Implementación de desencadenadores

Contenido

Notas para el instructor

En este módulo se proporciona a los alumnos la definición de desencadenador y se explica la forma crearlos. Los desencadenadores son una herramienta útil para los creadores de bases de datos que deseen que se realicen determinadas acciones cuando se inserten, actualicen o eliminen datos en una tabla específica. Son un método especialmente útil para exigir reglas de empresa y asegurar la integridad de los datos.

Se explican las siguientes partes de un desencadenador:

- La instrucción que lo activa (INSERT, UPDATE o DELETE)
- La tabla que protege
- La acción que realiza el desencadenador cuando se le invoca

Esta sección trata también los usos de los desencadenadores y los aspectos que hay que tener en cuenta al determinar si un desencadenador es la herramienta adecuada para llevar a cabo una tarea.

En la sección siguiente se describe el proceso de crear y quitar desencadenadores, y se ofrece información detallada acerca del funcionamiento de los cuatro tipos de desencadenadores: INSERT, UPDATE, DELETE e INSTEAD OF. En esta sección se explica también cómo alterar los desencadenadores.

A continuación, se expone una descripción de cómo trabajar con desencadenadores, que incluye ejemplos de desencadenadores anidados, y cómo pueden utilizarse para exigir la integridad de los datos, la integridad referencial y las reglas de empresa.

El módulo concluye con una lista de recomendaciones y consideraciones acerca del rendimiento que hay que tener en cuenta al crear desencadenadores y trabajar con ellos.

En la práctica, los alumnos crearán desencadenadores y probarán su efectividad.

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

- Crear un desencadenador.
- Quitar un desencadenador.
- Alterar un desencadenador.
- Describir cómo funcionan diversos desencadenadores.
- Evaluar las consideraciones de rendimiento que afectan al uso de los desencadenadores.

Introducción

Objetivo del tema
Proporcionar una introducción a los temas y objetivos del módulo.

Explicación previa
En este módulo aprenderá acerca de la creación de desencadenadores.

- Introducción a los desencadenadores
- Definición de desencadenadores
- Funcionamiento de los desencadenadores
- Ejemplos de desencadenadores
- Consideraciones acerca del rendimiento

Un *desencadenador* es un procedimiento almacenado que se ejecuta cuando se modifican los datos de una tabla determinada. Los desencadenadores se suelen crear para exigir integridad referencial o coherencia entre datos relacionados de forma lógica en diferentes tablas. Como los usuarios no pueden evitar los desencadenadores, éstos se pueden utilizar para exigir reglas de empresa complejas que mantengan la integridad de los datos.

Después de completar este módulo, el alumno será capaz de:

- Crear un desencadenador.
- Quitar un desencadenador.
- Alterar un desencadenador.
- Describir cómo funcionan diversos desencadenadores.
- Evaluar las consideraciones de rendimiento que afectan al uso de los desencadenadores.

Introducción a los desencadenadores

Objetivo del tema
Presentar el concepto de objeto desencadenador.

Explicación previa
En esta sección aprenderá cuándo y cómo utilizar desencadenadores.

- ¿Qué es un desencadenador?
- Uso de los desencadenadores
- Consideraciones acerca del uso de desencadenadores

Esta sección presenta los desencadenadores y describe cuándo y cómo utilizarlos.

¿Qué es un desencadenador?

Objetivo del tema
Presentar el concepto de desencadenador y exponer las ventajas de su uso.

Explicación previa
Un desencadenador es una clase especial de procedimiento almacenado que se ejecuta automáticamente siempre que se intenta modificar los datos que el desencadenador protege. Los desencadenadores están asociados a tablas específicas.

- Asociación a una tabla
- Invocación automática
- Imposibilidad de llamada directa
- Identificación con una transacción

Puntos clave
Un desencadenador es un tipo especial de procedimiento almacenado. En él puede incluir cualquier instrucción de Transact-SQL. Los desencadenadores utilizan la caché de procedimientos para almacenar el plan de ejecución.

Un desencadenador es una clase especial de procedimiento almacenado que se ejecuta siempre que se intenta modificar los datos de una tabla que el desencadenador protege. Los desencadenadores están asociados a tablas específicas.

Asociación a una tabla

Los desencadenadores se definen para una tabla específica, denominada tabla del desencadenador.

Invocación automática

Cuando se intenta insertar, actualizar o eliminar datos de una tabla en la que se ha definido un desencadenador para esa acción específica, el desencadenador se ejecuta automáticamente. No es posible evitar su ejecución.

Imposibilidad de llamada directa

A diferencia de los procedimientos almacenados del sistema normales, no es posible invocar directamente los desencadenadores, que tampoco pasan ni aceptan parámetros.

Identificación con una transacción

Sugerencia
No describa las transacciones en detalle. Las transacciones se describen en otros módulos de este curso.

El desencadenador y la instrucción que causa su ejecución se tratan como una única transacción que puede deshacerse desde cualquier parte del desencadenador. Al utilizar desencadenadores, tenga en cuenta estos hechos e instrucciones:

- Las definiciones de desencadenadores pueden incluir una instrucción ROLLBACK TRANSACTION incluso cuando no haya una instrucción BEGIN TRANSACTION explícita.
- Si se encuentra una instrucción ROLLBACK TRANSACTION, se deshará toda la transacción. Si a continuación de la instrucción ROLLBACK TRANSACTION en la secuencia de comandos del desencadenador hay una instrucción, ésta se ejecutará. Por tanto, puede ser necesario utilizar una cláusula RETURN en una instrucción IF para impedir que se procesen las demás instrucciones.
- Si se activa un desencadenador que incluye una instrucción ROLLBACK TRANSACTION en una transacción definida por el usuario, la operación ROLLBACK TRANSACTION deshará toda la transacción. Un desencadenador ejecutado en un lote que incluye la instrucción ROLLBACK TRANSACTION cancela el lote, por lo que las instrucciones siguientes no se ejecutan.
- Es recomendable reducir o evitar el uso de ROLLBACK TRANSACTION en el código de los desencadenadores. Deshacer una transacción implica trabajo adicional, porque supone volver a realizar, a la inversa, todo el trabajo de la transacción completado hasta ese momento. Conlleva un efecto perjudicial en el rendimiento. La información debe comprobarse y validarse fuera de la transacción. Puede iniciar la transacción cuando todo esté comprobado.
- El usuario que invoca el desencadenador debe tener permiso para ejecutar todas las instrucciones en todas las tablas.

Usos de los desencadenadores

Objetivo del tema
Presentar las ventajas del uso de los desencadenadores.

Explicación previa
El uso de los desencadenadores tiene varias ventajas.

- Cambios en cascada en tablas relacionadas de una base de datos
- Exigir una integridad de datos más compleja que una restricción CHECK
- Definición de mensajes de error personalizados
- Mantenimiento de datos no normalizados
- Comparación del estado de los datos antes y después de su modificación

Sugerencia
Señale que se pueden utilizar desencadenadores para hacer actualizaciones y eliminaciones en cascada en tablas relacionadas de una base de datos.

Observe que muchas bases de datos creadas con versiones anteriores de SQL Server pueden contener este tipo de desencadenadores.

Los desencadenadores son adecuados para mantener la integridad de los datos en el nivel inferior, pero *no* para obtener resultados de consultas. La ventaja principal de los desencadenadores consiste en que pueden contener lógica compleja de proceso. Los desencadenadores pueden hacer cambios en cascada en tablas relacionadas de una base de datos, exigir integridad de datos más compleja que una restricción CHECK, definir mensajes de error personalizados, mantener datos *no normalizados* y comparar el estado de los datos antes y después de su modificación.

Cambios en cascada en tablas relacionadas de una base de datos

Los desencadenadores se pueden utilizar para hacer actualizaciones y eliminaciones en cascada en tablas relacionadas de una base de datos. Por ejemplo, un desencadenador de eliminación en la tabla **Products** de la base de datos **Northwind** puede eliminar de otras tablas las filas que tengan el mismo valor que la fila **ProductID** eliminada. Para ello, el desencadenador utiliza la columna de clave externa **ProductID** como una forma de ubicar las filas de la tabla **Order Details**.

