una vez completada dicha transacción, la instrucción de Transact-SQL siguiente inicia una nueva transacción.

El modo de transacción se establece en cada sesión. Si una sesión cambia de un modo de transacción a otro, el cambio no tendrá efecto sobre la sesión del modo de transacción.

Finalizar transacciones

Puede finalizar transacciones con una instrucción COMMIT o ROLLBACK.

La instrucción COMMIT indica que, si una transacción es correcta, SQL Server debe confirmarla. Una instrucción COMMIT garantiza que todas las modificaciones de la transacción son parte permanente de la base de datos. Una instrucción COMMIT también libera recursos, como los bloqueos, que utiliza la transacción.

La instrucción ROLLBACK cancela una transacción: deshace todas las modificaciones realizadas en la transacción al devolver los datos al estado que tenían al comienzo de la transacción. Una instrucción ROLLBACK también libera los recursos mantenidos por la transacción. Si se produce un error dentro de una transacción, SQL Server ejecuta automáticamente una instrucción ROLLBACK de la transacción en curso.

Ejemplo

En este ejemplo se transfieren 100 dólares de una cuenta de ahorro a la cuenta corriente de un cliente, mediante una transacción. Si durante cualquier punto de la transacción hubiera un error, se desharían los cambios realizados en los datos.

```
BEGIN TRANSACTION

  UPDATE savings
   SET balance = balance - 100
   WHERE custid = 78910

  IF @@ERROR <> 0
   BEGIN
     RAISERROR ('Error, transaction not completed!', 16, -1)
     ROLLBACK TRANSACTION
   END

  UPDATE checking
   SET balance = balance +100
   WHERE custid = 78910

  IF @@ERROR <> 0
   BEGIN
     RAISERROR ('Error, transaction not completed!', 16, -1)
     ROLLBACK TRANSACTION
   END

COMMIT TRANSACTION
```

Inserción de datos

Objetivos de la diapositiva
Presentar los temas que abarca esta sección.

Explicación previa
Puede insertar datos mediante una transacción si especifica un conjunto de valores o inserta los resultados de una instrucción SELECT.
.

- Inserción de una fila de datos mediante valores
- Uso de la instrucción INSERT...SELECT
- Creación de una tabla mediante la instrucción SELECT INTO
- Inserción de datos parciales
- Inserción de datos mediante valores de columna predeterminados

Puede insertar datos mediante una transacción si especifica un conjunto de valores o inserta los resultados de una instrucción SELECT. Puede crear una tabla e insertar datos al mismo tiempo. No tiene que insertar valores en todos los campos de datos de una fila.

Inserción de una fila de datos mediante valores

Objetivos de la diapositiva
Mostrar cómo agregar una fila de valores a una tabla con la instrucción INSERT.

Explicación previa
La instrucción INSERT agrega filas a las tablas.

- Debe atenerse a las restricciones de destino o la transacción INSERT fallará
- Use una lista de columnas para especificar las columnas de destino
- Especifique una lista de valores correspondiente

```
USE northwind
INSERT customers
      (customerid, companyname, contactname, contacttitle
      ,address, city, region, postalcode, country, phone
      ,fax)

VALUES ('PECOF', 'Pecos Coffee Company', 'Michael Dunn'
         ,'Owner', '1900 Oak Street', 'Vancouver', 'BC'
         ,'V3F 2K1', 'Canada', '(604) 555-3392'
         ,'(604) 555-7293')
GO
```

La instrucción INSERT agrega filas a una tabla.

Sintaxis parcial

```
INSERT [INTO]
  { nombreTabla | nombreVista}
  { [(listaColumnas)]
  { VALUES ( { DEFAULT | NULL| expresión}[,…n])
  | DEFAULT VALUES
```

Sugerencia
Señale en el ejemplo de la diapositiva que todos los valores de la tabla **customers** son valores de tipo carácter y, por tanto, se encierran entre comillas simples.

La instrucción INSERT con la cláusula VALUES permite agregar filas a una tabla. Al insertar filas, tenga en cuenta los siguientes hechos e instrucciones:

- Debe atenerse a las restricciones de destino o la transacción INSERT fallará.
- Utilice *listaColumnas* para especificar las columnas en las que se va a almacenar cada valor especificado. *listaColumnas* debe especificarse entre paréntesis y delimitarse con comas. Si especifica valores para todas las columnas, *listaColumnas* es opcional.
- Para especificar los valores que desea insertar, utilice la cláusula VALUES. Esta cláusula se requiere para cada columna de la tabla o de *listaColumnas*.

  El orden y el tipo de los nuevos datos debe corresponder al orden y al tipo de las columnas de la tabla. Muchos tipos de datos tienen un formato de entrada asociado. Por ejemplo, los datos de carácter y las fechas deben encerrarse entre comillas simples.

