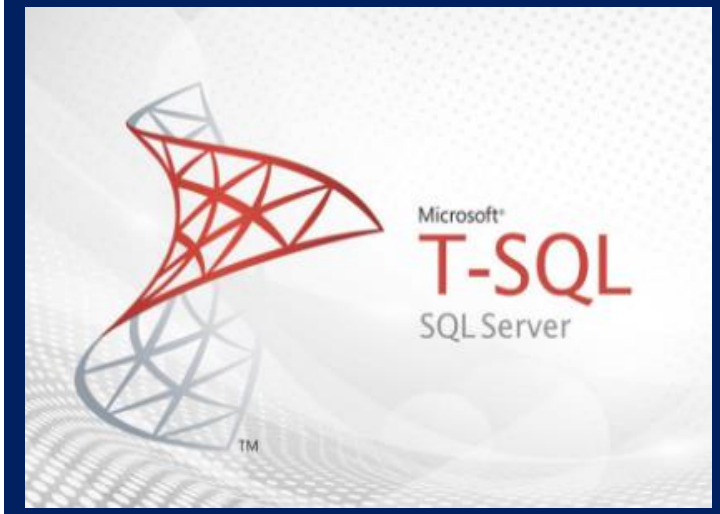
Tema

3B

Bases de Datos Avanzadas

Disparadores (Trigger)



Universidad Francisco de Vitoria
UFV Madrid

Grado en Ingeniería Informática
Escuela Politécnica Superior

Objetivos y resultados de aprendizaje

- El presente tema tiene como objetivo avanzar en el conocimiento de la programación Transact con SQL Server.

- Par ello se aborda la programación de Procedimientos Almacenados.

- Los objetivos específicos consisten en:

  - ✓ Introducción a los distintos tipos de Trigger.

  - ✓ Características de creación y uso

Evaluación del tema

- Los resultados de aprendizaje correspondiente a este tema se evaluarán con los siguientes tipos de pruebas:

  - ✓ Prueba escrita de carácter teórico
  - ✓ Prueba práctica de laboratorio
  - ✓ Participación en clase

Bibliografía

- Para obtener más información puedes consultar:

  - ✓ Libros en pantalla de SQL Server.
  - ✓ Bibliografía incluida en la guía didáctica
  - ✓ https://docs.microsoft.com/es-es/sql/t-sql/statements/create-trigger-transact-sql?view=sql-server-2016

Índice de contenidos

Introducción

- ✓ Como desarrollador o administrador de la BBDD deseamos poder controlar los datos que los usuarios insertan, actualizan o eliminan de las tablas. Posiblemente no sea suficiente con el uso de permisos y relaciones de claves externas. Será necesario el uso de desencadenadores.
- ✓ Un trigger es un tipo especial de procedimiento almacenado que se ejecuta automáticamente al intentarse efectuar una modificación de los datos, en la tabla a la que se encuentran asociados.
- ✓ Los desencadenadores del lenguaje de manipulación de datos (DML) se activan con instrucciones INSERT, UPDATE y/o DELETE.
- ✓ Puede definirse un trigger para cada una de ellas, o bien definir un trigger asociado a una combinación de las mismas.
- ✓ Existen otros eventos de sistemas a nivel de Tabla o Bases de Datos que activaran trigger al nivel correspondiente.
- ✓ No pueden ser invocados manualmente.
- ✓ Los desencadenadores se Crean-Modifican-Eliminan con instrucciones del lenguaje de definición de dato (DDL).

Introducción

- ✓ Se pueden habilitar (estado enabled) o deshabilitar (estado disabled) mediante el comando ALTER TRIGGER…ENABLE|DISABLE
- ✓ Según se desee, pueden crearse triggers para que se disparen una vez por cada registro afectado por el evento o bien una sola vez por cada sentencia, aunque esta afecte a muchos registros ([For each row]).
- ✓ La mayor utilidad que se confiere a un trigger, es la de asegurar la integridad referencial o el cumplimiento de las distintas reglas definidas, si bien estas son operaciones que pueden delegarse en el propio servidor, mediante las instrucciones y cláusulas de especificación de las reglas de integridad, definidas durante la creación de las tablas, o añadidas posteriormente.
- ✓ El hecho de tener algún trigger asociado a una tabla, incide de forma negativa en cuanto al rendimiento se refiere, si bien la mayor parte del tiempo empleado en su ejecución corresponde al acceso a las diferentes tablas implicadas en los chequeos de integridad.
- ✓ Otra de las mayores utilidades de los triggers es deshacer transacciones iniciadas antes de su activación, como consecuencia de un error detectado durante su ejecución
- ✓ En definitiva los Desencadenadores son un método más (junto a las claves externas, normalización,…) para hacer cumplir las reglas de negocio. Son perfectos para para implementar lógicas de negocio sencillas y generar campos calculados

Introducción (Recursividad)

- ✓ Aunque desde el trigger pueden referenciarse objetos de otras bases de datos, éste sólo puede ser creado en la base de datos en curso.