Exigir una integridad de datos más compleja que una restricción CHECK

A diferencia de las restricciones CHECK, los desencadenadores pueden hacer referencia a columnas de otras tablas. Por ejemplo, podría colocar un desencadenador de inserción en la tabla **Order Details** que compruebe la columna **UnitsInStock** de ese artículo en la tabla **Products**. El desencadenador podría determinar que, cuando el valor **UnitsInStock** sea menor de 10, la cantidad máxima de pedido sea tres artículos. Este tipo de comprobación hace referencia a columnas de otras tablas. Con una restricción CHECK esto no se permite.

Los desencadenadores pueden utilizarse para exigir la integridad referencial de las siguientes formas:

- Realización de actualizaciones o eliminaciones directas o en cascada.

  La integridad referencial puede definirse con las restricciones FOREIGN KEY y REFERENCE en la instrucción CREATE TABLE. Los desencadenadores son útiles para asegurar la realización de las acciones adecuadas cuando deban efectuarse eliminaciones o actualizaciones en cascada. Si hay restricciones en la tabla del desencadenador, se comprueban antes de la ejecución del mismo. Si se infringen las restricciones, el desencadenador no se ejecuta.

- Creación de desencadenadores para varias filas

  Si se insertan, actualizan o eliminan varias filas, debe escribir un desencadenador que se ocupe de estos cambios múltiples.

- Exigir la integridad referencial entre bases de datos.

Definición de mensajes de error personalizados

En ocasiones, una aplicación puede mejorarse con mensajes de error personalizados que indiquen el estado de una acción. Los desencadenadores permiten invocar mensajes de error personalizados predefinidos o dinámicos cuando se den determinadas condiciones durante la ejecución del desencadenador.

Mantenimiento de datos no normalizados

Los desencadenadores se pueden utilizar para mantener la integridad en el nivel inferior de los entornos de base de datos no normalizados. El mantenimiento de datos no normalizados difiere de los cambios en cascada en que, por lo general, éstos hacen referencia al mantenimiento de relaciones entre valores de claves principales y externas. Habitualmente, los datos no normalizados contienen valores calculados, derivados o redundantes. Se debe utilizar un desencadenador en las situaciones siguientes:

- La integridad referencial requiere algo distinto de una correspondencia exacta, como mantener datos derivados (ventas del año hasta la fecha) o columnas indicadoras (S o N para indicar si un producto está disponible).
- Son necesarios mensajes personalizados e información de errores compleja.

Nota Normalmente, los datos redundantes y derivados requieren el uso de desencadenadores.

Comparación del estado de los datos antes y después de su modificación

La mayor parte de los desencadenadores permiten hacer referencia a los cambios efectuados a los datos con las instrucciones INSERT, UPDATE o DELETE. Esto permite hacer referencia a las filas afectadas por las instrucciones de modificación en el desencadenador.

Nota Las restricciones, reglas y valores predeterminados sólo pueden comunicar los errores a través de los mensajes de error estándar del sistema. Si la aplicación requiere mensajes de error personalizados y un tratamiento de errores más complejo (o mejoraría con ellos), debe utilizar un desencadenador.

Consideraciones acerca del uso de desencadenadores

Objetivo del tema
Explicar diversos aspectos que los alumnos deben tener en cuenta al utilizar desencadenadores.

Explicación previa
Al trabajar con desencadenadores, tenga en cuenta los siguientes hechos e instrucciones.

- Los desencadenadores son reactivos, mientras que las restricciones son proactivas
- Las restricciones se comprueban antes
- Las tablas pueden tener varios desencadenadores para cualquier acción
- Los propietarios de las tablas pueden designar el primer y último desencadenador que se debe activar
- Debe tener permiso para ejecutar todas las instrucciones definidas en los desencadenadores
- Los propietarios de tablas no pueden crear desencadenadores AFTER en vistas o en tablas temporales

Sugerencia
Asegúrese de tratar *todos* los puntos de la lista, incluso si no aparecen en la diapositiva.

Al trabajar con desencadenadores, tenga en cuenta los siguientes hechos e instrucciones:

- La mayor parte de los desencadenadores son reactivos; las restricciones y el desencadenador INSTEAD OF son proactivos.

  Los desencadenadores se ejecutan después de la ejecución de una instrucción INSERT, UPDATE o DELETE en la tabla en la que están definidos. Por ejemplo, si una instrucción UPDATE actualiza una fila de una tabla, el desencadenador de esa tabla se ejecuta automáticamente. Las restricciones se comprueban antes de la ejecución de la instrucción INSERT, UPDATE o DELETE.

- Las restricciones se comprueban primero.

  Si hay restricciones en la tabla del desencadenador, se comprueban antes de la ejecución del mismo. Si se infringen las restricciones, el desencadenador no se ejecuta.

- Las tablas pueden tener varios desencadenadores para cualquier acción.

  SQL Server 2000 permite anidar varios desencadenadores en una misma tabla. Una tabla puede tener definidos múltiples desencadenadores. Cada uno de ellos puede definirse para una sola acción o para varias.

- Los propietarios de las tablas pueden designar el primer y último desencadenador que se debe activar.

  Cuando se colocan varios desencadenadores en una tabla, su propietario puede utilizar el procedimiento almacenado del sistema **sp_settriggerorder** para especificar el primer y último desencadenador que se debe activar. El orden de activación de los demás desencadenadores no se puede establecer.

- Debe tener permiso para ejecutar todas las instrucciones definidas en los desencadenadores.

  Sólo el propietario de la tabla, los miembros de la función fija de servidor **sysadmin** y los miembros de las funciones fijas de base de datos **db_owner** y **db_ddladmin** pueden crear y eliminar desencadenadores de esa tabla. Estos permisos no pueden transferirse.

  Además, el creador del desencadenador debe tener permiso para ejecutar todas las instrucciones en todas las tablas afectadas. Si no tiene permiso para ejecutar alguna de las instrucciones de Transact-SQL contenidas en el desencadenador, toda la transacción se deshace.

- Los propietarios de tablas no pueden crear desencadenadores AFTER en vistas o en tablas temporales. Sin embargo, los desencadenadores pueden hacer referencia a vistas y tablas temporales.

- Los propietarios de las tablas pueden crear desencadenadores INSTEAD OF en vistas y tablas, con lo que se amplía enormemente el tipo de actualizaciones que puede admitir una vista.

Sugerencia
Recuerde a los alumnos que los desencadenadores no devuelven conjuntos de resultados ni pasan parámetros.

- Los desencadenadores no deben devolver conjuntos de resultados.

  Los desencadenadores contienen instrucciones de Transact-SQL del mismo modo que los procedimientos almacenados. Al igual que éstos, los desencadenadores pueden contener instrucciones que devuelven un conjunto de resultados. Sin embargo, esto no se recomienda porque los usuarios o programadores no esperan ver ningún conjunto de resultados cuando ejecutan una instrucción UPDATE, INSERT o DELETE.

- Los desencadenadores pueden tratar acciones que impliquen a múltiples filas.

  Una instrucción INSERT, UPDATE o DELETE que invoque a un desencadenador puede afectar a varias filas. En tal caso, puede elegir entre:

  - Procesar todas las filas juntas, con lo que todas las filas afectadas deberán cumplir los criterios del desencadenador para que se produzca la acción.
  - Permitir acciones condicionales.

    Por ejemplo, si desea eliminar tres clientes de la tabla **Customers**, puede definir un desencadenador que asegure que no queden pedidos activos ni facturas pendientes para cada cliente eliminado. Si uno de los tres clientes tiene una factura pendiente, no se eliminará, pero los demás que cumplan la condición sí.