Ejemplo

En el ejemplo siguiente se agrega Pecos Coffee Company como nuevo cliente.

```
USE northwind
INSERT customers
      (customerid, companyname, contactname, contacttitle
      ,address, city, region, postalcode, country, phone
      ,fax)
VALUES ('PECOF', 'Pecos Coffee Company','Michael Dunn'
       ,'Owner', '1900 Oak Street', 'Vancouver', 'BC'
       ,'V3F 2K1', 'Canada', '(604) 555-3392'
       ,'(604) 555-7293')
GO
```

Para comprobar que se ha agregado Pecos Coffee Company a la tabla **customers**, puede ejecutar la instrucción siguiente:

```
USE northwind
SELECT companyname, contactname
 FROM customers
 WHERE customerid = 'PECOF'
GO
```

Resultado

Companyname	contactname
Pecos Coffee Company	Michael Dunn

(1 fila afectada)

Uso de la instrucción INSERT…SELECT

Objetivos de la diapositiva
Mostrar cómo insertar filas de una tabla en otra mediante instrucciones SELECT anidadas.

Explicación previa
Se pueden insertar filas de una tabla en la misma tabla o en otra mediante instrucciones SELECT anidadas.

- Todas las filas que cumplan la instrucción SELECT se insertan
- Compruebe que existe la tabla que recibe las nuevas filas
- Asegúrese de que son compatibles los tipos de datos
- Determine si existe un valor predeterminado o si se permiten valores Null

```
USE northwind
INSERT customers
 SELECT substring (firstname, 1, 3)
         + substring (lastname, 1, 2)
       ,lastname, firstname, title, address, city
       ,region, postalcode, country, homephone, NULL
  FROM employees
GO
```

La instrucción INSERT…SELECT agrega filas a una tabla mediante la inserción del conjunto de resultados de una instrucción SELECT.

Puede emplear la instrucción INSERT…SELECT para agregar filas de otros orígenes a una tabla existente. Utilizar la instrucción INSERT…SELECT es más eficiente que escribir varias instrucciones INSERT de una sola fila. Al utilizar la instrucción INSERT…SELECT, tenga en cuenta los siguientes hechos e instrucciones:

- Todas las filas que cumplan la instrucción SELECT se insertan en la tabla más externa de la consulta.
- Debe comprobar que la tabla que recibe las nuevas filas existe en la base de datos.
- Debe asegurarse de que las columnas de la tabla que recibe los nuevos valores tienen tipos de datos compatibles con las columnas de la tabla de origen.
- Debe determinar si existe un valor predeterminado o si se permiten valores NULL en alguna de las columnas que se omiten. Si no se permiten valores NULL, debe proporcionar valores para esas columnas.

Sintaxis

INSERT *nombreTabla*
SELECT *listaColumnas*
FROM *listaTablas*
WHERE *condicionesBúsqueda*

Ejemplo

En este ejemplo se agregan nuevos clientes a la tabla **customers**. Los empleados de Northwind Traders pueden comprar productos de la empresa. Esta consulta contiene una instrucción INSERT con una cláusula SELECT que agrega información de los empleados a la tabla **customers**.

La nueva columna **customerid** consta de las tres primeras letras del nombre del empleado, concatenadas con las dos primeras letras del apellido. El apellido del empleado se utiliza como el nombre de la nueva compañía y el nombre se utiliza como nombre de contacto.

```
USE northwind
INSERT customers
 SELECT substring (firstname, 1, 3)
         + substring (lastname, 1, 2)
       ,lastname, firstname, title, address, city
       ,region, postalcode, country, homephone, NULL
 FROM employees
GO
```

Creación de una tabla mediante la instrucción SELECT INTO

Objetivos de la diapositiva
Explicar el propósito y la función de la instrucción SELECT INTO.

Explicación previa
Mediante la instrucción SELECT INTO, el conjunto de resultados de cualquier consulta se puede colocar en una tabla nueva.

- Úsela para crear una tabla e insertar filas en ella en una sola operación
- Puede crear una tabla temporal local o global
- Establezca la opción de base de datos select into/ bulkcopy en ON para crear una tabla permanente
- En la lista de selección, debe crer alias de columnas o especificar los nombres de las columnas de la nueva tabla

```
USE northwind
SELECT productname AS products
      ,unitprice AS price
      ,(unitprice * 1.1) AS tax
 INTO #pricetable
 FROM products
GO
```

Mediante la instrucción SELECT INTO, el conjunto de resultados de cualquier consulta se puede colocar en una tabla nueva.

Utilice la instrucción SELECT INTO para llenar tablas nuevas de una base de datos con datos importados. También se puede utilizar la instrucción SELECT INTO para resolver problemas complejos que requieran un conjunto de datos de varios orígenes. Si crea primero una tabla temporal, las consultas que ejecute sobre ella serán más sencillas que las que ejecutaría sobre varias tablas o bases de datos.

