DuocUC

SQL SERVER APLICADO (SSA010)

Ariel Alexis Fierro Sáez
afierrosaez@gmail.com

Disparadores (Trigger)

- Un triggers es una clase especial de procedimiento almacenado que se dispara automáticamente su ejecución cuando se produce un evento en el servidor de base de datos.

- Los trigger en transactSQL pueden ser de tres tipos:

  - **Trigger DML** se ejecutan cuando un usuario intenta modificar datos mediante un evento de lenguaje de manipulación de datos (INSERT, UPDATE o DELETE).

  - **Trigger DDL** se ejecutan en respuesta a una variedad de eventos de lenguaje de definición de datos ,los cuales corresponden principalmente a instrucciones CREATE, ALTER y DROP de Transact-SQL.

  - **Trigger LOGON** se activan en respuesta al evento LOGON que se genera cuando se establece la sesión de un usuario con una instancia SQL Server.

Disparadores (Trigger)

- Los Trigger DML se utilizan frecuentemente para imponer las reglas de negocios y la integridad de los datos.
- Los Trigger DDL se utilizan frecuentemente para tareas como de auditoria y regulación de las operaciones sobre la base de datos.
- Los Trigger LOGON se pueden utilizar para controlar sesiones de servidor, restricciones de inicio de sesiones o bien restricciones al numero de sesiones.
- Los Trigger y las secuencias que desencadenan al momento de su ejecución trabajando como una transacción.

Disparadores (Trigger)

- Creación de un Trigger DML

```
CREATE TRIGGER <nombre_trigger>
ON <nombre_tabla >
 FOR | AFTER | INSTEAD OF
< INSERT > , < UPDATE > , < DELETE >
AS
BEGIN

        sql_statement
        ........
END
```

Disparadores (Trigger)

- Creación de un Trigger DDL

```
CREATE TRIGGER <nombre_trigger>
ON <ALL SERVER> | <DATABASE>>
FOR | AFTER
< CREATE > ,  < ALTER > , < DROP >
AS
BEGIN

        sql_statement
        ........

END
```

Eventos en trigger DDL: http://msdn.microsoft.com/en-us/library/ms189871%28v=sql.90%29.aspx

Disparadores (Trigger)

- Creación de un Trigger LOGON

```
CREATE TRIGGER <nombre_trigger>
ON <ALL SERVER>
FOR | AFTER  < LOGON >
AS
BEGIN

        sql_statement
        ........

END
```

Disparadores (Trigger)

- Argumentos
  - **AFTER**, especifica que el desencadenador solo se activa cuando todas las operaciones especificadas en la instrucción SQL desencadenadora se han ejecutado correctamente. Además, todas las acciones referenciales en cascada y las comprobaciones de restricciones deben ser correctas para que este desencadenador se ejecute.
  - **FOR**, la especificación de AFTER produce el mismo efecto que especificar FOR, que es la única opción disponible en las versiones anteriores de Microsoft SQL Server
  - **INSTEAD OF**, especifica que se ejecuta el trigger DML *en vez de* la instrucción SQL desencadenadora, por lo que se suplantan las acciones de las instrucciones desencadenadoras. INSTEAD OF no se puede especificar para los trigger DDL o LOGON.
  - **ALL SERVER**, aplica el ámbito de un desencadenador DDL o logon al servidor actual. Si se especifica, el desencadenador se activa cada vez que *event_type* tienen lugar en la base de datos actual.
  - **DATABASE**, aplica el ámbito de un desencadenador DDL a la base de datos actual. Si se especifica, el desencadenador se activa cada vez que *event_type* tienen lugar en la base de datos actual.

Disparadores (Trigger)

- **TRIGGER DML**
  - Los Trigger DML utilizan dos tablas especiales: la tabla **delete** y la tabla **inserted**, SQL Server es quien crea y administra automáticamente ambas tablas cache.
  - Estas tablas temporales residentes en memoria para probar los efectos de determinadas modificaciones de datos y para establecer condiciones para las acciones del trigger DML.
  - La tabla **deleted** almacena copias de las filas afectadas por las instrucciones DELETE y UPDATE. Durante la ejecución de una instrucción DELETE o UPDATE, las filas se eliminan de la tabla y se transfieren a la tabla **deleted.**
  - La tabla **inserted** almacena copias de las filas afectadas durante las instrucciones INSERT y UPDATE. Durante una transacción de inserción o actualización, se agregan nuevas filas a la tabla **inserted** y a la tabla del desencadenador
  - Una transacción de actualización es similar a una operación de eliminación seguida de una operación de inserción; primero, se copian las filas antiguas en la tabla **deleted** y luego se copian las filas nuevas en la tabla del desencadenador y en la tabla **inserted**.

Disparadores (Trigger)

- **TRIGGER DDL**
  - Cuando se activa un trigger DDL, la información referente al evento que dispara dicho trigger se encuentra en una función denominada EVENTDATA().
  - EVENTDATA(), retorna un valor XML en el cual incluye información del evento, como por ejemplo:
    - Hora del evento
    - El Id. de proceso del sistema (SPID) de la conexión en la cual se ha ejecutado el desencadenador.
    - El tipo de evento que ha activado el desencadenador.

