Capítulo 11

Triggers

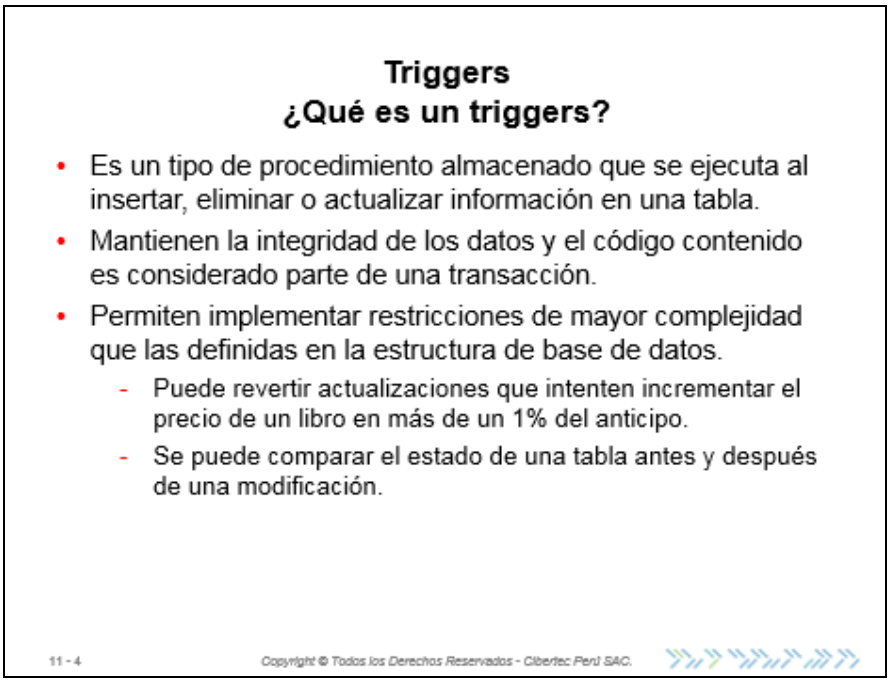
Al finalizar el capítulo, el alumno podrá:

- Identificar las formas de crear disparadores que validen reglas de negocio dentro de la base de datos.

Temas:

1. Definición
2. ¿Cómo funciona un trigger?
3. Empleo de procedimiento almacenados en triggers

1. Definición



1.1 Definición

Un disparador es un tipo especial de procedimiento almacenado, que se ejecuta cuando se insertan, eliminan o actualizan datos de una tabla especificada. Los disparadores pueden ayudar a mantener la integridad de referencia de los datos conservando la consistencia entre los datos relacionados, lógicamente de distintas tablas.

Integridad de referencia significa que los valores de las claves primarias y los valores correspondientes de las claves externas deben coincidir de forma exacta.

La principal ventaja de los disparadores es que son automáticos, es decir, funcionan cualquiera sea el origen de la modificación de los datos, una introducción de datos por parte de un empleado o una acción de una aplicación. Cada disparador es específico de una o más operaciones de modificación de datos, UPDATE, INSERT o DELETE. El disparador se ejecuta una vez por cada instrucción SQL.

Un disparador se "dispara" solo cuando la instrucción de modificación de datos finaliza y SQL Server verifica la posible violación de tipos de datos, reglas o restricciones de integridad. El disparador y la instrucción que lo "dispara" se consideran una sola transacción que puede revertirse desde dentro del disparador. Si se detecta un error grave, se revierte toda la transacción.

Existen situaciones en las que los disparadores son de mayor utilidad. Los disparadores pueden imponer restricciones de mucha mayor complejidad que las definidas con las reglas. Al contrario de lo que ocurre con las reglas, los disparadores pueden hacer referencia a columnas u objetos de base de datos.

Además, los disparadores pueden llevar a cabo análisis de hipótesis sencillos. Por ejemplo, un disparador puede comparar el estado de una tabla antes y después de una modificación de datos y llevar a cabo acciones basándose en esa comparación.

1.2 Tipos de Triggers

Existen los siguientes tipos de triggers:

