DISPARADORES
(Evento – condición - acción)

Un trigger (disparador) es una clase especial de procedimiento almacenado que se ejecuta automáticamente cuando se produce un evento en el servidor de bases de datos, como efecto secundario de la modificación de la base de datos. El disparador debe cumplir con dos cosas:

- Especificar las condiciones en las que se va a ejecutar el disparador; Esto consiste en un evento para el cual se ejecutara el disparador, siempre y cuando cumpla una condición.

- Especificar las acciones que se van a realizar cuando se ejecute el disparador.

Podemos concluir entonces, que un disparador es un procedimiento que se ejecuta cuando se cumple una condición establecida al realizar una modificación de datos (DML), o comandos DDL.

Los disparadores definen las acciones que se deben ejecutar automáticamente cuando se producen ciertos eventos. Tienen muchos usos: la implementación de reglas de negocios, registro histórico de la auditoria e incluso para realizar acciones fuera del sistema de bases de datos.

Un Trigger devuelve resultados al programa que lo desencadena de la misma forma que un Stored Procedure aunque no es lo mas idóneo; Para impedir que una instrucción de asignación devuelva un resultado se puede utilizar la sentencia SET NOCOUNT al principio del Trigger.

TRIGGERS EN TRANSACT SQL

SQL Server proporciona los siguientes tipos de triggers:

- Trigger DML, se ejecutan cuando un usuario intenta modificar datos mediante un evento de lenguaje de manipulación de datos (DML). Los eventos DML son instrucciones INSERT, UPDATE o DELETE de una tabla o vista.

- Trigger DDL, se ejecutan en respuesta a una variedad de eventos de lenguaje de definición de datos (DDL). Estos eventos corresponden principalmente a instrucciones CREATE, ALTER y DROP de Transact-SQL, y a determinados procedimientos almacenados del sistema que ejecutan operaciones de tipo DDL.

- Trigger LOGIN

TRIGGER DML

La sintaxis general de un trigger es la siguiente.

```
CREATE TRIGGER <Trigger_Name, sysname, Trigger_Name>

ON <Table_Name, sysname, Table_Name>

INSTEAD OF / AFTER <INSERT,DELETE,UPDATE>

AS

BEGIN

-- SET NOCOUNT ON impide que se generen mensajes de texto

SET NOCOUNT ON;

-- código para definer la acción que realizara el trigger

END
```

Antes de ver un ejemplo es necesario conocer las tablas inserted y deleted. Las instrucciones de triggers DML utilizan dos tablas especiales denominadas inserted y deleted. SQL Server 2008 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 ella están los valores resultantes después de la inserción o actualización. Es decir, los datos insertados. Inserted estará vacía en una operación DELETE.

En la segunda (deleted), disponible en las operaciones UPDATE y DELETE, están los valores anteriores a la ejecución de la actualización o borrado. Es decir, los datos que serán borrados. Deleted estará vacía en una operacion INSERT.

¿No existe una 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 simultáneamente. No se pueden modificar directamente los datos de estas tablas.

El siguiente ejemplo, graba un histórico de saldos cada vez que se modifica un saldo de la tabla cuentas.

```
CREATE TRIGGER TR_CUENTAS
ON CUENTAS
AFTER UPDATE
AS
BEGIN
-- SET NOCOUNT ON impide que se generen mensajes de texto
-- con cada instrucción
     SET NOCOUNT ON;
     INSERT INTO HCO_SALDOS
     (IDCUENTA, SALDO, FXSALDO)
     SELECT IDCUENTA, SALDO, getdate()
     FROM INSERTED
END
```

La siguiente instrucción provocará que el trigger se ejecute:

```
UPDATE CUENTAS
SET SALDO = SALDO + 10
WHERE IDCUENTA = 1
```

Una consideración a tener en cuenta es que el trigger se ejecutará aunque la instrucción DML (UPDATE, INSERT o DELETE) no haya afectado a ninguna fila. En este caso inserted y deleted devolverán un conjunto de datos vacío.