Disparadores (Trigger)

- **XML retornado por EVENTDATE()**

```
<EVENT_INSTANCE>
<EventType>type</EventType>
<PostTime>datetime</PostTime>
<SPID>spid</SPID>
<ServerName>name</ServerName>
<LoginName>name</LoginName>
<UserName>name</UserName>
<DatabaseName>name</DatabaseName>
<SchemaName>name</SchemaName>
<ObjectName>name</ObjectName>
<ObjectType>type</ObjectType>
<TSQLCommand>command</TSQLCommand>
</EVENT_INSTANCE>
```

Disparadores (Trigger)

- **Resumiendo**

TRIGER DML

- La tabla **deleted** contiene todos los registros que se intentan borrar o modificar, es decir realiza un respaldo de los registros antes que sean afectados por las sentencias del tipo DELETE o UPDATE
- La tabla **inserted** contiene todos los registros que serán insertando o modificados, es decir contiene los nuevos registros que son afectados por las sentencias del tipo INSERT o UPDATE
- Utilice ambas tablas para según sea el caso para la gestión del trigger.
- La tabla **inserted** estará vacía en una operación DELETE.
- La tabla **deleted** estará vacía en una operación INSERT.

TRIGER DDL

- EVENTDATA(), retorna información del evento que produjo la ejecución del trigger DDL.

Disparadores (Trigger)

- Activación/Desactivación de Trigger de una tabla.

Desactiva un trigger

DISABLE TRIGGER <nombre_trigger> **ON** <nombre_tabla>

Activa un trigger

ENABLE TRIGGER <nombre_trigger>**ON** <nombre_tabla>

Desactiva todos los trigger

ALTER TABLE <nombre_tabla>**DISABLE TRIGGER** ALL

Activa todos los trigger

ALTER TABLE <nombre_tabla> **ENABLE TRIGGER** ALL

Disparadores (Trigger)

- Borrar Trigger de una tabla.

Borrar Trigger DML

DROP TRIGGER <nombre_trigger>

Borrar Trigger DDL

DROP TRIGGER <nombre_trigger> **ON** {DATABASE | ALL SERVER}

Borrar Trigger LOGON

DROP TRIGGER <nombre_tabla> **ON** ALL SERVER

Disparadores (Trigger)

- Permisos:
  - Para crear un Trigger DML, es necesario contar con permiso ALTER sobre la tabla o vista en la que se crea el desencadenador.
  - Para crear un Trigger DDL con ámbito de servidor (ON ALL SERVER) o un Trigger LOGON se requiere el permiso CONTROL SERVER en el servidor.
  - Para crear un desencadenador DDL con ámbito en la base de datos (ON DATABASE) es necesario un permiso ALTER ANY DATABASE DDL TRIGGER en la base de datos actual.

Ejemplos Trigger

- Ejemplo trigger LOGON, que valida que no hallan más de 3 sesiones de un mimo usuario. Verificar versión de SQL (select @@version)

```
USE master;
GO
CREATE LOGIN login_test WITH PASSWORD = '3KHJ6dhx(oxVYsdf' MUST_CHANGE,
CHECK_EXPIRATION = ON;
GO
GRANT VIEW SERVER STATE TO login_test;
GO
CREATE TRIGGER connection_limit_trigger  ON ALL SERVER
WITH EXECUTE AS 'login_test'
FOR LOGON
AS
BEGIN
        IF ORIGINAL_LOGIN()= 'login_test'
        AND (SELECT COUNT(*) FROM sys.dm_exec_sessions WHERE is_user_process = 1  AND
original_login_name = 'login_test') > 3 ROLLBACK;
END
```

Ejemplos Trigger

- Ejemplo trigger DML, imprime un mensaje en el cliente cuando alguien intenta agregar o cambiar datos en la tabla cargo.

```
USE Agenda
GO
CREATE TRIGGER reminder1 ON cargo
AFTER INSERT, UPDATE
AS
BEGIN
        RAISERROR ('No puede insertar ni modificar esta tabla', 16, 10)
END
```

Ejemplos Trigger

- Ejemplo trigger DDL, imprime un mensaje en el cliente cuando alguien intenta agregar o cambiar datos en la tabla Customer.

```
USE Agenda
GO
CREATE TRIGGER VALIDADOR
ON DATABASE
FOR CREATE_TABLE
AS
BEGIN
  PRINT 'CREATE TABLE .....'
  SELECT EVENTDATA().value('(/EVENT_INSTANCE/TSQLCommand)[1]','nvarchar(max)')
  RAISERROR ('No se pueden crear tablas en la base de datos.', 16, 1)
  ROLLBACK
END
```

Ejercicios

- Crear un trigger que cada vez que se modifique el sueldo base de un cargo, además varié la grafitización en el mismo porcentaje que la variación del sueldo base.

Referencias

- CREATE TRIGGER (Transact-SQL)
  http://msdn.microsoft.com/es-es/library/ms189799%28v=sql.100%29.aspx
- Tipos de desencadenadores DML
  http://msdn.microsoft.com/es-ar/library/ms178134%28v=sql.105%29.aspx
- Attempting to create a server-Level DDL trigger can result in a confusing error message
  http://connect.microsoft.com/SQLServer/feedback/details/124717/attempting-to-create-a-server-level-ddl-trigger-can-result-in-a-confusing-error-message