  Para determinar si hay varias filas afectadas, puede utilizar la función del sistema @@ROWCOUNT.

Definición de desencadenadores

Objetivo del tema
Presentar los temas acerca de la creación, alteración y eliminación de desencadenadores tratados en esta sección.

Explicación previa
Ahora que sabe qué son los desencadenadores, veremos cómo crearlos, alterarlos y quitarlos.

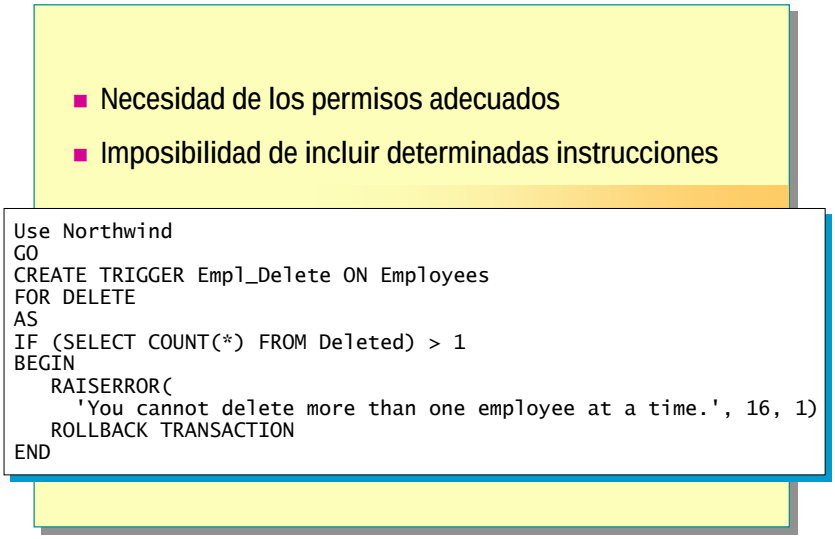
- Creación de desencadenadores
- Alteración y eliminación de desencadenadores

Esta sección trata la creación, alteración y eliminación de los desencadenadores. También se describen los permisos necesarios y las instrucciones que hay que tener en cuenta al definir desencadenadores.

Creación de desencadenadores

Objetivo del tema
Presentar la sintaxis de CREATE TRIGGER.

Explicación previa
Al crear desencadenadores, tenga en cuenta estos hechos e instrucciones.



Los desencadenadores se crean con la instrucción CREATE TRIGGER. Esta instrucción especifica la tabla en la que se define el desencadenador, los sucesos para los que se ejecuta y las instrucciones que contiene.

Sintaxis

```
CREATE TRIGGER [propietario.] nombreDesencadenador
  ON [propietario.] nombreTabla
  [WITH ENCRYPTION]
  {FOR | AFTER | INSTEAD OF} {INSERT | UPDATE | DELETE}
  AS
  [IF UPDATE (nombreColumna)...]
    [{AND | OR} UPDATE (nombreColumna)...]
    instruccionesSQL}
```

Cuando se especifica una acción FOR UPDATE, la cláusula IF UPDATE (*nombreColumna*) permite centrar la acción en una columna específica que se actualice.

Tanto FOR como AFTER son sintaxis equivalentes que crean el mismo tipo de desencadenador, que se activa después de la acción (INSERT, UPDATE o DELETE) que ha iniciado el desencadenador.

Los desencadenadores INSTEAD OF cancelan la acción desencadenante y realizan una nueva función en su lugar.

Al crear un desencadenador, la información acerca del mismo se inserta en las tablas del sistema **sysobjects** y **syscomments**. Si se crea un desencadenador con el mismo nombre que uno existente, el nuevo reemplazará al original.

Nota SQL Server no permite agregar desencadenadores definidos por el usuario a las tablas del sistema.

Necesidad de los permisos adecuados

Los propietarios de las tablas y los miembros de las funciones de propietario de base de datos (**db_owner**) y administradores del sistema (**sysadmin**) tienen permiso para crear desencadenadores.

Para evitar situaciones en las que el propietario de una vista y el propietario de las tablas subyacentes sean distintos, se recomienda que el usuario **dbo** (propietario de base de datos) sea el propietario de todos los objetos de la base de datos. Como un usuario puede ser miembro de varias funciones, debe especificar siempre el usuario **dbo** como propietario al crear el objeto. En caso contrario, el objeto se creará con su nombre de usuario como propietario.

Imposibilidad de incluir determinadas instrucciones

SQL Server no permite utilizar las instrucciones siguientes en la definición de un desencadenador:

- ALTER DATABASE
- CREATE DATABASE
- DISK INIT
- DISK RESIZE
- DROP DATABASE
- LOAD DATABASE
- LOAD LOG
- RECONFIGURE
- RESTORE DATABASE
- RESTORE LOG

Para conocer qué tablas tienen desencadenadores, ejecute el procedimiento almacenado del sistema **sp_depends** <*nombreTabla*>. Para ver la definición de un desencadenador, ejecute el procedimiento almacenado del sistema **sp_helptext** <*nombreDesencadenador*>. Para determinar los desencadenadores que hay en una tabla específica y sus acciones respectivas, ejecute el procedimiento almacenado del sistema **sp_helptrigger** <*nombreTabla*>.

Ejemplo

En el ejemplo siguiente se crea un desencadenador en la tabla **Employees** que impide que los usuarios puedan eliminar varios empleados a la vez. El desencadenador se activa cada vez que se elimina un registro o grupo de registros de la tabla. El desencadenador comprueba el número de registros que se están eliminando mediante la consulta de la tabla **Deleted**. Si se está eliminando más de un registro, el desencadenador devuelve un mensaje de error personalizado y deshace la transacción.

```
Use Northwind
GO

CREATE TRIGGER Empl_Delete ON NewEmployees
FOR DELETE
AS
IF (SELECT COUNT(*) FROM Deleted) > 1
BEGIN
   RAISERROR(
      'You cannot delete more than one employee at a time.',
       16, 1)
   ROLLBACK TRANSACTION
END
```

La instrucción DELETE siguiente activa el desencadenador y evita la transacción.

```
DELETE FROM Employees WHERE EmployeeID > 6
```

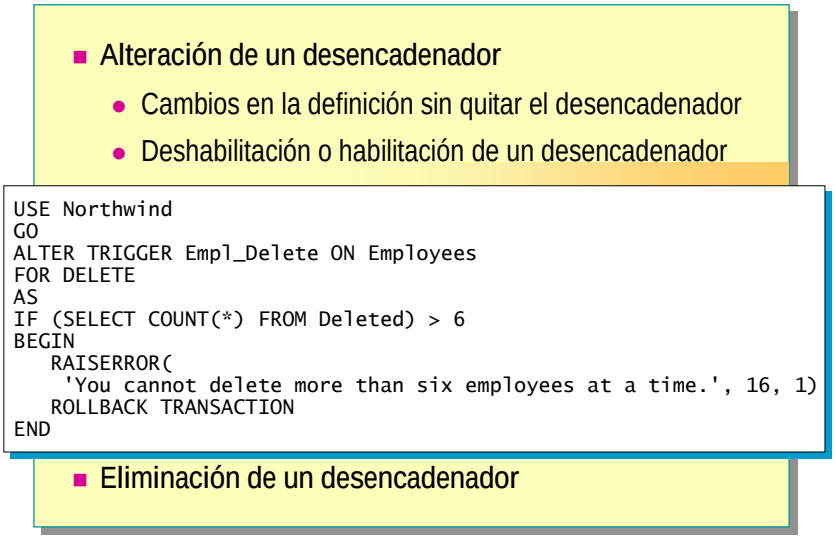
La instrucción DELETE siguiente activa el desencadenador y permite la transacción.

```
DELETE FROM Employees WHERE EmployeeID = 6
```

Alteración y eliminación de desencadenadores

Objetivo del tema
Presentar el concepto de alteración de un desencadenador.

