UNIVERSIDAD TÉCNOLOGICA NACIONAL
FACULTAD REGIONAL GENERAL PACHECO

TECNICATURA UNIVERSITARIA EN PROGRAMACIÓN

Base de Datos II

Teoría de Triggers

ING. ARIEL HERRERA

Contenido

Triggers en Transact SQL

Un trigger (o desencadenador) es una clase especial de procedimiento almacenado que se ejecuta automáticamente cuando se produce un evento en el servidor de bases de datos.

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 DML.

Los 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.

La sintaxis general de un trigger es la siguiente.

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

ON <Table_Name, sysname, Table_Name> AFTER

<Data_Modification_Statements,
, INSERT,DELETE,UPDATE>

AS BEGIN

-- SET NOCOUNT ON added to prevent extra result sets frominterfering  with
SELECT
-- statements.

SET NOCOUNT ON;

-- Insert statements for trigger here

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 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 despues de la inserción o actualización. Es decir, los datos insertados. **Inserted** estará vacia 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 vacia 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 simultaneamente.

No puede se modificar directamente los datos de estas tablas.

El siguiente ejemplo, graba un historico 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 instruccion DML (UPDATE, INSERT o DELETE) no haya afectado a ninguna fila. En este caso **inserted** y **deleted** devolverán un conjunto de datos vacio.

Podemos especificar a que columnas de la tabla debe afectar el trigger.

```
ALTER 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;
        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 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
        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 través 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 latabla CUENTAS

ALTER TABLE CUENTAS
DISABLE TRIGGER ALL
GO

-- Activa todos los trigger de la tabla CUENTAS

ALTER TABLE CUENTAS
ENABLE TRIGGER ALL
```

Trigger DDL

Los 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. 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
```