Al utilizar la instrucción SELECT INTO, tenga en cuenta los siguientes hechos e instrucciones:

- Puede utilizar la instrucción SELECT INTO para crear una tabla e insertar filas en la tabla en una sola operación.

  Asegúrese de que el nombre de la tabla que se especifique en la instrucción SELECT INTO sea único. Si ya existe una tabla con el mismo nombre, la instrucción SELECT INTO producirá un error.

- Puede crear una tabla temporal local o global.

  Para crear una tabla temporal local, preceda el nombre de la tabla con el signo de almohadilla (#) o con dos signos de almohadilla (##) para crear una tabla temporal global.

  Una tabla temporal local sólo es visible en la sesión actual. Una tabla temporal global es visible en todas las sesiones:

  - El espacio de las tablas temporales locales se recupera cuando el usuario finaliza la sesión.
  - El espacio de las tablas temporales locales se recupera cuando la tabla deja de utilizarse en todas las sesiones.

- Establezca la opción de base de datos **select into/bulkcopy** (select into o copia masiva) en ON para crear una tabla permanente.
- En la lista de selección, debe crear alias de columnas o especificar los nombres de las columnas de la nueva tabla.

Sintaxis parcial

SELECT <*listaSelección*>
INTO *nuevaTabla*
FROM {<*tablaOrigen*>}[,…*n*]
WHERE <*condiciónBúsqueda*>

Ejemplo

En este ejemplo se crea una tabla temporal local en función de una consulta efectuada en la tabla **products**. Observe que para tratar el conjunto de resultados se pueden utilizar funciones de cadena y funciones matemáticas.

Sugerencia
Muestre este ejemplo con el Analizador de consultas SQL.

```
USE northwind
SELECT productname AS products
      ,unitprice AS price
      ,(unitprice * 1.1) AS tax
 INTO #pricetable
 FROM products
GO
```

Para ver el conjunto de resultados debe ejecutar la consulta siguiente.

```
USE northwind
SELECT * FROM #pricetable
GO
```

Resultado

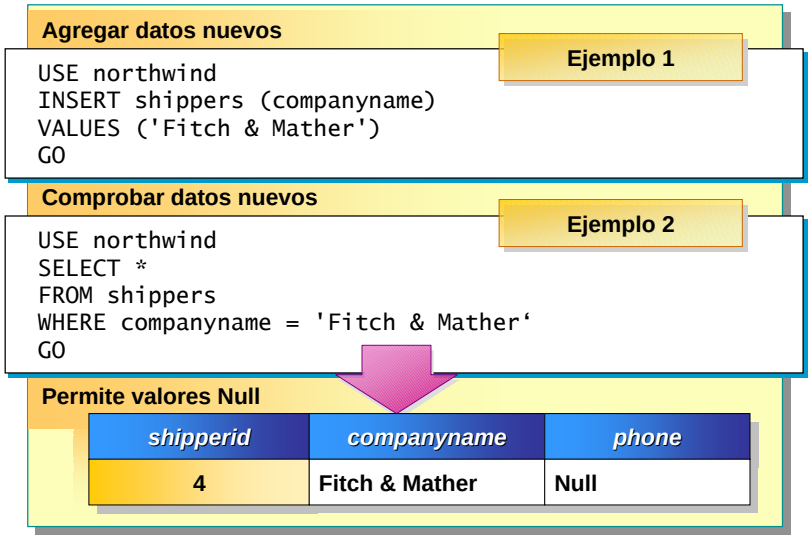
products	price	tax
Chai	18	19.8
Chang	19	20.9
Aniseed Syrup	10	11
Chef Anton's Cajun Seasoning	22	24.2
Chef Anton's Gumbo Mix	21.35	23.485
Grandma's Boysenberry Spread	27.5	30.25
Uncle Bob's Organic Dried Pears	33	36.3
Northwoods Cranberry Sauce	44	48.4
Mishi Kobe Niku	97	106.7
Ikura	31	34.1
Queso Cabrales	21	23.1
Queso Manchego La Pastora	38	41.8
Konbu	6	6.6
Tofu	23.25	25.575
Genen Shouyu	15.5	17.05
.		
.		
.		

(77 filas afectadas)

Inserción de datos parciales

Objetivos de la diapositiva
Explicar cómo insertar una fila sin especificar todos sus datos.

Explicación previa
Si una columna tiene un valor predeterminado o acepta valores NULL, puede dejarla fuera de la instrucción INSERT. SQL Server insertará automáticamente el valor correspondiente.



Sugerencia
Compare los ejemplos 1 y 2. En el ejemplo 1 no se utiliza la palabra clave DEFAULT. Ambos ejemplos devuelven el mismo conjunto de resultados.