Explicación previa
Si debe cambiar la definición de un desencadenador existente, puede alterarlo sin necesidad de quitarlo.



Los desencadenadores se pueden alterar o quitar.

Alteración de un desencadenador

Si debe cambiar la definición de un desencadenador existente, puede alterarlo sin necesidad de quitarlo.

Cambios en la definición sin quitar el desencadenador

Al cambiar la definición se reemplaza la definición existente del desencadenador por la nueva. También es posible alterar la acción del desencadenador. Por ejemplo, si crea un desencadenador para INSERT y, posteriormente, cambia la acción por UPDATE, el desencadenador modificado se ejecutará siempre que se actualice la tabla.

La resolución diferida de nombres permite que en un desencadenador haya referencias a tablas y vistas que aún no existen. Si el objeto no existe en el momento de crear el desencadenador, aparecerá un mensaje de advertencia y SQL Server actualizará la definición del desencadenador inmediatamente.

Sintaxis

```
ALTER TRIGGER nombreDesencadenador
  ON tabla
  [WITH ENCRYPTION]
  {{FOR {[,] [DELETE] [,] [UPDATE] [,][INSERT]}
  [NOT FOR REPLICATION]
  AS
  instrucciónSQL [...n] }
  |
  {FOR {[,] [INSERT] [,] [UPDATE]}
  [NOT FOR REPLICATION]
  AS
  IF UPDATE (columna)
  [{AND | OR} UPDATE (columna) [,...n]]
  instrucciónSQL [...n] }
  }
```

Ejemplo

En este ejemplo se altera el desencadenador de eliminación creado en el ejemplo anterior. Se suministra nuevo contenido para el desencadenador que cambia el límite de eliminación de uno a seis registros.

```
Use Northwind
GO
ALTER TRIGGER Empl_Delete ON Employees
FOR DELETE
AS
IF (SELECT COUNT(*) FROM Deleted) > 6
BEGIN
   RAISERROR(
     'You cannot delete more than six employees at a time.',
      16, 1)
   ROLLBACK TRANSACTION
END
```

Deshabilitación o habilitación de un desencadenador

Si lo desea, puede deshabilitar o habilitar un desencadenador específico de una tabla o todos los desencadenadores que haya en ella. Cuando se deshabilita un desencadenador, su definición se mantiene, pero la ejecución de una instrucción INSERT, UPDATE o DELETE en la tabla no activa la ejecución de las acciones del desencadenador hasta que éste se vuelva a habilitar.

Los desencadenadores se pueden habilitar o deshabilitar en la instrucción ALTER TABLE.

Sintaxis parcial

ALTER TABLE *tabla*
{ENABLE | DISABLE} TRIGGER
{ALL | *nombreDesencadenador*[**,**…*n*]}

Eliminación de un desencadenador

Si desea eliminar un desencadenador, puede quitarlo. Los desencadenadores se eliminan automáticamente cuando se elimina la tabla a la que están asociados.

De forma predeterminada, el permiso para eliminar un desencadenador corresponde al propietario de la tabla y no se puede transferir. Sin embargo, los miembros de las funciones de administradores del sistema (sysadmin) y propietario de la base de datos (db_owner) pueden eliminar cualquier objeto si especifican el propietario en la instrucción DROP TRIGGER.

Sintaxis

DROP TRIGGER *nombreDesencadenador*

Funcionamiento de los desencadenadores

Objetivo del tema
Presentar la sección que trata el funcionamiento de los desencadenadores.

Explicación previa
Examinemos cómo funcionan los distintos tipos de desencadenadores.

- Funcionamiento de un desencadenador INSERT
- Funcionamiento de un desencadenador DELETE
- Funcionamiento de un desencadenador UPDATE
- Funcionamiento de un desencadenador INSTEAD OF
- Funcionamiento de los desencadenadores anidados
- Desencadenadores recursivos

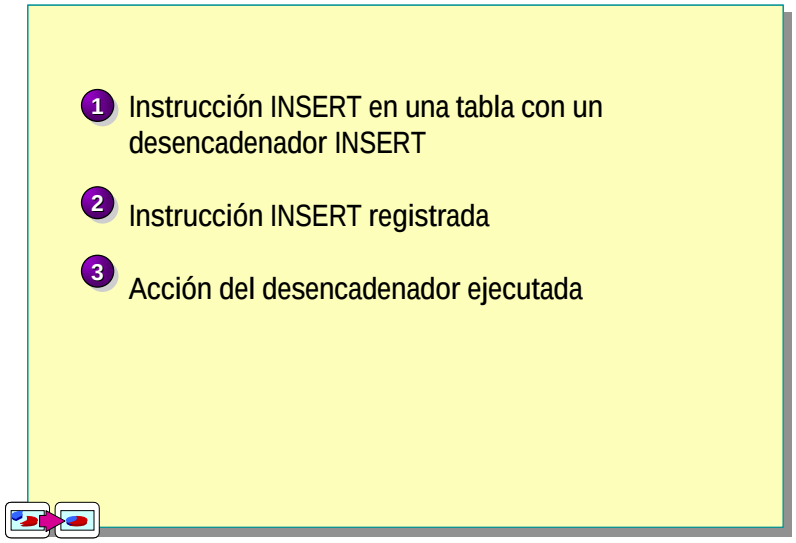
Cuando se diseñan desencadenadores, es importante comprender su funcionamiento. Esta sección trata los desencadenadores INSERT, DELETE, UPDATE, INSTEAD OF, anidados y recursivos.

Funcionamiento de un desencadenador INSERT

Objetivo del tema
Mostrar un ejemplo de un desencadenador INSERT.

Explicación previa
Un desencadenador INSERT se invoca cuando se intenta insertar una fila en una tabla que el desencadenador protege.

Todas las inserciones se registran en una tabla especial llamada **inserted**, como muestra la diapositiva.



Sugerencia
Indique cómo leer las instrucciones CREATE TRIGGER rápidamente para buscar el nombre del desencadenador, la tabla y la acción. Cuando se ejecuta, el desencadenador lleva a cabo todo lo que se encuentre a continuación de la instrucción AS.

Puede definir un desencadenador de modo que se ejecute siempre que una instrucción INSERT inserte datos en una tabla.

Cuando se activa un desencadenador INSERT, las nuevas filas se agregan a la tabla del desencadenador y a la tabla **inserted**. Se trata de una tabla lógica que mantiene una copia de las filas insertadas. La tabla **inserted** contiene la actividad de inserción registrada proveniente de la instrucción INSERT. La tabla **inserted** permite hacer referencia a los datos registrados por la instrucción INSERT que ha iniciado el desencadenador. El desencadenador puede examinar la tabla **inserted** para determinar qué acciones debe realizar o cómo ejecutarlas. Las filas de la tabla **inserted** son siempre duplicados de una o varias filas de la tabla del desencadenador.

Se registra toda la actividad de modificación de datos (instrucciones INSERT, UPDATE y DELETE), pero la información del registro de transacciones es ilegible. Sin embargo, la tabla **inserted** permite hacer referencia a los cambios registrados provocados por la instrucción INSERT. Así, es posible comparar los cambios a los datos insertados para comprobarlos o realizar acciones adicionales. También se puede hacer referencia a los datos insertados sin necesidad de almacenarlos en variables.

Ejemplo

El desencadenador de este ejemplo se creó para actualizar una columna (**UnitsInStock**) de la tabla **Products** siempre que se pida un producto (siempre que se inserte un registro en la tabla **Order Details**). El nuevo valor se establece al valor anterior menos la cantidad pedida.