Podemos especificar que las acciones se realicen solo si se modifico determinadas columnas de la tabla, de la siguiente forma:

```
ALTER TRIGGER TR_CUENTAS

ON CUENTAS

AFTER UPDATE

AS

BEGIN

     SET NOCOUNT ON;

     IF UPDATE(SALDO) -- Solo si se actualiza SALDO

     BEGIN

          INSERT INTO HCO_SALDOS

          (IDCUENTA, SALDO, FXSALDO)

          SELECT IDCUENTA, SALDO, getdate()

          FROM INSERTED

    END

END
```

Los trigger están dentro de la transacción original (Insert, Delete o Update) por lo cual si dentro de nuestro trigger hacemos un RollBack Tran, no solo estaremos echando atrás nuestro trigger sino también toda la transacción; en otras palabras si en un trigger ponemos un RollBack Tran, la transacción de Insert, Delete o Update volverá toda hacia atrás.

```
ALTER TRIGGER TR_CUENTAS

ON CUENTAS

AFTER UPDATE

AS

BEGIN

     SET NOCOUNT ON;

     INSERT INTO HCO_SALDOS

     (IDCUENTA, SALDO, FXSALDO)

     SELECT IDCUENTA, SALDO, getdate()

     FROM INSERTED

      ROLLBACK

END
```

En este caso obtendremos el siguiente mensaje de error:

La transacción terminó en el desencadenador. Se anuló el lote.

Podemos activar y desactivar Triggers a tarvés de las siguientes instrucciones.

```
 -- Desactiva el trigger TR_CUENTAS
DISABLE TRIGGER TR_CUENTAS ON CUENTAS
GO
-- activa el trigger TR_CUENTAS
ENABLE TRIGGER TR_CUENTAS ON CUENTAS
GO
-- Desactiva todos los trigger de la tabla CUENTAS
ALTER TABLE CUENTAS DISABLE TRIGGER ALL
GO
-- Activa todos los trigger de la tabla CUENTAS
ALTER TABLE CUENTAS ENABLE TRIGGER ALL
```

TRIGGER DDL

La sintaxis general de un trigger es la siguiente.

```
CREATE TRIGGER <trigger_name, sysname, table_alter_drop_safety>
ON DATABASE
FOR <data_definition_statements, , DROP_TABLE, ALTER_TABLE>
AS
BEGIN
END
```

La siguiente instrucción impide que se ejecuten sentencias DROP TABLE y ALTER TABLE en la base de datos.

```
CREATE TRIGGER TR_SEGURIDAD

ON DATABASE FOR DROP_TABLE, ALTER_TABLE

AS

BEGIN

      RAISERROR ('No está permitido borrar ni modificar tablas !' , 16, 1)

      ROLLBACK TRANSACTION

END
```

EJEMPLO DE APLICACIÓN

Como ejemplo crearemos un Trigger que avise al webmaster con un mail cuando un usuario se da de alta en nuestro web. El funcionamiento del trigger es muy sencillo, declaramos dos variables, una para el mensaje que se enviará en el mail y otra para obtener el ID del registro recién insertado y luego este ID lo concatenamos al mensaje para enviárselo al webmaster.

```
Alter Trigger Trigger_Aviso_al_Webmaster
On dbo.pr_usuarios
After Insert
As
BEGIN
        -- Declaramos las variables del mensaje y del ID del nuevo usuario
        Declare @Mensaje varchar(200)
        Declare @ID numeric

        -- Obtenemos el id del usuario recien insertado
        Select @ID = (Select IDUsuario From Inserted )
        Select @Mensaje = 'Nuevo Usuarios en el web : ' + Convert(varchar(10), @ID)

        Exec master.dbo.xp_sendmail
        @recipients = 'webmaster@dominio.com',
        @subject = 'Nuevo usuario',
        @message = @Mensaje
END
```