Si una columna tiene un valor predeterminado o admite valores NULL, puede omitirla en la instrucción INSERT. SQL Server insertará automáticamente el valor correspondiente.

Al insertar datos parciales, tenga en cuenta los siguientes hechos e instrucciones:

- Incluya sólo los nombres de las columnas para las que especifica datos en la instrucción INSERT.
- Especifique las columnas para las que proporciona un valor en *listaColumnas*. Los datos de la cláusula VALUES corresponden a las columnas especificadas. Las columnas no mencionadas se llenan como si se hubieran incluido y se hubiera especificado un valor predeterminado.
- No incluya en *listaColumnas* las columnas que tengan la propiedad IDENTITY o admitan valores predeterminados o NULL.
- Si desea especificar explícitamente un valor NULL, escriba NULL sin comillas.

Ejemplo 1

En este ejemplo se agrega la compañía Fitch & Mather como nuevo transportista en la tabla **shippers**. No se especifican datos para las columnas que tienen la propiedad IDENTITY o admiten valores predeterminados o NULL. Compare este ejemplo con el ejemplo 2. Observe que se omite la palabra clave DEFAULT.

```
USE northwind
INSERT shippers (companyname)
VALUES ('Fitch & Mather')
GO
```

Para comprobar que se ha agregado Fitch & Mather a la tabla **shippers** puede ejecutar la instrucción siguiente:

```
USE northwind
SELECT *
FROM shippers
WHERE companyname = 'Fitch & Mather'
GO
```

Resultado

shipperid	companyname	phone
4	Fitch & Mather	NULL

```
(1 fila afectada)
```

Ejemplo 2

En este ejemplo también se agrega Fitch & Mather como nuevo transportista a la tabla **shippers**. Observe que se utiliza la palabra clave DEFAULT para las columnas que admiten valores predeterminados o NULL. Compare este ejemplo con el ejemplo 1.

```
USE northwind
INSERT shippers (companyname, Phone)
VALUES ('Fitch & Mather', DEFAULT)
GO
```

Resultado

shipperid	companyname	phone
4	Fitch & Mather	NULL

```
(1 fila afectada)
```

Inserción de datos mediante valores de columna predeterminados

Objetivos de la diapositiva
Explicar las palabras clave DEFAULT y DEFAULT VALUES.

Explicación previa
Utilice la instrucción INSERT con la palabra clave DEFAULT si desea insertar el valor predeterminado de columnas específicas o utilice la palabra clave DEFAULT VALUES para insertar una fila completa en una tabla.

- Palabra clave DEFAULT
  - Inserta valores predeterminados para columnas específicas
  - Las columnas deben tener un valor predeterminado o permitir valores nulos

```
USE northwind
INSERT shippers (companyname, phone)
 VALUES ('Kenya Coffee Co.', DEFAULT)
GO
```

- Palabra clave DEFAULT VALUES
  - Inserta valores predeterminados para todas las columnas
  - Las columnas deben tener un valor predeterminado o permitir valores nulos

Al insertar filas en una tabla puede ahorrar tiempo si en lugar de escribir los valores, utiliza las palabras clave DEFAULT o DEFAULT VALUES en la instrucción INSERT.

Sugerencia
Céntrese en la sintaxis parcial y compare en ella las palabras clave DEFAULT y DEFAULT VALUES.

Palabra clave DEFAULT

Cuando una tabla tiene restricciones predeterminadas o cuando una columna tiene un valor predeterminado, puede utilizar la palabra clave DEFAULT en la instrucción INSERT para que SQL Server proporcione automáticamente el valor predeterminado.

Al utilizar la palabra clave DEFAULT, tenga en cuenta los siguientes hechos e instrucciones:

- SQL Server inserta un valor NULL en las columnas que admiten valores NULL y no tienen un valor predeterminado.
- Si utiliza la palabra clave DEFAULT y las columnas no tienen valores predeterminados ni admiten valores NULL, la instrucción INSERT causará un error.
- No es posible utilizar la palabra clave DEFAULT para las columnas que tengan la propiedad IDENTITY (un valor asignado e incrementado automáticamente). Por tanto, no debe incluir estas columnas en *listaColumnas* o en la cláusula VALUES.
- SQL Server inserta automáticamente el valor adecuado en las columnas definidas con el tipo de datos **rowversion**.