- ✓ El intento de crear un trigger sobre un trigger ya existente, provoca la sustitución inmediata y sin aviso, del anterior.

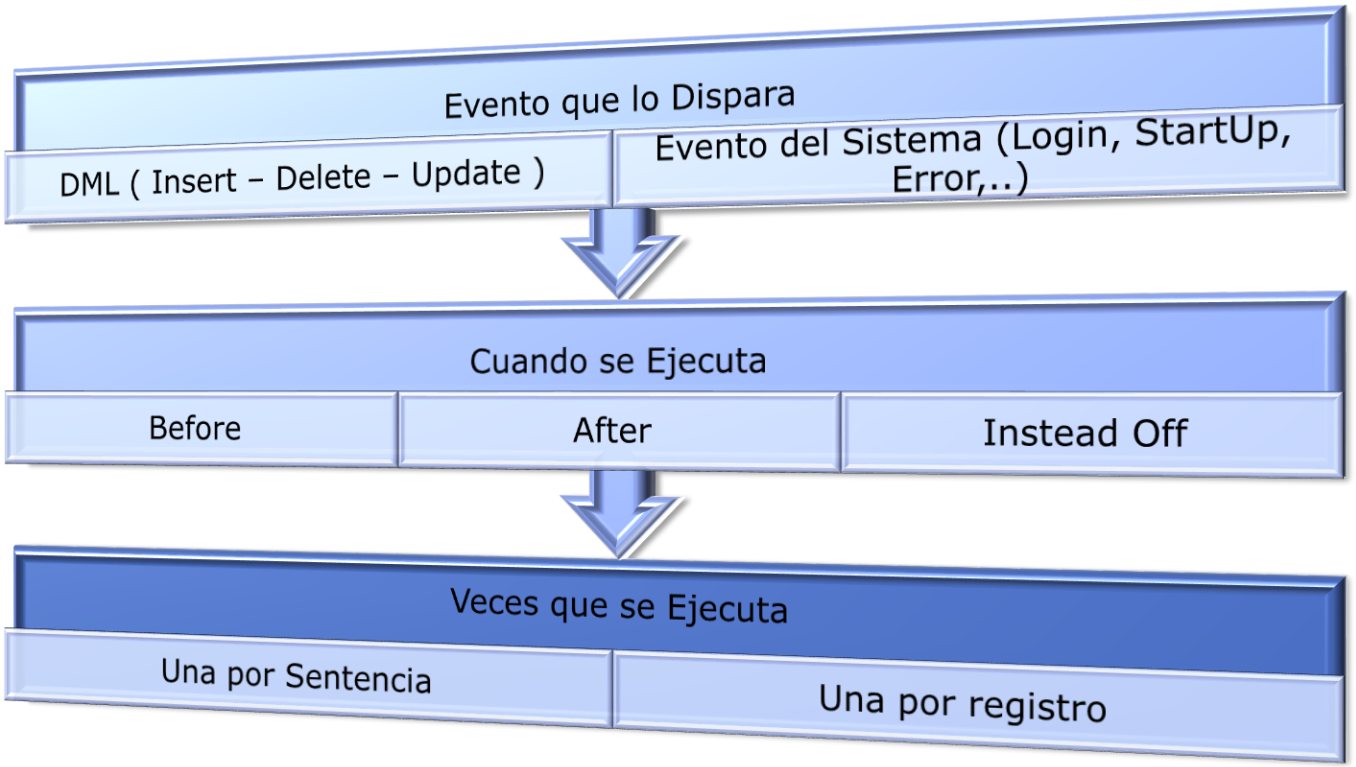
- ✓ Los triggers pueden ser anidados y permiten un nivel máximo, dependiendo del SGBD y su versión, de forma habitual el anidamiento es de 16. En caso de que un trigger caiga en un bucle infinito, se acabará producción un error de desbordamiento del nivel de anidamiento.

- ✓ Los triggers no son reentrantes lo que quiere decir que en caso de que un trigger en ejecución, realice una actualización que provoque la activación del mismo, tal activación no se producirá.

- ✓ Con relación a las transacciones hay que decir que en el caso de comenzarse una transacción y activarse un trigger que contenga y ejecute el comando ROLLBACK TRANSACTION, éste deshará la transacción completa iniciada antes de su activación.

