**TRIGGERS (Disparadores)**

**Introducción**

Un TRIGGER es un tipo especial de STORE PROCEDURE que se dispara cuando sucede un evento en particular.

TRIGGER traducido como desencadenador, es una clase especial de procedimiento almacenado que se ejecuta automáticamente cuando se produce un evento en el servidor de bases de datos.

Los usos más comunes para un TRIGGER son:

* Mantener la integridad de los datos.
* Mantener la integridad referencial
* Mantener auditoría de los cambios en los registros de las tablas
* Invocar acciones externas, tales como enviar un mail…

Se pueden disparar los TRIGGERS cuando se modifican datos con alguna de las sentencias: INSERT, DELETE o UPDATE

Existen dos tipos de TRIGGERS:

* AFTER TRIGGER
* INSTEAD-OF TRIGGER

**AFTER TRIGGER**

Es el TRIGGER por defecto en SQL Server.

Se pueden definir múltiples TRIGGERS para una tabla y para un mismo evento.

Cada TRIGGER puede invocar varios STORES PROCEDURES.

Se puede crear un solo TRIGGER que se ejecute para alguno/s o para todos los eventos de modificación de una tabla: INSERT, UPDATE o DELETE.

SQL Server no ofrece la posibilidad de disparar un TRIGGER desde la sentencia SELECT.

Estos TRIGGERS solo puede existir para tablas y no para vistas.

SQL Server permite la ejecución de TRIGGER recursivos, siempre y cuando la opción recursive triggers este seteada en TRUE. Esto significa que si el TRIGGER modifica la tabla dentro del mismo TRIGGER esto dispararía una segunda ejecución del TRIGGER. Este procedimiento no puede entrar en un LOOP infinito porque SQL Server permite un máximo de 32 niveles de profundidad.

**INSTEAD-OF  TRIGGERS**

INSTEAD OF TRIGGERS son ejecutados en forma automática antes de que SQL SERVER realice los chequeos de PRIMARY KEY y FOREIGN KEY. Mientras que los TRIGGERS tradicionales son ejecutados automáticamente después de chequear estas CONSTRAINTS.

Solo se puede tener un instead-of trigger por cada acción (INSERT, UPDATE, y DELETE).

No se puede combinar instead-of triggers y claves foráneas que tengan definiciones de CASCADA

Instead-of triggers están previstos para permitir actualizaciones sobre vistas, que normalmente no son modificables.

**Creando TRIGGERS**

Son procedimientos almacenados que se ejecutan como respuesta de la modificación de una tabla.

Una de sus principales utilidades es de mantener la integridad de los datos.

La sentencia encargada de su creación es CREATE TRIGGER, con una sintaxis similar a los otros objetos.

CREATE TRIGGER [propietario.] nombretrigger

ON [propietario.] nombretabla

[WITH ENCRYPTION]

{ FOR | AFTER | INSTEAD OF } {INSERT, UPDATE, DELETE}

AS sentenciasSQL

**Restricciones**

No se permite recursividad de ejecución en un trigger.

Cuando se produce un TRUNCATE TABLE no se ejecutan triggers.

No permite la ejecución de las siguientes sentencias:

CREATE VIEW

CREATE TRIGGER

CREATE DEFAULT

CREATE PROCEDURE

CREATE RULE

CREATE DATABASE

CREATE INDEX

DROP y de modificación de estructuras como:

ALTER DATABASE

ALTER TABLE

GRANT

REVOKE

**INSERT TRIGGER, DELETE TRIGGER, UPDATE TRIGGER**

Cuando se producen cambios sobre las tablas ya sean inserción, borrado o actualización, SQL Server ejecuta el trigger y además dispone de dos tablas temporales inserted y deleted con los datos antes de la actualización o borrado.

Estas tablas se las puede manipular con SELECT.

Las tablas inserted y deleted no pueden ser alteradas ni modificadas. Solo pueden ser consultadas.

La cláusula ROLLBACK

**Prácticas con TRIGGERS**

El siguiente ejemplo crea una tabla de prueba y se le asigna un trigger para imprimir en pantalla cada vez que inserten un registro en la tabla prueba. El objetivo de este ejemplo es experimentar como es el funcionamiento del trigger en el evento INSERT.

