Taller	SQL SERVER PROGRAMACIÓN
Docente	Mg. Ing. Eric Gustavo Coronel Castillo
Tema	**DESENCADENANTES DDL**

INTRODUCCIÓN

Los desencadenantes DDL se pueden usar para auditar las operaciones de base de datos o de servidor que crean, modifican o eliminan objetos de base de datos o garantizar que las instrucciones DDL aplican las reglas de negocios antes de que se ejecuten.

Los desencadenantes DDL se inician en respuesta a una variedad de eventos de lenguaje de definición de datos (DDL). Estos eventos corresponden principalmente a las instrucciones Transact SQL que comienzan por las palabras clave CREATE, ALTER, DROP, GRANT, DENY, REVOKE o UPDATE STATISTICS. Algunos procedimientos almacenados del sistema que ejecutan operaciones de tipo DDL también pueden activar desencadenadores DDL.

> **Importante**
>
> Los desencadenantes DDL se pueden utilizar para verificar la respuesta a los procedimientos almacenados del sistema. Por ejemplo, la sentencia CREATE TYPE y el procedimiento almacenado SP_ADDTYPE activarán un desencadenante DDL activado por el evento CREATE_TYPE.

Cuando usar desencadenantes DDL:

- Para evitar determinados cambios en el esquema de la base de datos.
- Se quiere que ocurra algún evento en la base de datos como respuesta a un cambio realizado en el esquema de la base de datos.
- Se quiere registrar cambios o eventos del esquema de la base de datos.

Para mayor información sobre los eventos DDL consultar la siguiente dirección:

http://msdn.microsoft.com/es-es/library/bb522542.aspx

TIPOS DE DESENCADENANTES DDL

- Desencadenante Transact-SQL DDL

  Se trata de un tipo especial de procedimiento almacenado de Transact-SQL que ejecuta una o más instrucciones Transact-SQL en respuesta a un evento de servidor o de base de datos.

  Por ejemplo, un desencadenante DDL se puede activar si se ejecuta una instrucción como ALTER SERVER CONFIGURATION o si se elimina una tabla mediante DROP TABLE.

- **Desencadenante CLR DLL**

  En lugar de ejecutar un procedimiento almacenado Transact-SQL, un desencadenante CLR ejecuta uno o más métodos escritos en código administrado que son miembros de un ensamblado creado en .NET Framework y cargado en SQL Server.

Los desencadenantes DDL solo se activan cuando se ejecutan las instrucciones DDL que los desencadenan. Los desencadenadores DDL no se pueden usar como desencadenantes INSTEAD OF. Los desencadenantes DDL no se activan como respuesta a eventos que afectan a procedimientos almacenados y tablas temporales, ya sean locales o globales.

Los desencadenadores DDL no crean las tablas especiales **inserted** y **deleted**.

La información acerca de un evento que activa un desencadenante DDL y las modificaciones posteriores provocadas por el mismo se capturan con la función EVENTDATA.

Para cada evento DDL se pueden crear varios desencadenantes.

A diferencia de los desencadenantes DML, los desencadenantes DDL no tienen como ámbito los esquemas. Por tanto, las funciones como OBJECT_ID, OBJECT_NAME, OBJECTPROPERTY y OBJECTPROPERTYEX no se pueden usar para efectuar consultas en metadatos sobre desencadenantes DDL. En su lugar se debe usar las vistas de catálogo.

Los desencadenantes DDL con ámbito en el servidor aparecen en el Explorador de objetos de SQL Server Management Studio, en la carpeta **Triggers**. Dicha carpeta se encuentra en la carpeta **Server Objects**. Los desencadenantes DDL de ámbito de base de datos aparecen en la carpeta **Database Triggers**. Esta carpeta se encuentra en la carpeta **Programmability** de la base de datos correspondiente.