Ejemplo

En este ejemplo se inserta una nueva fila para Kenya Coffee Company sin utilizar *listaColumnas*. La columna **shippers.shipperid** tiene la propiedad IDENTITY y no se incluye en la lista de columnas. La columna **phone** admite valores NULL.

```
USE northwind
INSERT shippers (companyname, phone)
 VALUES ('Kenya Coffee Co.', DEFAULT)
GO
```

Para comprobar que se ha agregado Kenya Coffee Co. a la tabla **shippers**, puede ejecutar la instrucción siguiente:

```
USE northwind
SELECT *
 FROM shippers
 WHERE companyname = 'Kenya Coffee Co.'
GO
```

Resultado

shipperid	companyname	Phone
10	Kenya Coffee Co.	NULL

```
(1 fila afectada)
```

Palabra clave DEFAULT VALUES

La palabra clave DEFAULT VALUES permite insertar una fila completa en una tabla. Al utilizar la palabra clave DEFAULT VALUES, tenga en cuenta los siguientes hechos e instrucciones:

- SQL Server inserta valores NULL en las columnas que los admiten y no tienen un valor predeterminado.
- Si utiliza la palabra clave DEFAULT VALUES y las columnas no tienen valores predeterminados ni admiten valores NULL, la instrucción INSERT causará un error.
- SQL Server inserta automáticamente el valor adecuado en las columnas con la propiedad IDENTITY o con el tipo de datos **rowversion**.
- Puede utilizar la palabra clave DEFAULT VALUES para generar datos de ejemplo y llenar tablas con valores predeterminados.

◆ Eliminación de datos

Objetivos de la diapositiva
Presentar los temas que abarca esta sección.

Explicación previa
Puede especificar los datos que desea eliminar.
.

- Uso de la instrucción DELETE
- Uso de la instrucción TRUNCATE TABLE
- Eliminación de filas basada en otras tablas

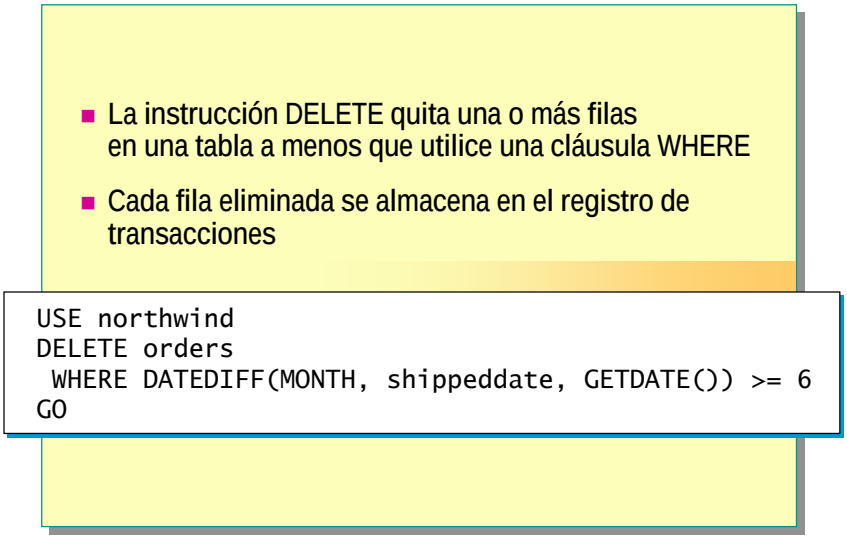
Puede especificar los datos que desea eliminar.

La instrucción DELETE quita una o varias filas de una tabla o una vista mediante una transacción. Puede especificar las filas que elimina SQL Server si filtra la tabla de destino o mediante una cláusula JOIN o una subconsulta. La instrucción TRUNCATE TABLE se utiliza para quitar todas las filas de una tabla sin utilizar transacciones.

Uso de la instrucción DELETE

Objetivos de la diapositiva
Explicar cómo eliminar filas de las tablas.

Explicación previa
La instrucción DELETE quita filas de las tablas.



La instrucción DELETE quita filas de las tablas. La instrucción DELETE permite quitar una o varias filas de una tabla.

Sintaxis parcial

DELETE [*from*] {*nombreTabla*|*nombreVista*}
WHERE *condicionesBúsqueda*

Ejemplo

En este ejemplo se eliminan todos los registros de pedidos con seis o más meses de antigüedad.

```
USE northwind
DELETE orders
WHERE DATEDIFF(MONTH, shippeddate, GETDATE()) >= 6

GO
```