/\* CREO LA TABLA \*/

CREATE TABLE PRUEBA (CODIGO INT, NOMBRE VARCHAR (50))

/\* CREO EL TRIGGER \*/

CREATE TRIGGER TI\_PRUEBA

        ON PRUEBA FOR INSERT

AS

BEGIN

        PRINT 'REGISTRO NUEVO'

END

/\* INSERTO UN REGISTRO EN LA TABLA PRUEBA Y  ESO DISPARARA EL TRIGGER \*/

INSERT INTO PRUEBA

SELECT 1,'GABRIEL'

INSERT INTO PRUEBA

SELECT 2,'CARLOS'

INSERT INTO PRUEBA

SELECT 3,'JUAN'

En este ejemplo desarrollamos la misma práctica pero para el evento UPDATE.

/\* CREO EL TRIGGER \*/

CREATE TRIGGER TU\_PRUEBA

        ON PRUEBA FOR UPDATE

AS

BEGIN

        PRINT 'REGISTRO MODIFICADO'

END

/\* MODIFICO UN REGISTRO EN LA TABLA PRUEBA Y

 ESO DISPARARA EL TRIGGER \*/

UPDATE PRUEBA

SET NOMBRE='OMAR'

WHERE CODIGO=1

En este ejemplo desarrollamos la misma práctica, pero para el evento DELETE.

CREATE TRIGGER TD\_PRUEBA

        ON PRUEBA FOR DELETE

AS

BEGIN

        PRINT 'REGISTRO BORRADO'

END

/\* BORRO UN REGISTRO EN LA TABLA PRUEBA Y

 ESO DISPARARA EL TRIGGER \*/

DELETE FROM PRUEBA

WHERE NOMBRE LIKE '%OMAR%'

Ejemplo de trigger combinando varios eventos INSERT; UPDATE y DELETE.

/\* CREO EL TRIGGER \*/

CREATE TRIGGER TV\_PRUEBA

        ON PRUEBA FOR INSERT, UPDATE,DELETE

AS

BEGIN

        PRINT 'SE REALIZO ALGUNA OPERACION EN EL REGISTRO'

END

/\* BORRO UN REGISTRO EN LA TABLA PRUEBA  Y

 ESO DISPARARA EL TRIGGER \*/

DELETE FROM PRUEBA

WHERE NOMBRE LIKE '%JUAN%'

En este ejemplo desarrollamos un trigger para el evento INSERT donde mostramos el usuario que intenta insertar, la fecha y hora del momento de insert y los datos de la tabla.

------------------------------------------------------------------------------------

/\* CREO EL TRIGGER \*/

CREATE TRIGGER TIN\_PRUEBA

        ON PRUEBA FOR INSERT

AS

BEGIN

        SELECT HOST\_NAME() USUARIO,SUSER\_NAME(),GETDATE() FECHA, 'DATOS INGRESADOS',CODIGO,NOMBRE

        FROM INSERTED /\* DELETED ES CUANDO ELIMINA \*/

END

/\* INSERTO UN REGISTRO EN LA TABLA PRUEBA Y

 ESO DISPARARA EL TRIGGER \*/

INSERT INTO PRUEBA

SELECT 4,'GABRIEL'

Este es el mismo ejemplo que el anterior pero para el evento UPDATE

------------------------------------------------------------------------------------------

/\* CREO EL TRIGGER \*/

ALTER TRIGGER TIN\_PRUEBA

        ON PRUEBA FOR UPDATE

AS

BEGIN

        SELECT HOST\_NAME() USUARIO,SUSER\_NAME(),GETDATE() FECHA, 'DATOS INGRESADOS',CODIGO,NOMBRE

        FROM INSERTED /\* DELETED ES CUANDO ELIMINA \*/

        UNION

        SELECT HOST\_NAME() USUARIO,SUSER\_NAME(),GETDATE() FECHA, 'DATOS INGRESADOS',CODIGO,NOMBRE

        FROM DELETED

END

DROP TRIGGER TV\_PRUEBA

/\* INSERTO UN REGISTRO EN LA TABLA PRUEBA Y

 ESO DISPARARA EL TRIGGER \*/

UPDATE PRUEBA

SET NOMBRE='OMAR'

WHERE CODIGO=1