TEMA 7: TRIGGERS CURSORES Y EXCEPCIONES

BERNAT COSTA

BERNAT.COSTA@CESURFORMACION.COM

¿QUÉ PODEMOS HACER CON T-SQL?

Guiones (Scripts)

Métodos o funciones

Triggers

TRIGGERS

DISPARADORES AUTOMÁTICOS

¿QUÉ ES UN TRIGGER?

Los triggers o disparadores, son procedimientos que se ejecutan de forma automática cuando pasa algo en la bbdd.

Podremos decidir cuando se ejecutan y que se ejecuta.

Estas operaciones pueden ser
de actualización (UPDATE), inserción (INSERT) , borrado (DELETE)...

UN TRIGGER ES UN PROCEDIMIENTO...

POR LO TANTO, PODRÁ MODIFICAR NUESTRA BBDD.

USOS DE LOS TRIGGERS

Nos permite registrar, auditar y monitorear los procesos de cambio de valores a las tablas de la base de datos activas.

Puede validar los valores aprobando o negando acciones realizadas por las sentencias SQL.

Puede preservar la consistencia y claridad de los valores, ejec utando acciones relacionadas con los campos de la bbdd.

(Recalcular el total de un pedido al añadir una linea, por ejemplo).
(Validar un DNI)
...

VENTAJAS

- Un trigger ofrece chequeos de seguridad en valores de las tablas de una base de datos.
- Fuerzan restricciones dinámicas de integridad de datos y de integridad referencial
- Aseguran que las operaciones relacionadas se realizan juntas de forma implícita

INCONVENIENTES

- Se tiene que programar anticipadamente lo que tiene que realizar un trigger
- Aunque es un procedimiento no se puede invocar directamente
- Los triggers siempre serán creados para un conjunto de registros y no para uno solo ya que se dispara por operación SQL

2 TIPOS DE TRIGGERS:

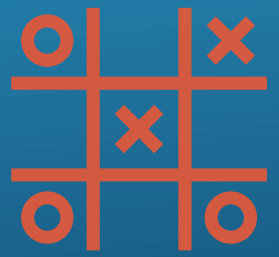
Sobre una tabla

Sobre una BBDD o Servidor

DIFERENCIAS

Trigger sobre una BBDD

Se disparará cuando pase algo en la bbdd. Se cree un usuario, se loguee algún usuario, se borre una tabla...



Trigger sobre una tabla

Se disparará cuando pase algo en una tabla (un insert, update o delete)

TRIGGER SOBRE UNA TABLA

```
GO
CREATE TRIGGER NOMBRE_TRIGGER
ON [TABLE | VIEW ]
FOR | INSTEAD OF
[ INSERT ][ , ][ UPDATE ][ ,] [ DELETE ]
AS
BEGIN
   SENTENCIA_SQL
END
GO
```

TRIGGER SOBRE UNA BBDD

```
GO
CREATE TRIGGER NOMBRE_TRIGGER
ON [ALL SERVER | DATABASE]
FOR
AS
BEGIN
  SENTENCIA_SQL
END
GO
```

AL CREAR UN TRIGGER

Sobre TABLA

- Indicar si se dispara con un insert, update o delete
- Indicar la tabla que lo dispara

Sobre BBDD/SERVER

- Indicar si es sobre BBDD o Server
- Indicar el disparador (mirar lista en internet)

https://docs.microsoft.com/es-es/sql/relational-databases/triggers/ddl-events?view=sql-server-ver15

VAMOS A CREAR UN TRIGGERS

Queremos guardar en una tabla Menu_HTO todo cambio de precio en la tabla menu con la fecha en que se ha cambiado.

1° CREAMOS UNA TABLA...

```
Use arepazo


CREATE TABLE Menu_HCO (
  Id int IDENTITY(1,1) PRIMARY KEY,
  IdMenu int NOT NULL,
  Nombre Varchar(100) NOT NULL,
  PrecioVenta varchar(100) NOT NULL,
  Fecha DateTime NOT NULL
)
```

2°

CREAMOS EL TRIGGER

```
GO
CREATE TRIGGER HistoricoPrecio
ON MENU
FOR UPDATE,INSERT
AS
BEGIN
  SET NOCOUNT ON;
  IF UPDATE(PrecioVenta)--
   Solo si se actualiza pvp
    BEGIN
        INSERT INTO MENU_HCO
        SELECT id,nombre,precioVenta, getdate()
        FROM INSERTED
    END
END
GO
```

COMPROBEMOS SI FUNCIONA...