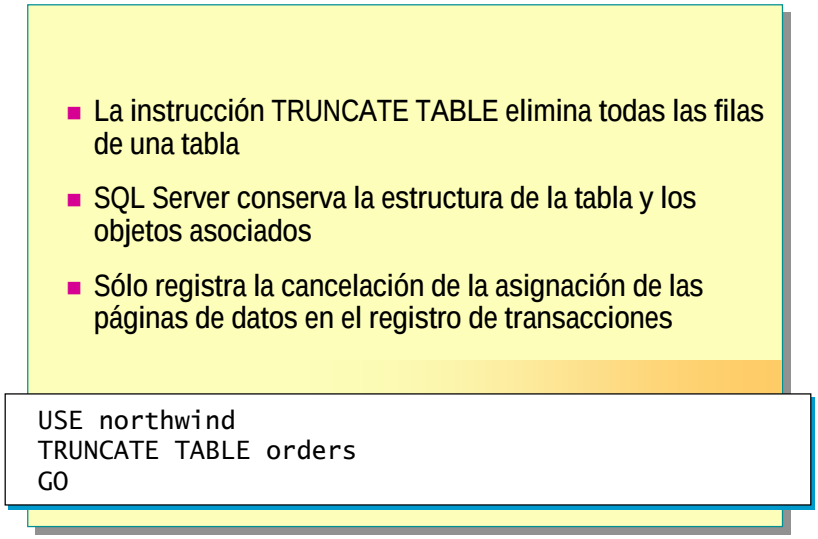
Al utilizar la instrucción DELETE, tenga en cuenta los hechos siguientes:

- SQL Server eliminará todas las filas de la tabla si no incluye una cláusula WHERE en la instrucción DELETE.
- Cada fila eliminada se almacena en el registro de transacciones.

Uso de la instrucción TRUNCATE TABLE

Objetivos de la diapositiva
Describir cómo usar la instrucción TRUNCATE TABLE.

Explicación previa
La instrucción TRUNCATE TABLE quita todos los datos de una tabla.



La instrucción TRUNCATE TABLE quita todos los datos de una tabla. Utilice la instrucción TRUNCATE TABLE para realizar una eliminación no registrada de todas las filas.

Sintaxis

TRUNCATE TABLE [[*baseDatos.*]*propietario.*]*nombreTabla*

Ejemplo

En este ejemplo se quitan todos los datos de la tabla **orders**.

```
USE northwind
TRUNCATE TABLE orders
GO
```

Al utilizar la instrucción TRUNCATE TABLE, tenga en cuenta los hechos siguientes:

- SQL Server elimina todas las filas pero conserva la estructura de la tabla y los objetos asociados a ella.
- La instrucción TRUNCATE TABLE se ejecuta más rápidamente que la instrucción DELETE, ya que SQL Server sólo registra la cancelación de la asignación de las páginas de datos.
- Si una tabla tiene una columna IDENTITY, la instrucción TRUNCATE TABLE restablece el valor de inicio.

Para su información
No puede utilizar TRUNCATE TABLE en una tabla a la que haga referencia una restricción de clave externa; en su lugar, utilice la instrucción DELETE sin una cláusula WHERE.

Eliminación de filas basada en otras tablas

Objetivos de la diapositiva
Mostrar cómo eliminar datos de una tabla en función de los datos de otras tablas.

Explicación previa
La instrucción DELETE se puede utilizar con una cláusula FROM adicional (o una subconsulta en la cláusula WHERE) para ver los datos de otras tablas y determinar si debe eliminarse una fila.

- Uso de una cláusula FROM adicional
  - La primera cláusula FROM indica la tabla que se va a modificar
  - La segunda cláusula FROM especifica los criterios de restricción para la instrucción DELETE
- Especificación de condiciones en la cláusula WHERE
  - Las subconsultas determinan qué filas eliminar

Sugerencia
Compare los ejemplos 1 y 2.

Puede utilizar la instrucción DELETE con combinaciones o subconsultas para quitar filas de una tabla en función de los datos almacenados en otras tablas. Esto es más eficiente que escribir varias instrucciones DELETE de una sola fila.

Uso de una cláusula FROM adicional

En una instrucción DELETE, la cláusula WHERE hace referencia a los valores de la propia tabla y se utiliza para decidir qué filas se deben eliminar. Si utiliza una cláusula FROM adicional, puede hacer referencia a otras tablas para tomar esta decisión. Al utilizar la instrucción DELETE con una cláusula FROM adicional, tenga en cuenta los siguientes hechos:

- La primera cláusula FROM indica la tabla de la que se eliminarán filas.
- La segunda cláusula FROM puede introducir una combinación y actúa como los criterios de restricción para la instrucción DELETE.

Sintaxis

DELETE [FROM] {*nombreTabla* | *nombreVista*}
[FROM {<*origenTabla*>} [, …*n*]]
[WHERE *condicionesBúsqueda*]

Ejemplo 1

En este ejemplo se utiliza una operación de combinación con la instrucción DELETE para quitar de la tabla **order details** las filas de los pedidos recibidos el 14/4/1998.

Sugerencia
Señale la cláusula FROM adicional en la instrucción.

```
USE northwind
DELETE FROM [order details]
 FROM orders AS o
 INNER JOIN [order details] AS od
  ON o.orderid = od.orderid
 WHERE orderdate = '14/4/1998'
GO
```

Especificación de condiciones en la cláusula WHERE

También puede utilizar subconsultas para determinar qué filas se van a eliminar de una tabla en función de las filas de otra tabla. Puede especificar las condiciones de la cláusula WHERE en lugar de utilizar una cláusula FROM adicional. Utilice una subconsulta anidada o correlacionada en la cláusula WHERE para determinar las filas que deben eliminarse.