Introducción

- TIPOS
  - ✓ Según su comportamiento los Trigger pueden ser clasificados en función de varios criterios:



Trigger

- INSERT
  - ✓ Como desarrollador o administrador de la BBDD deseamos poder controlar los datos que los usuarios insertan, actualizan o eliminan de las tablas. Posiblemente no sea suficiente con el uso de permisos y relaciones de claves externas. Será necesario el uso de desencadenadores.
  - ✓ Se pueden usar para modificar o incluso desautorizar un registro que se esta insertando. Se activan cada vez que se intenta crear un nuevo registro en la tabla con el comando INSERT.
  - ✓ Los desencadenadores INSERT se activan cada vez que alguien intenta crear un nuevo registro en una tabla mediante la instrucción INSERT.
  - ✓ En cuanto se pretende ejecutar dicha instrucción, SQL Server copia dicho registro en una tabla de la BBDD llamada “tabla de desencadenadores” y en una tabla especial almacenada en memoria llamada «tabla inserted».

Trigger

- DELETE
  - ✓ permite restringir los datos que los usuarios pueden eliminar de una base de datos.
  - ✓ Normalmente, cuando un usuario ejecuta una sentencia DELETE, se quita el registro de la tabla. Este comportamiento cambia cuando se añade un desencadenador DELETE a la tabla, como si nunca hubiera existido.
  - ✓ Cuando se activa el desencadenador se mueve el registro que se va a eliminar a una tabla lógica en al memoria llamada «tabla deleted».
  - ✓ Los registros no desaparecerán por completo y aun es posible hacer referencia a ellos en el código.
  - ✓ Esto es útil una vez mas para hacer cumplir las complejas reglas de negocio o hacer el seguimiento de las acciones llevadas a cabo.

Trigger

- UPDATE
  - ✓ Los desencadenadores UPDATE están diseñados para restringir los datos existentes que los usuarios pueden modificar.
  - ✓ El método que usa el desencadenador UPDATE es una combinación de los métodos que empleados por los desencadenadores INSERT y DELETE.
  - ✓ Esto se debe a que la acción UPDATE se compone en realidad de dos acciones distintas: Una eliminación seguida de una inserción.
  - ✓ Con un desencadenador UPDATE se colocara el registro existente en una la tabla deleted y el nuevo registro en la tabla inserted.
  - ✓ De este modo se podrá comparar las dos tablas para ver si debería completarse la transacción.

Trigger

- SINTAXIS
  - ✓ La instrucción que permite la creación de un trigger es la siguiente:

CREATE TRIGGER Nombre_del_Trigger
ON Nombre_de_la_tabla
{FOR | AFTER | INSTEAD OF} {[INSERT] [,] [UPDATE] [,] [DELETE]}
AS (Sentecias_SQL)

- Nombre_del_Trigger: Debe cumplir las reglas de construcción de identificadores.
- Nombre_de_la_tabla: Nombre de la tabla sobre la que será ejecutado el trigger.
- FOR | AFTER: AFTER especifica que el Trigger sólo se activa cuando todas las operaciones especificadas en la instrucción SQL desencadenadora se han ejecutado correctamente. AFTER es el valor predeterminado. No se pueden definir en las vistas.
- INSTEAD OF: Especifica que se ejecuta el Trigger en vez de la instrucción SQL desencadenadora.
- Sentencias_SQL: Cualquier tipo de sentencia SQL, a excepción de las siguientes:

  - Cualquier instrucción CREATE
  - Cualquier instrucción DROP
  - ALTER TABLE y ALTER DATABASE
  - SELECT INTO
  - GRANT y REVOKE

Trigger

- Ejemplo
  - ✓ Diversos ejemplos de creación de desencadenadores en lenguaje T-SQL (SGBD SQL Server) pueden ser accesible en la propia ayuda de la maquina virtual facilitada.

```
CREATE TRIGGER trgAfterInsert ON [dbo].[Employee_Test]
FOR INSERT
AS
        declare @empid int;
        declare @empname varchar(100);
        declare @empsal decimal(10,2);
        declare @audit_action varchar(100);

        select @empid=i.Emp_ID from inserted i;
        select @empname=i.Emp_Name from inserted i;
        select @empsal=i.Emp_Sal from inserted i;
        set @audit_action='Inserted Record -- After Insert Trigger.';

        insert into Employee_Test_Audit
           (Emp_ID,Emp_Name,Emp_Sal,Audit_Action,Audit_Timestamp)
        values(@empid,@empname,@empsal,@audit_action,getdate());

        PRINT 'AFTER INSERT trigger fired.'
GO
```