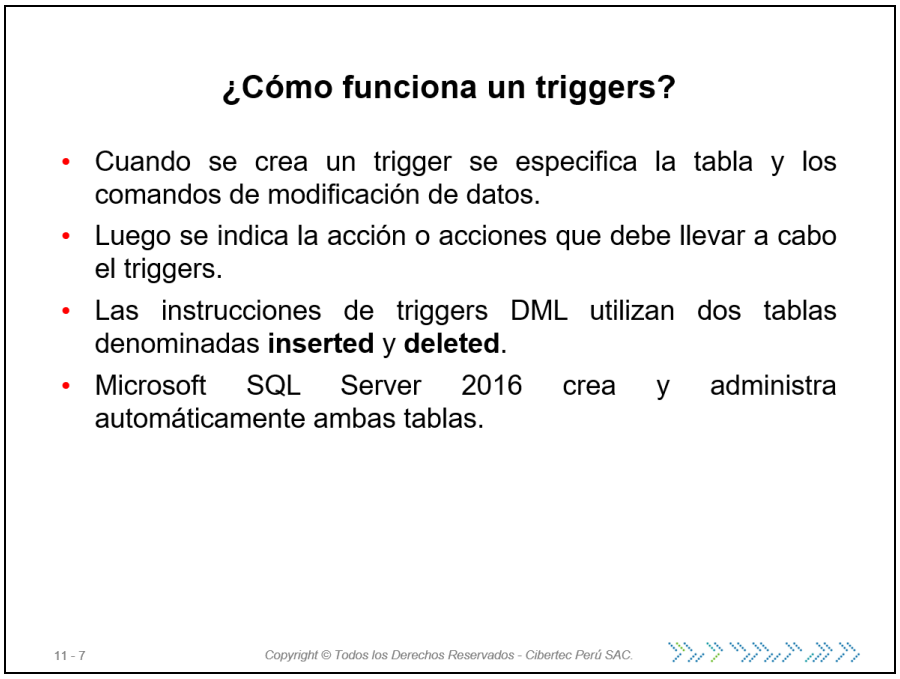
- Los triggers DML que se ejecutan cuando se realizan operaciones de manipulación de datos (DML). Los eventos DML son instrucciones INSERT, UPDATE o DELETE realizados en una tabla o en una vista.
- Los triggers DDL que se ejecutan al realizar eventos del lenguaje de definición de datos (DDL). Estos eventos corresponden a instrucciones CREATE, ALTER, DROP.
- Los triggers Logon, que se disparan al ejecutarse un inicio de sesión en SQL Server.

1.3 Consideraciones

Se deben tener en cuenta las siguientes consideraciones:

- Una tabla puede tener un máximo de tres triggers; uno de actualización, uno de inserción y uno de eliminación.
- Cada trigger puede aplicarse a una sola tabla o vista. Por otro lado, un mismo trigger se puede aplicar a las tres acciones: UPDATE, INSERT y DELETE.
- Los trigger no se permiten en las tablas del sistema.

2. ¿Cómo funciona un trigger?



2.1 Creando disparadores

Un disparador es un objeto de base de datos. Cuando se crea un disparador, se especifica la tabla y los comandos de modificación de datos que deben "disparar" o activar. Luego, se indica la acción o acciones que debe llevar a cabo el disparador.

A continuación, se muestra la sintaxis completa de create trigger:

```
CREATE [OR ALTER] TRIGGER [Propietario.]TRIGGER_NAME
    ON [ OWNER.]  TABLE_NAME
     {FOR {INSERT, UPDATE, DELETE}
   AS   SQL_EXPRESION

-- o bien, usando la cláusula IF UPDATE:

CREATE [OR ALTER] TRIGGER [Propietario.]TRIGGER_NAME
 ON [ Propietario.]TABLE_NAME
 FOR {INSERT, UPDATE}
 AS
     [ IF UPDATE (COLUMN_NAME)
         [{AND | OR} UPDATE (COLUMN_NAME)] ...]
           SQL_EXPRESION
       [IF UPDATE (COLUMN_NAME)
         [{AND | OR} UPDATE (COLUMN_NAME)] ...
           SQL_EXPRESION]...
```

La cláusula CREATE [OR ALTER] crea o modifica el disparador y le asigna un nombre. El nombre del disparador debe cumplir con las reglas para identificadores.

La cláusula ON indica el nombre de la tabla que activa el disparador. Esta tabla se denomina en ocasiones, tabla de disparadores.

Los disparadores se crean en la base de datos actual, aunque pueden hacer referencia a objetos de otras bases de datos. El nombre del propietario que califica el nombre de disparador debe ser el mismo que el de la tabla. Nadie, excepto el propietario de la tabla, puede crear un disparador en una tabla. Si se indica el propietario de la tabla con el nombre de tabla en la cláusula CREATE OR ALTER TRIGGER o la cláusula ON, también debe especificarse en la otra cláusula.

La cláusula FOR especifica los comandos de modificación de datos de la tabla de disparadores que activan el disparador.

Las instrucciones SQL indican las condiciones y acciones del disparador. Las condiciones del disparador especifican criterios adicionales que determinan si el comando INSERT, DELETE O UPDATE hará que se lleven a cabo las acciones del disparador. Las acciones de disparador múltiples de una cláusula IF deben agruparse con BEGIN y END.