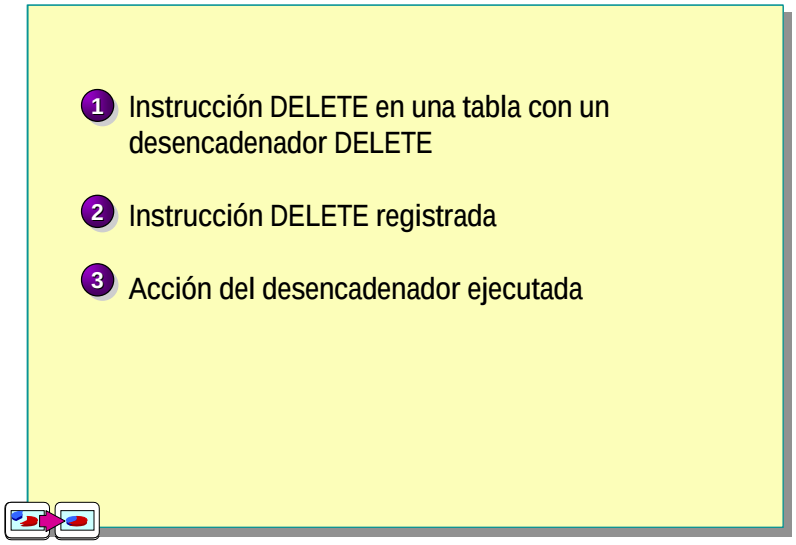
```
USE Northwind
GO
CREATE TRIGGER OrdDet_Insert
ON [Order Details]
FOR INSERT
AS
UPDATE P SET
UnitsInStock = (P.UnitsInStock – I.Quantity)
FROM Products AS P INNER JOIN Inserted AS I
ON P.ProductID = I.ProductID
```

Funcionamiento de un desencadenador DELETE

Objetivo del tema
Mostrar un ejemplo de un desencadenador DELETE.

Explicación previa
Un desencadenador DELETE se invoca siempre que se intenta eliminar información de la tabla en la que está definido.

Cuando se eliminan filas de una tabla, se agregan a una tabla especial llamada **deleted**, como muestra la diapositiva.



Cuando se activa un desencadenador DELETE, las filas eliminadas en la tabla afectada se agregan a una tabla especial llamada **deleted**. Se trata de una tabla lógica que mantiene una copia de las filas eliminadas. La tabla **deleted** permite hacer referencia a los datos registrados por la instrucción DELETE que ha iniciado la ejecución del desencadenador.

Al utilizar el desencadenador DELETE, tenga en cuenta los hechos siguientes:

- Cuando se agrega una fila a la tabla **deleted**, la fila deja de existir en la tabla de la base de datos, por lo que la tabla **deleted** y las tablas de la base de datos no tienen ninguna fila en común.
- Para crear la tabla **deleted** se asigna espacio de memoria. La tabla **deleted** está siempre en la caché.
- Los desencadenadores definidos para la acción DELETE no se ejecutan con la instrucción TRUNCATE TABLE, ya que TRUNCATE TABLE no se registra.

Ejemplo

El desencadenador de este ejemplo se creó para actualizar una columna **Discontinued** de la tabla **Products** cuando se elimine una categoría (cuando se elimine un registro de la tabla **Categories**). Todos los productos afectados se marcan con 1, lo que indica que ya no se suministran.

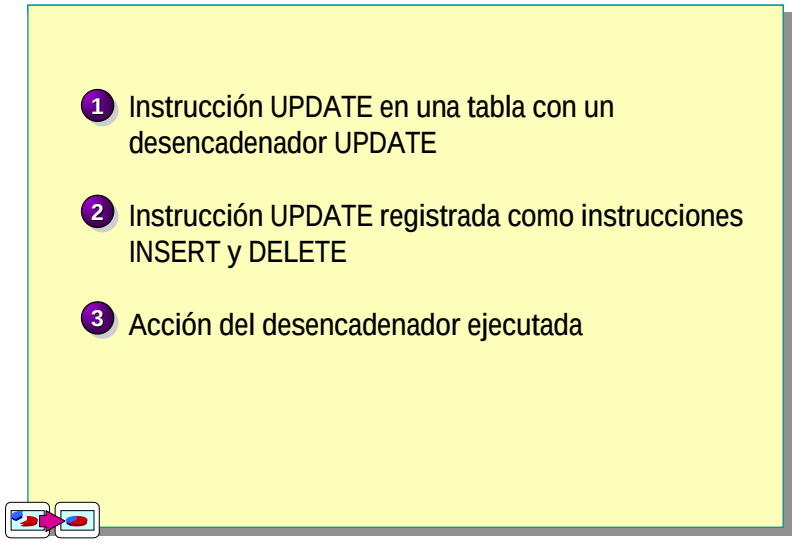
```
USE Northwind
GO
CREATE TRIGGER Category_Delete
  ON Categories
  FOR DELETE
AS
  UPDATE P SET Discontinued = 1
      FROM Products AS P INNER JOIN deleted AS d
      ON P.CategoryID = d.CategoryID
```

Funcionamiento de un desencadenador UPDATE

Objetivo del tema
Ofrecer un ejemplo de un desencadenador UPDATE.

Explicación previa
Un desencadenador definido para una instrucción UPDATE se invoca siempre que se intenta actualizar datos de la tabla en la que está definido.

La instrucción UPDATE mueve la fila original a la tabla **deleted** e inserta la fila actualizada en la tabla **inserted**, como muestra el gráfico.



Se puede considerar que una instrucción UPDATE está formada por dos pasos: el paso DELETE que captura la *imagen anterior* de los datos y el paso INSERT que captura la *imagen posterior*. Cuando se ejecuta una instrucción UPDATE en una tabla que tiene definido un desencadenador, las filas originales (imagen anterior) se mueven a la tabla **deleted** y las filas actualizadas (imagen posterior) se agregan a la tabla **inserted**.

El desencadenador puede examinar las tablas **deleted** e **inserted** así como la tabla actualizada, para determinar si se han actualizado múltiples filas y cómo debe ejecutar las acciones oportunas.

Para definir un desencadenador que supervise las actualizaciones de los datos de una columna específica puede utilizar la instrucción IF UPDATE. De este modo, el desencadenador puede aislar fácilmente la actividad de una columna específica. Cuando detecte una actualización en esa columna, realizará las acciones apropiadas, como mostrar un mensaje de error que indique que la columna no se puede actualizar o procesar un conjunto de instrucciones en función del nuevo valor de la columna.

Sintaxis

IF UPDATE (<*nombreColumna*>)

Ejemplo 1

En este ejemplo se evita que un usuario modifique la columna **EmployeeID** de la tabla **Employees**.

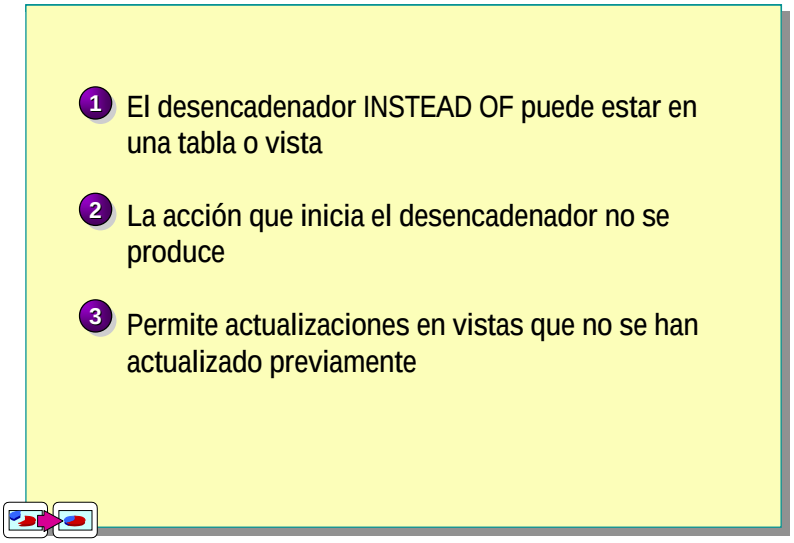
Sugerencia
El carácter barra diagonal inversa (\) de la instrucción RAISERROR es un indicador de continuación que permite que todo el mensaje de error aparezca en la misma línea.

```
USE Northwind
GO
CREATE TRIGGER Employee_Update
  ON Employees
  FOR UPDATE
AS
IF UPDATE (EmployeeID)
BEGIN TRANSACTION
  RAISERROR ('Transaction cannot be processed.\
  ***** Employee ID number cannot be modified.', 10, 1)
  ROLLBACK TRANSACTION
```

Funcionamiento de un desencadenador INSTEAD OF

Objetivo del tema
Mostrar un ejemplo de un desencadenador INSTEAD OF.