Ejemplo 2

En este ejemplo se quitan de la tabla **order details** las mismas filas que en el ejemplo 1 y se muestra cómo puede convertir una operación de combinación en una subconsulta anidada.

```
USE northwind
DELETE FROM [order details]
 WHERE orderid IN (
                   SELECT orderid
                    FROM orders
                    WHERE orderdate = '14/4/1998'
                  )
GO
```

Actualización de datos

Objetivos de la diapositiva
Presentar los temas que abarca esta sección.

Explicación previa
La instrucción UPDATE puede cambiar valores de datos de filas individuales, grupos de filas o todas las filas de una tabla o de una vista.

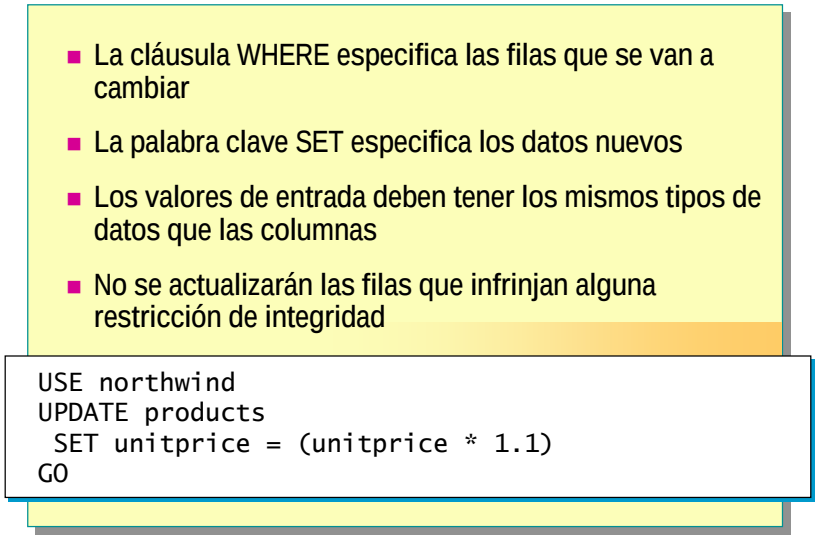
- Actualización de filas basada en datos de la tabla
- Actualización de filas basada en otras tablas

La instrucción UPDATE puede cambiar valores de datos de filas individuales, grupos de filas o todas las filas de una tabla o de una vista. Puede actualizar una tabla según sus datos o los datos de otras tablas.

Actualización de filas basada en datos de la tabla

Objetivos de la diapositiva
Mostrar cómo se actualizan las filas con la instrucción UPDATE.

Explicación previa
La instrucción UPDATE permite cambiar los datos de las filas existentes en una tabla.



La instrucción UPDATE modifica datos existentes.

Sintaxis parcial

UPDATE {*nombreTabla* | *nombreVista*}
SET
{ *nombreColumna* = {*expresión* | DEFAULT | NULL} |
@variable=expresión}[**,** …*n*]
WHERE {*condicionesBúsqueda*}

La instrucción UPDATE permite cambiar filas individuales, grupos de filas o todas las filas de una tabla. Al actualizar filas, tenga en cuenta los hechos e instrucciones siguientes:

- Especifique las filas que desea actualizar con la cláusula WHERE.
- Especifique los nuevos valores con la cláusula SET.
- Compruebe que los valores de entrada tienen los mismos tipos de datos que los definidos para las columnas.
- SQL Server no hará actualizaciones que infrinjan alguna restricción de integridad. En ese caso, no se producirán los cambios y la instrucción se deshará.
- Sólo es posible cambiar los datos de una tabla cada vez.
- Puede establecer una expresión en una o varias columnas o variables. Por ejemplo, una expresión puede ser un cálculo (como **price** * 2) o la suma de dos columnas.

Ejemplo

En el ejemplo siguiente se agrega el 10 por ciento a los precios actuales de todos los productos de Northwind Traders.

```
USE northwind
UPDATE products
 SET unitprice = (unitprice * 1.1)
GO
```

Actualización de filas basada en otras tablas

Objetivos de la diapositiva
Mostrar cómo puede utilizar combinaciones o subconsultas para actualizar los datos de una tabla en función de los datos de otra tabla.

Explicación previa
La instrucción UPDATE se puede utilizar para actualizar filas en función de otras tablas.

- Uso de la instrucción UPDATE
  - Nunca actualiza la misma fila dos veces
  - Requiere prefijos de tablas en nombres de columnas ambiguos
- Especificación de filas para actualizar con combinaciones
  - Utilice la cláusula FROM
- Especificación de filas para actualizar con subconsultas
  - Correlacione la subconsulta con la tabla actualizada

Puede utilizar la instrucción UPDATE con una cláusula FROM para modificar una tabla en función de los valores de otras tablas.