Una cláusula IF UPDATE verifica si existe una inserción o actualización para una columna en particular. En el caso de las actualizaciones, la cláusula IF UPDATE da como resultado verdadero, cuando el nombre de la columna se incluye en la cláusula SET de una instrucción UPDATE, aunque la actualización no cambie el valor de la columna. Por otro lado, IF UPDATE no se utiliza con DELETE. Es posible especificar más de una columna y usar más de una cláusula IF UPDATE en una instrucción CREATE OR ALTER TRIGGER. Puesto que el nombre de tabla se especifica en la cláusula ON, no se debe usar el nombre de tabla delante del nombre de columna con IF UPDATE.

Antes de ver un ejemplo es necesario conocer las tablas **inserted** y **deleted.**

Las instrucciones de triggers DML utilizan dos tablas denominadas **inserted** y **deleted.** Microsoft SQL Server 2016 crea y administra automáticamente ambas tablas. La estructura de las tablas **inserted** y **deleted** es la misma que tiene la tabla que ha desencadenado la ejecución del trigger.

La primera tabla (**inserted)** solo está disponible en las operaciones INSERT y UPDATE y en ellas están los valores resultantes después de la inserción o actualización. Es decir, los datos insertados. **Inserted** estará vacía en una operación DELETE.

La segunda tabla (deleted), solo está disponible en las operaciones de UPDATE y DELETE, están los valores anteriores a la ejecución de la actualización o borrado. Es decir, los datos que serán borrados. **Deleted** estará vacío en una operación INSERT.

¿No existe la tabla UPDATED? No, hacer una actualización es lo mismo que borrar (**deleted)** e insertar los nuevos (**inserted).** La sentencia UPDATE es la única en la que **inserted** y **deleted** tienen datos simultáneos.

No se pueden modificar los datos de estas tablas.

A continuación, se muestra un ejemplo sencillo. Este disparador imprime un mensaje cada vez que alguien trata de insertar, eliminar o actualizar datos de la tabla Categories.

```
CREATE OR ALTER TRIGGER utg_Categories ON Categories
FOR INSERT, UPDATE, DELETE
AS
       PRINT 'USTED ACABA DE MODIFICAR LOS VALORES DE LA TABLA Customers'
```

Instrucciones SQL no permitidas en los disparadores

Dado que los disparadores se ejecutan como parte de la transacción, no se permiten las siguientes instrucciones en un disparador.

- Todos los comandos CREATE, incluidos CREATE DATABASE, CREATE TABLE, CREATE INDEX, CREATE PROCEDURE, CREATE DEFAULT, CREATE RULE, CREATE TRIGGER y CREATE VIEW
- Todos los comandos DROP
- ALTER TABLE y ALTER DATABASE
- TRUNCATE TABLE
- GRANT y REVOKE
- UPDATE STATISTICS
- RECONFIGURE
- LOAD DATABASE y LOAD TRANSACTION
- DISK INIT, DISK MIRROR, DISK REFIT, DISK REINIT, DISK REMIRROR, DISK UNMIRROR
- SELECT INTO

Eliminación de disparadores: se puede quitar un disparador omitiéndolo u omitiendo la tabla de disparadores a la que está asociado.

La sintaxis de DROP TRIGGER es la que se muestra a continuación.

```
DROP TRIGGER [PROPIETARIO.]TRIGGER_NAME
            [, [PROPIETARIO.]TRIGGER_NAME]...
```

Cuando se omite una tabla, los disparadores asociados a ella se omiten automáticamente. El permiso drop trigger está asignado de forma predeterminada al propietario de la tabla de disparadores y no es transferible.

2.2 Disparadores de inserción

Este ejemplo muestra un trigger que se dispara para comprobar que el producto que se desea adquirir tenga stock suficiente. Para obtener el stock debe hacer referencia a la tabla Products. Si el producto no tiene stock suficiente, se obtiene un mensaje y no se ejecuta la inserción.

```
CREATE OR ALTER TRIGGER TG_StockProducts ON [Order Details]
FOR INSERT
AS
IF EXISTS (SELECT * FROM Products AS p
            INNER JOIN inserted AS i ON p.ProductID = i.ProductID
            WHERE p.UnitsInStock < i.Quantity)
BEGIN
    PRINT 'There is not enough stock for the selected product'
    ROLLBACK TRANSACTION
    RETURN
END
```

2.3 Disparadores de eliminación

Este ejemplo envía un mensaje de correo electrónico a una persona especificada cuando cambia se elimina un registro de la tabla Customers.

```
CREATE OR ALTER TRIGGER utg_reminder ON Employees
FOR DELETE
```

```
AS
   EXEC msdb.dbo.sp_send_dbmail
        @profile_name = 'Northwind Administrator',
        @recipients = 'danw@northwind.com',
        @body = 'Don''t forget to print a report for the sales force.',
        @subject = 'Reminder'
```