Explicación previa
Los desencadenadores INSTEAD OF cancelan la acción desencadenante original y realizan su propia función en su lugar.



Puntos clave
Compare un desencadenador INSTEAD OF con un desencadenador AFTER.

Señale que, cuando se utiliza un desencadenador INSTEAD OF, la acción desencadenante original (INSERT, UPDATE o DELETE) *no* se produce. Observe que puede colocar un desencadenador INSTEAD OF en tablas y vistas.

Un desencadenador INSTEAD OF se puede especificar en tablas y vistas. Este desencadenador se ejecuta en lugar de la acción desencadenante original. Los desencadenadores INSTEAD OF aumentan la variedad de tipos de actualizaciones que se pueden realizar en una vista. Cada tabla o vista está limitada a un desencadenador INSTEAD OF por cada acción desencadenante (INSERT, UPDATE o DELETE).

No se puede crear un desencadenador INSTEAD OF en vistas que tengan definido WITH CHECK OPTION.

Ejemplo

En este ejemplo se crea una tabla con clientes de Alemania (Germany) y una tabla con clientes de México. Mediante un desencadenador INSTEAD OF colocado en la vista se redirigen las actualizaciones a la tabla subyacente apropiada. Se produce la inserción en la tabla **CustomersGer** *en lugar de* la inserción en la vista.

Sugerencia
En este ejemplo se crean las tablas CustomersGer y CustomersMex y, a continuación, una vista denominada CustomersView.

Muestre cómo se produce un error en la actualización de la vista. A continuación, cree el desencadenador y muestre cómo redirige la actualización.

Cree dos tablas con datos de clientes

```
SELECT * INTO CustomersGer FROM Customers WHERE
Customers.Country = 'Germany'
SELECT * INTO CustomersMex FROM Customers WHERE
Customers.Country = 'Mexico'
GO
```

Cree una vista en esos datos

```
CREATE VIEW CustomersView AS
SELECT * FROM CustomersGer
UNION
SELECT * FROM CustomersMex
GO
```

Cree un desencadenador INSTEAD OF en la vista

```
CREATE TRIGGER Customers_Update2
ON CustomersView
INSTEAD OF UPDATE AS
DECLARE @Country nvarchar(15)
SET @Country = (SELECT Country FROM Inserted)
IF @Country = 'Germany'
  BEGIN
   UPDATE CustomersGer
    SET CustomersGer.Phone = Inserted.Phone
    FROM CustomersGer JOIN Inserted
    ON CustomersGer.CustomerID = Inserted.CustomerID
  END
ELSE
  IF @Country = 'Mexico'
  BEGIN
   UPDATE CustomersMex
    SET CustomersMex.Phone = Inserted.Phone
    FROM CustomersMex JOIN Inserted
    ON CustomersMex.CustomerID = Inserted.CustomerID
  END
```

Si los alumnos preguntan por qué aparentemente esta secuencia de comandos actualizó CustomersView, indique que la vista obtiene la información de la tabla CustomerGer.

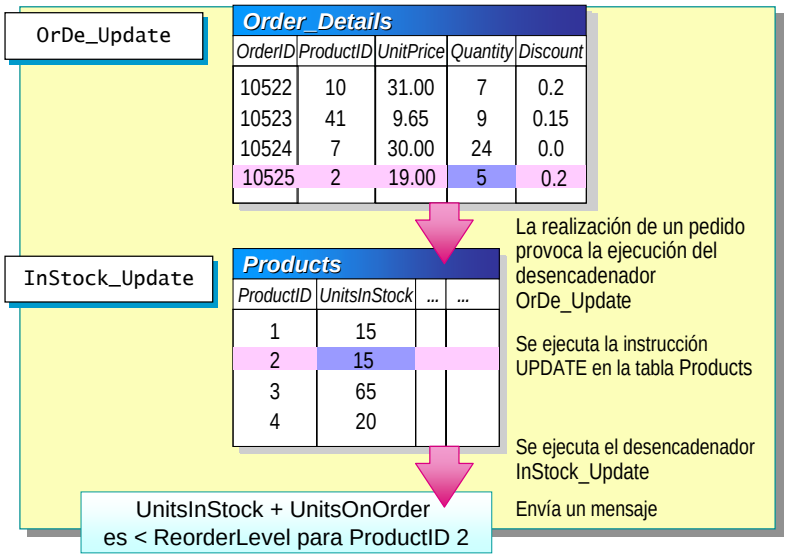
Pruebe el desencadenador mediante la actualización de la vista

```
UPDATE CustomersView SET Phone = ' 030-007xxxx'
  WHERE CustomerID = 'ALFKI'
SELECT CustomerID, Phone FROM CustomersView
  WHERE CustomerID = 'ALFKI'
SELECT CustomerID, Phone FROM CustomersGer
  WHERE CustomerID = 'ALFKI'
```

Funcionamiento de los desencadenadores anidados

Objetivo del tema
Explicar el uso de los desencadenadores anidados.

Explicación previa
Como se mencionó anteriormente, los desencadenadores pueden anidarse hasta 32 niveles. Si los desencadenadores anidados están habilitados, un desencadenador que modifique una tabla podrá activar un segundo desencadenador, que a su vez podrá activar un tercero, y así sucesivamente.



Cualquier desencadenador puede contener una instrucción UPDATE, INSERT o DELETE que afecte a otra tabla. Cuando el anidamiento está habilitado, un desencadenador que cambie una tabla podrá activar un segundo desencadenador, que a su vez podrá activar un tercero y así sucesivamente. El anidamiento se habilita durante la instalación, pero se puede deshabilitar y volver a habilitar con el procedimiento almacenado del sistema **sp_configure**.

Los desencadenadores pueden anidarse hasta 32 niveles. Si un desencadenador de una cadena anidada provoca un bucle infinito, se superará el nivel de anidamiento. Por lo tanto, el desencadenador terminará y deshará la transacción. Los desencadenadores anidados pueden utilizarse para realizar funciones como almacenar una copia de seguridad de las filas afectadas por un desencadenador anterior. Al utilizar desencadenadores anidados, tenga en cuenta los siguientes hechos:

- De forma predeterminada, la opción de configuración de desencadenadores anidados está activada.
- Un desencadenador anidado no se activará dos veces en la misma transacción; un desencadenador no se llama a sí mismo en respuesta a una segunda actualización de la misma tabla en el desencadenador. Por ejemplo, si un desencadenador modifica una tabla que, a su vez, modifica la tabla original del desencadenador, éste no se vuelve a activar.
- Los desencadenadores son transacciones, por lo que un error en cualquier nivel de un conjunto de desencadenadores anidados cancela toda la transacción y las modificaciones a los datos se deshacen. Por tanto, se recomienda incluir instrucciones PRINT al probar los desencadenadores para determinar dónde se producen errores.

Sugerencia
La función @@NESTLEVEL es útil para probar y solucionar problemas de desencadenadores, pero normalmente no se utiliza en un entorno de producción.

Comprobación del nivel de anidamiento

Cada vez que se activa un desencadenador anidado, el nivel de anidamiento se incrementa. SQL Server admite hasta 32 niveles de anidamiento, pero puede ser conveniente limitar los niveles para evitar exceder el máximo. La función @@NESTLEVEL permite ver el nivel actual de anidamiento.