- Cambia el precio de la arepa albina a 4.00€
- Cambia el nombre de los tequeños a Pequeños
- Mira la tabla Menu_hco

2 COSAS A TENER EN CUENTA

Función UPDATE()

Tablas Inserted y Deleted

FUNCIÓN UPDATE

- Nos devuelve true o false indicándonos si el evento que ha disparado el trigger ha sido un cambio en el valor del campo indicado.
- El campo tiene que ser de la tabla que ha disparado el trigger.

TABLAS INSERTED Y DELETED

- Nos servirán en nuestros triggers para recuperar los valores que se han modificado y han disparado nuestro trigger.
- En un trigger de update, insert, tendremos la tabla inserted con los valores que se han insertados.
- En un trigger de delete,update, la tabla deleted con los valores borrados.

CREAMOS UN TRIGGER SOBRE LA BBDD

Vamos a crear un trigger
que nos impida hacer un drop table.

EJEMPLO TRIGGER

```
CREATE TRIGGER SeguridadBorrarTabla

ON DATABASE

FOR DROP_TABLE

AS

PRINT
    'Para borrar esta tabla debes deshabilitar el Trigger Seg
    uridadBorrarTabla'

ROLLBACK
```

INTENTA BORRAR UNA TABLA...

Drop table menu_hto

PODEMOS DESHABILITAR, HABILITAR O BORRAR UN TRIGGER

```
DISABLE TRIGGER HistoricoPrecio
    ON Menu
```

o...

```
ENABLE TRIGGER HistoricoPrecio
    ON Menu
```

```
DISABLE trigger SeguridadBorrarTabla on DATABASE
```

EJERCICIOS

1. Crear un trigger para que guarde en una tabla historico, todo cambio o inserción en la tabla clientes, guardando la fecha actual en el registro.

2. Quiero que creeis un trigger para que, cuando se añada un elemento al menú de la categoria Arepas (id cat = 1, se le añada el ingrediente masa arepa (id ingrediente=1) a la receta automaticamente.

CURSORES EN SQL SERVER

¿PARA QUE SIRVE UN CURSOR?

- Servirán para recorrer fila a fila una tabla desde un procedimiento o función

- El uso de cursores implicará 5 fases.
  - Declaración, Apertura, Acceso a datos, Cierre y Desalojo.

LAS 5 FASES

Declaración	Apertura	Acceso a datos	Cierre	Desalojo.
• Deberemos declarar nuestro cursor igual que declaramos una variable.	• El cursor se tiene que abrir para poder ser usado.	• Accederemos a los datos de la fila desde un bucle en TSQL	• Debe cerrarse el cursor antes de liberar la memoria.	• Liberamos la memoria del cursor y lo destruimos.

DECLARACIÓN

```
DECLARE CursorCliente CURSOR FOR SELECT email FROM
Clientes
```

APERTURA

- En este momento, es donde se ejecuta la consulta y se guarda en nuestro cursor.
- El cursor se puede interpretar como un Array con los resultados de la consulta.

```
OPEN CursorCliente
```

ACCESO A DATOS

- Metemos el primer registro de nuestro array en una variable y movemos el cursor al siguiente registro.
- La variable @emailcliente hay que declararla antes.
  - Declare @emailcliente varchar(100)…

```
FETCH NEXT FROM CursorCliente INTO @emailcliente
```

¿Y CUANDO DEJO DE LEER?

- Como se cuando he llegado al final?
- Gracias a la variable @@FETCH_STATUS que nos devolverá 0 cuando no haya mas filas.

```
…
WHILE @@fetch_status = 0
  BEGIN
   …
  END
```

CIERRE Y DESALOJO

- El cursor hay que cerrarlo fuera del bucle.

```
CLOSE CursorCliente
```

También debemos liberar la memoria del cursor con la instrucción:

```
DEALLOCATE CursorCliente
```

A VER TODO JUNTO...

```
DECLARE @emailCliente AS nvarchar(400)
DECLARE CursorEmail CURSOR FOR SELECT email FROM Clientes
OPEN CursorEmail
FETCH NEXT FROM CursorEmail INTO @emailCliente
WHILE @@fetch_status = 0
  BEGIN
    PRINT 'El email es: ' + @emailCliente
    FETCH NEXT FROM CursorEmail INTO @emailCliente
  END
CLOSE CursorEmail
DEALLOCATE CursorEmail
```

EJERCICIOS

1. Crea una función que devuelva una cadena con todos los emails de la tabla clientes separados por ,

2. Vamos a complicar el ejercicio anterior. Quiero que devuelva una tupla por cada email encontrado en la tabla clientes. Una tupla formada por el email y el numero de pedidos que ha realizado el cliente separados por |. Tiene que devolver algo asi:

   Pepe@gmail.com|23, angeles@yahoo.es|5, …


That's all Folks!