2.4 Disparadores de actualización

Finalmente, este ejemplo muestra un trigger de actualización. Que guarda aquellos cambios realizados en la tabla Categories.

```
CREATE TABLE Categories_Log
(DscModified   varchar (500))

CREATE OR ALTER TRIGGER utg_Categories_Log ON Categories
FOR UPDATE
AS
BEGIN
    DECLARE @DscModified    varchar (500), @CategoryID    Int

    SET @DscModified = 'Se modificó el valor de '
    SELECT @CategoryID = CategoryID FROM inserted

    IF UPDATE(CategoryName)
    BEGIN
        SELECT @DscModified = @DscModified + ' CategoryName de ' +
               d.CategoryName + ' a ' + i.CategoryName
          FROM inserted i, deleted d
    END

    IF UPDATE(Description)
    BEGIN
        SELECT @DscModified = @DscModified + 'Description
               de ' + d.Description + ' a ' + i.Description
          FROM inserted i, deleted d
    END

    INSERT INTO Categories_Log VALUES (@DscModified + ' del CategoryID ' +
                CAST(@CategoryID as varchar))
END
```

Para recordar…

Restricciones de los disparadores

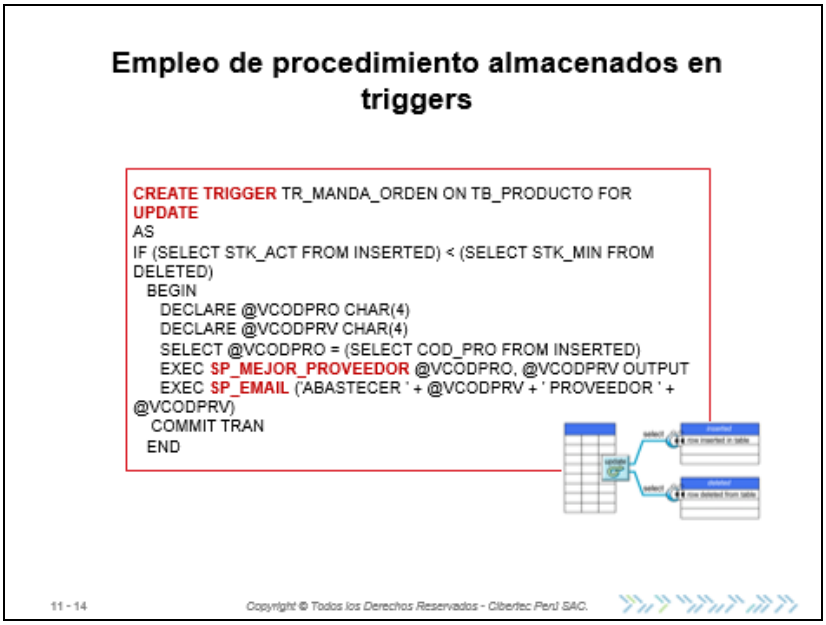
A continuación, se describen algunas limitaciones o restricciones impuestas a los disparadores por SQL Server:

- *Una tabla puede tener un máximo de tres disparadores: uno de actualización, uno de inserción y uno de eliminación.*
- *Cada disparador puede aplicarse a una sola tabla. Sin embargo, un mismo disparador se puede aplicar a las tres acciones del usuario: UPDATE, INSERT y DELETE.*
- *No se puede crear un disparador en una vista ni en una tabla temporal, aunque los disparadores pueden hacer referencia a las vistas o tablas temporales.*
- *Los disparadores no se permiten en las tablas del sistema. Aunque no aparece ningún mensaje de error si se crea un disparador en una tabla del sistema, el disparador no se utilizará.*

Disparadores y rendimiento

En términos de rendimiento, la sobrecarga de disparador es generalmente, muy baja. El tiempo invertido en ejecutar un disparador se emplea, principalmente, para hacer referencia a otras tablas, que pueden estar en memoria o en el dispositivo de base de datos.
Las tablas de verificación de disparadores DELETED e INSERTED siempre están en la memoria activa. La ubicación de otras tablas a las que hace referencia el disparador determina la cantidad de tiempo necesaria para realizar la operación.

3. Empleo de procedimientos almacenados en triggers



En algunos casos es posible que los trigger interactúen con los procedimientos almacenados, de tal forma que realicen de mejor forma su trabajo.

Este ejemplo envía un mensaje de correo electrónico a una persona especificada cuando cambia la tabla Customers.

```
CREATE TRIGGER reminder2
ON Customers
AFTER INSERT, UPDATE, DELETE
AS
   EXEC msdb.dbo.sp_send_dbmail
        @profile_name = 'Northwind Administrator',
        @recipients = 'danw@northwind.com',
        @body = 'Don''t forget to print a report for the sales force.',
        @subject = 'Reminder'
```