Conveniencia del uso del anidamiento

El anidamiento es una característica eficaz que puede utilizar para mantener la integridad de la información de una base de datos. Sin embargo, en ocasiones puede considerar conveniente deshabilitarlo. Si el anidamiento está deshabilitado, un desencadenador que modifique otra tabla no invocará ninguno de los desencadenadores de esa tabla.

Para deshabilitar el anidamiento, utilice la instrucción siguiente:

Sintaxis

sp_configure 'nested triggers', 0

Las siguientes son algunas razones por las que podría decidir deshabilitar el anidamiento:

- Los desencadenadores anidados requieren un diseño complejo y bien planeado. Los cambios en cascada pueden modificar datos que no se deseaba cambiar.
- Una modificación de datos en cualquier punto de un conjunto de desencadenadores anidados activa todos los desencadenadores. Aunque esto supone una protección eficaz de los datos, puede ser un problema si las tablas deben actualizarse en un orden específico.

Es posible conseguir la misma funcionalidad con y sin la característica de anidamiento; sin embargo, el diseño de los desencadenadores será sustancialmente distinto. Al diseñar desencadenadores anidados, cada desencadenador sólo debe iniciar la siguiente modificación de los datos, por lo que el diseño será modular. En el diseño sin anidamiento, cada desencadenador tiene que iniciar todas las modificaciones de datos que deba realizar.

Ejemplo

En este ejemplo se muestra cómo la realización de un pedido provoca la ejecución del desencadenador **OrDe_Update**. Este desencadenador ejecuta una instrucción UPDATE en la columna **UnitsInStock** de la tabla **Products**. Cuando se produce la actualización, se activa el desencadenador **Products_Update** y compara el nuevo valor de las existencias en inventario, más las existencias en pedido, con el nivel de reabastecimiento. Si las existencias en inventario más las pedidas se encuentran por debajo del nivel de reabastecimiento, se envía un mensaje que alerta sobre la necesidad de comprar más existencias.

```
USE Northwind
GO
CREATE TRIGGER Products_Update
  ON Products
  FOR UPDATE
AS
IF UPDATE (UnitsInStock)
  IF (Products.UnitsInStock + Products.UnitsOnOrder) <
Products.ReorderLevel
BEGIN
  --Enviar mensaje al departamento de compras
END
```

Desencadenadores recursivos

Objetivo del tema
Explicar el uso de los desencadenadores recursivos.

Explicación previa
Cuando la opción de desencadenadores recursivos está habilitada, un desencadenador que cambie datos en una tabla puede activar un segundo desencadenador que, a su vez puede, activar el primer desencadenador al modificar datos de la tabla original.

- Activación recursiva de un desencadenador
- Tipos de desencadenadores recursivos
  - *Recursividad directa*, que se da cuando un desencadenador se ejecuta y realiza una acción que lo activa de nuevo
  - *Recursividad indirecta*, que se da cuando un desencadenador se activa y realiza una acción que activa un desencadenador de otra tabla
- Conveniencia del uso de los desencadenadores recursivos

Cualquier desencadenador puede contener una instrucción UPDATE, INSERT o DELETE que afecte a la misma tabla o a otra distinta. Cuando la opción de desencadenadores recursivos está habilitada, un desencadenador que cambie datos de una tabla puede activarse de nuevo a sí mismo, en ejecución recursiva. Esta opción se deshabilita de forma predeterminada al crear una base de datos, pero puede habilitarla con la instrucción ALTER DATABASE.

Activación recursiva de un desencadenador

Para habilitar los desencadenadores recursivos, utilice la instrucción siguiente:

Sintaxis

```
ALTER DATABASE ClassNorthwind SET RECURSIVE_TRIGGERS ON

sp_dboption nombreBaseDatos, 'recursive triggers', True
```

Nota Utilice el procedimiento almacenado del sistema **sp_settriggerorder** para especificar un desencadenador que se active como primer desencadenador AFTER o como último desencadenador AFTER. Cuando se han definido varios desencadenadores para un mismo suceso, su ejecución no sigue un orden determinado. Cada desencadenador debe ser autocontenido.

Si la opción de desencadenadores anidados está desactivada, la de desencadenadores recursivos también lo estará, sin importar la configuración de desencadenadores recursivos de la base de datos.

Las tablas **inserted** y **deleted** de un desencadenador dado sólo contienen las filas correspondientes a la instrucción UPDATE, INSERT o DELETE que lo invocó la última vez.

La recursividad de desencadenadores puede llegar hasta 32 niveles. Si un desencadenador provoca un bucle recursivo infinito, se superará el nivel de anidamiento, por lo que el desencadenador terminará y se deshará la transacción.

Tipos de desencadenadores recursivos

Hay dos tipos de recursividad distintos:

- Recursividad directa, que se da cuando un desencadenador se ejecuta y realiza una acción que lo activa de nuevo.

  Por ejemplo, una aplicación actualiza la tabla **T1**, lo que hace que se ejecute **Desen1**. **Desen1** actualiza de nuevo la tabla **T1**, con lo que **Desen1** se activa una vez más.

- Recursividad indirecta, que se da cuando un desencadenador se activa y realiza un acción que activa un desencadenador de otra tabla, que a su vez causa una actualización de la tabla original. De este modo, el desencadenador original se activa de nuevo.

  Por ejemplo, una aplicación actualiza la tabla **T2**, lo que hace que se ejecute **Desen2**. **Desen2** actualiza la tabla **T3**, con lo que **Desen3** se activa una vez más. A su vez, **Desen3** actualiza la tabla **T2**, de modo que **Desen2** se activa de nuevo.

Conveniencia del uso de los desencadenadores recursivos

Los desencadenadores recursivos son una característica compleja que se puede utilizar para resolver relaciones complejas, como las de autorreferencia (conocidas también como *cierres transitivos*). En estas situaciones especiales, puede ser conveniente habilitar los desencadenadores recursivos.

Sugerencia
Señale que el primer ejemplo no se refiere ni se aplica a la tabla Employees de la base de datos Northwind.

Los desencadenadores recursivos pueden resultar útiles cuando se deba mantener:

- El número de columnas de informe de la tabla **employee**, donde la tabla contiene una columna **employee ID** y una columna **manager ID**.

  Por ejemplo, supongamos que en la tabla **employee** se han definido dos desencadenadores de actualización, **tr_update_employee** y **tr_update_manager**. El desencadenador **tr_update_employee** actualiza la tabla **employee**.

  Una instrucción UPDATE activa una vez **tr_update_employee** y también **tr_update_manager**. Además, la ejecución de **tr_update_employee** desencadena de nuevo (recursivamente) la activación de **tr_update_employee** y **tr_update_manager**.

- Un gráfico con datos de programación de producción cuando existe una jerarquía de programación implícita.
- Un sistema de seguimiento de composición en el que se hace un seguimiento de cada subcomponente hasta el conjunto del que forma parte.

Antes de utilizar desencadenadores recursivos, tenga en cuenta las instrucciones siguientes:

- Los desencadenadores recursivos son complejos y precisan un buen diseño y una prueba minuciosa. Además requieren código de lógica de control de bucle (comprobación de terminación). En caso contrario, se superará el límite de 32 niveles de anidamiento.
- Una modificación de datos en cualquier punto puede iniciar la serie de desencadenadores. Aunque esto permite procesar relaciones complejas, puede convertirse en un problema si las tablas se deben actualizar en un orden específico.

Es posible lograr la misma funcionalidad sin utilizar la característica de desencadenadores recursivos; sin embargo, el diseño de desencadenadores diferirá sustancialmente. Al diseñar desencadenadores recursivos, cada uno debe contener una comprobación condicional para detener el procesamiento recursivo cuando la condición sea falsa. Al diseñar desencadenadores no recursivos, cada desencadenador debe contener las estructuras completas de bucle de programación y comprobaciones.

◆ Ejemplos de desencadenadores

Objetivo del tema
Explicar la razón por la que los desencadenadores son necesarios en SQL Server.

Explicación previa
Los desencadenadores exigen la integridad referencial y aplican las reglas de empresa.

