**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 from  -- interfering 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 2005 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 operación 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 puede 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 a qué 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 la tabla CUENTAS  **ALTER** **TABLE** CUENTAS **DISABLE** **TRIGGER** ALL  GO  -- Activa todos los trigger de la tabla CUENTAS  **ALTER** **TABLE** CUENTAS **ENABLE** **TRIGGER** ALL |

Usuario: System\_user