Uso de la instrucción UPDATE

Al utilizar combinaciones y subconsultas con la instrucción UPDATE, tenga en cuenta los siguientes hechos e instrucciones:

- SQL Server nunca actualiza la misma fila dos veces en una única instrucción UPDATE. Ésta es una restricción integrada que disminuye la cantidad de registros que se genera durante las actualizaciones.
- Utilice la palabra clave SET para presentar la lista de columnas o nombres de variables que deben actualizarse. Las columnas a las que se haga referencia con la palabra clave SET no deben ser ambiguas. Por ejemplo, puede utilizar un prefijo de tabla para eliminar ambigüedades.

Sintaxis parcial

```
UPDATE {nombreTabla | nombreVista}
  SET
  { nombreColumna={expresión | DEFAULT | NULL}
    |@variable=expresión}[, …n]
  [FROM {   <origenTabla>
  ]
  [WHERE condicionesBúsqueda]
```

Especificación de filas para actualizar con combinaciones

Al emplear combinaciones para actualizar filas, utilice la cláusula FROM para especificar las combinaciones en la instrucción UPDATE.

Ejemplo 1

En este ejemplo se utiliza una combinación con el fin de actualizar la tabla **products** y se suman 2,00 dólares a la columna **unitprice** para todos los productos suministrados por los proveedores en Estados Unidos.

```
USE northwind
UPDATE products
 SET unitprice = unitprice + 2
 FROM products
 INNER JOIN suppliers
  ON products.supplierid = suppliers.supplierid
 WHERE suppliers.country = 'USA'
GO
```

Especificación de filas para actualizar con subconsultas

Al utilizar subconsultas para actualizar filas, tenga en cuenta los siguientes hechos e instrucciones:

- Si la subconsulta no devuelve un único valor, debe escribirla con la palabra clave IN, EXISTS, ANY o ALL.
- Considere la utilización de funciones de agregado con las subconsultas correlacionadas, ya que SQL Server nunca actualiza la misma fila dos veces en una única instrucción UPDATE.

Ejemplo 2

En este ejemplo se utiliza una subconsulta con el fin de actualizar la tabla **products** y se suman 2,00 dólares a la columna **unitprice** para todos los productos suministrados por los proveedores de Estados Unidos. Observe que cada producto tiene un único proveedor.

```
USE northwind
UPDATE products
 SET unitprice = unitprice + 2
 WHERE supplierid IN (
                      SELECT supplierid
                       FROM suppliers
                       WHERE country = 'USA'
                     )
GO
```

Ejemplo 3

En este ejemplo se actualiza el total de ventas de todos los pedidos de cada producto de la tabla **products**. Puede haber muchos pedidos por cada producto. Como SQL Server nunca actualiza dos veces la misma fila, debe utilizar una función de agregado con una subconsulta correlacionada para actualizar el número total de ventas hasta la fecha de cada producto. Si desea ejecutar el ejemplo siguiente, debe agregar una columna **todatesales** con el valor predeterminado 0 a la tabla **products**.

```
USE northwind
UPDATE products
 SET todatesales = (
                    SELECT SUM(quantity)
                     FROM [order details] AS od
                     WHERE products.productid = od.productid
                    )
GO
```

Consideraciones acerca del rendimiento

Objetivos de la diapositiva
Describir las consideraciones acerca del rendimiento al utilizar vistas, desencadenadores o procedimientos almacenados.

Explicación previa
Las modificaciones de datos que se producen dentro de las transacciones pueden afectar al rendimiento de SQL Server.

- Las modificaciones de datos se producen en una transacción
- Se producen asignaciones de páginas de datos
- La modificación de datos indizados disminuye el rendimiento
- Los índices pueden mejorar los criterios de búsquedas

Puntos clave
Sólo una transacción cada vez puede modificar una fila específica. Una transacción en curso bloquea la ejecución de otras transacciones hasta que la transacción se confirma o deshace.

Las modificaciones de datos que se producen dentro de las transacciones pueden afectar al rendimiento de SQL Server. Al modificar datos, recuerde que:

- El bloqueo de datos durante una transacción individual puede impedir que otras transacciones y consultas se ejecuten hasta que no se complete la transacción.
- Modificar tablas puede cambiar la forma en que los datos se almacenan físicamente, lo que conlleva asignaciones de páginas de datos que tienen que producirse dentro de la transacción.
- Al modificar columnas de datos que están indizadas, los índices de dichas columnas cambian como parte de la transacción.
- Colocar índices en las columnas utilizadas en la cláusula WHERE de una instrucción de modificación de datos mejora el rendimiento.