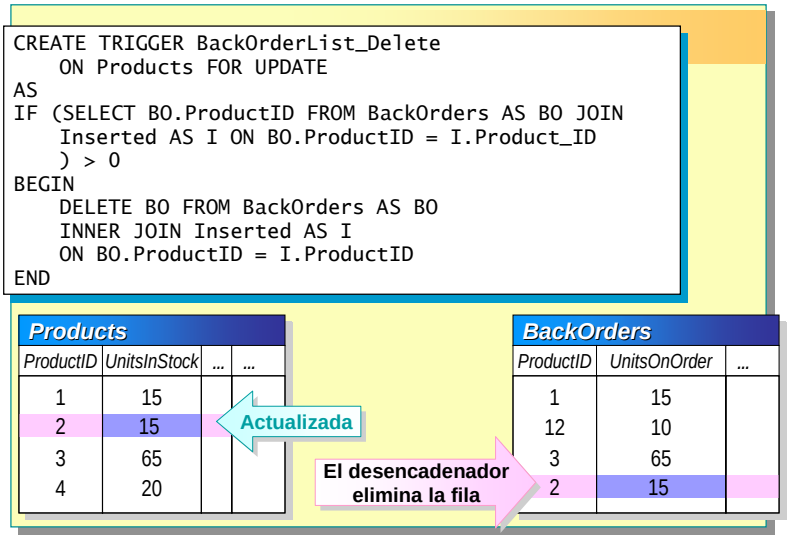
- Exigir la integridad de los datos
- Exigir reglas de empresa

Los desencadenadores exigen la integridad referencial y aplican las reglas de empresa. Algunas de las acciones que llevan a cabo los desencadenadores pueden realizarse también mediante restricciones y, en determinados casos, debe considerarse primero el uso de éstas. Sin embargo, los desencadenadores son necesarios para exigir diversos grados de carencia de normalización y para implementar reglas complejas de empresa.

Exigir la integridad de los datos

Objetivo del tema
Mostrar un ejemplo de cómo los desencadenadores exigen la integridad de los datos.

Explicación previa
Los desencadenadores pueden utilizarse para aplicar en cascada los cambios a las tablas relacionadas de toda la base de datos y mantener así la integridad de los datos.



Los desencadenadores pueden utilizarse para aplicar en cascada los cambios a las tablas relacionadas de toda la base de datos y mantener así la integridad de los datos.

Ejemplo

En el ejemplo siguiente se muestra cómo un desencadenador mantiene la integridad de los datos en la tabla **BackOrders**. El desencadenador **BackOrderList_delete** mantiene la lista de productos de la tabla **BackOrders**. Cuando se reciben productos, el desencadenador UPDATE de la tabla **Products** elimina registros de la tabla **BackOrders**.

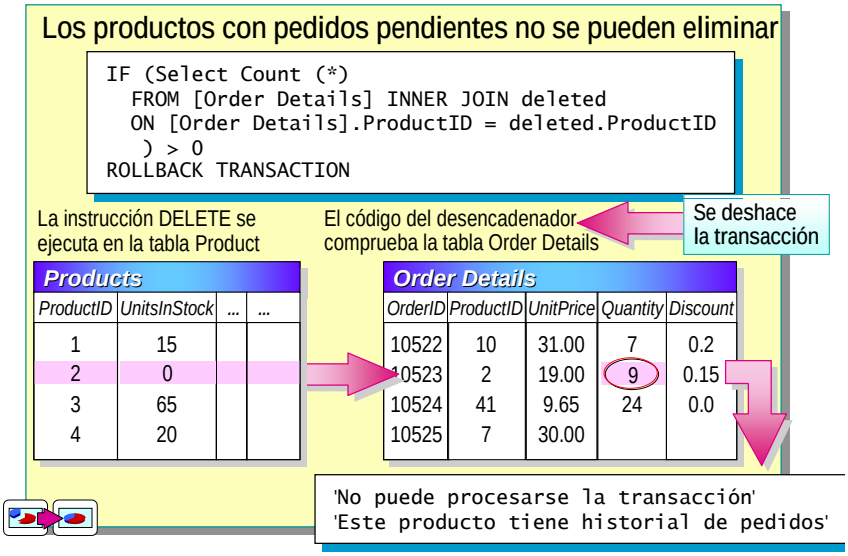
Para su información
Este ejemplo es hipotético. No existe la tabla BackOrders en la base de datos Northwind.

```
CREATE TRIGGER BackOrderList_Delete
  ON Products FOR UPDATE
AS
IF (SELECT BO.ProductID FROM BackOrders AS BO JOIN
  Inserted AS I ON BO.ProductID = I.Product_ID
  ) > 0
BEGIN
  DELETE BO FROM BackOrders AS BO
  INNER JOIN Inserted AS I
  ON BO.ProductID = I.ProductID
END
```

Exigir reglas de empresa

Objetivo del tema
Mostrar un ejemplo de cómo exigir las reglas de empresa.

Explicación previa
Los desencadenadores también permiten exigir reglas de empresa determinadas, como “No eliminar los productos con historial de pedidos”.



Puede utilizar los desencadenadores para exigir reglas de empresa que son demasiado complejas para la restricción CHECK. Esto incluye la comprobación del estado de las filas en otras tablas.

Por ejemplo, puede asegurarse de que las multas pendientes de un socio se paguen antes de permitirle darse de baja.

Ejemplo

Este ejemplo crea un desencadenador que determina si un producto tiene historial de pedidos. Si es así, la transacción DELETE se deshace y el desencadenador devuelve un mensaje de error.

```
Use Northwind
GO
CREATE TRIGGER Product_Delete
  ON Products FOR DELETE
AS
IF (Select Count (*)
    FROM [Order Details] INNER JOIN deleted
    ON [Order Details].ProductID = deleted.ProductID
    ) > 0
BEGIN
   RAISERROR('Transaction cannot be processed.\
              This product has order history.', 16, 1)
   ROLLBACK TRANSACTION
END
```

Consideraciones acerca del rendimiento

Objetivo del tema
Presentar una serie de consideraciones acerca del rendimiento que hay que tener en cuenta al utilizar desencadenadores.

Explicación previa
Cuando utilice desencadenadores, debe tener en cuenta los siguientes aspectos acerca del rendimiento.

- Los desencadenadores trabajan rápidamente porque las tablas insertadas y eliminadas están en la caché
- El tiempo de ejecución está determinado por:
  - Número de tablas a las que se hace referencia
  - Número de filas afectadas
- Las acciones contenidas en un desencadenador forman parte de una transacción

Cuando utilice desencadenadores, debe tener en cuenta los siguientes aspectos acerca del rendimiento:

- Los desencadenadores trabajan rápidamente porque las tablas **inserted** y **deleted** están en la memoria caché.

  Las tablas **inserted** y **deleted** siempre están en memoria y no en un disco, ya que son tablas lógicas y, normalmente, son muy pequeñas.
- El número de tablas a las que se hace referencia y el número de filas afectadas determina el tiempo de ejecución.

  El tiempo necesario para invocar un desencadenador es mínimo. La mayor parte del tiempo de ejecución se invierte en hacer referencia a otras tablas (que pueden estar en memoria o en un disco) y en modificar datos, si así lo especifica la definición del desencadenador.
- Las acciones contenidas en un desencadenador forman parte de una transacción.

  Una vez definido un desencadenador, la acción del usuario (instrucción INSERT, UPDATE o DELETE) en la tabla que lo activa es siempre, implícitamente, parte de una transacción, así como el propio desencadenador. Si se encuentra una instrucción ROLLBACK TRANSACTION, se deshará toda la transacción. Si en la secuencia de comandos del desencadenador hay instrucciones después de ROLLBACK TRANSACTION, también se ejecutarán. Por tanto, puede ser necesario utilizar una cláusula RETURN en una instrucción IF para impedir que se procesen las demás instrucciones.