ÁMBITO DE LOS DESENCADENANTES DDL

Los desencadenantes DDL pueden activarse en respuesta a un evento de Transact-SQL procesado en la base de datos actual o en el servidor actual. El ámbito del desencadenante depende del evento.

Por ejemplo, un desencadenador DDL creado para activarse como respuesta al evento CREATE_TABLE se activará siempre que se produzca un evento CREATE_TABLE en la base de datos o en la instancia de servidor. Un desencadenador DDL creado para activarse como respuesta a un evento CREATE_LOGIN se activará únicamente cuando se produzca un evento CREATE_LOGIN en la instancia de servidor.

Ejemplo 1

En el siguiente ejemplo, el desencadenante DDL TR_SEGURIDAD se activará siempre que se produzca un evento DROP_TABLE o ALTER_TABLE en la base de datos, impidiendo que se elimine o modifique una tabla.

```
USE EDUCA;
GO

IF EXISTS ( SELECT 1 FROM sys.triggers WHERE name = 'tr_seguridad')
BEGIN
    DROP TRIGGER tr_seguridad ON DATABASE;
END;
GO

CREATE TRIGGER tr_seguridad
ON DATABASE
FOR DROP_TABLE, ALTER_TABLE
AS
BEGIN
    RAISERROR('No se permite eliminar ni modificar tablas.',16,1);
    ROLLBACK;
END;
GO

DROP TABLE dbo.PAGO;
GO
```

Se obtiene el siguiente mensaje:

```
Mens 50000, Nivel 16, Estado 1, Procedimiento tr_seguridad, Línea 7
No se permite eliminar ni modificar tablas.
Mens. 3609, Nivel 16, Estado 2, Línea 2
La transacción terminó en el desencadenador. Se anuló el lote.
```

Eliminemos el desencadenante:

```
DROP TRIGGER tr_seguridad ON DATABASE;
GO
```

Ejemplo 2

En el siguiente ejemplo, un desencadenante DDL imprime un mensaje si se produce algún evento CREATE_DATABASE en la instancia de servidor actual. El ejemplo usa la función EVENTDATA para recuperar el texto de la instrucción Transact-SQL correspondiente.

```
USE master;
GO

IF EXISTS (SELECT * FROM sys.server_triggers WHERE name = 'tr_ddl_database')
BEGIN
  DROP TRIGGER tr_ddl_database ON ALL SERVER;
END;
GO

CREATE TRIGGER tr_ddl_database
ON ALL SERVER
FOR CREATE_DATABASE
AS
BEGIN
  DECLARE  @data XML, @sql varchar(2000);
  set @data = EVENTDATA();
  set @sql = @data.value('(/EVENT_INSTANCE/TSQLCommand)[1]', 'varchar(2000)');
  PRINT 'Base de datos creada.'
  PRINT CONCAT('Sentencia: ', @sql);
END;
GO

CREATE DATABASE DEMO;
GO
```

Se obtiene el siguiente mensaje:

```
Base de datos creada.
Sentencia: CREATE DATABASE DEMO4;
```

Eliminemos el desencadenante:

```
DROP TRIGGER tr_ddl_database ON ALL SERVER;
GO
```

Los desencadenantes DDL de base de datos se almacenan como objetos en la base de datos donde se crean. Se puede utilizar la vista de catálogo sys.triggers para obtener información de estos desencadenantes.

Los desencadenantes DDL del servidor se almacenan como objetos en la base de datos master. Sin embargo, puede obtenerse información sobre los desencadenantes DDL de servidor si se consulta la vista de catálogo sys.server_triggers en cualquier contexto de base de datos.

MANTENIMIENTO DE DESENCADENANTES DDL

Creación de desencadenantes DDL

Los desencadenadores DDL se crean con la instrucción CREATE TRIGGER de Transact-SQL para desencadenadores DDL.

Para crear un desencadenantes:

```
CREATE TRIGGER . . .
```

Modificar desencadenantes DDL

Si necesita modificar la definición de un desencadenante DDL, puede quitarlo y volver a crear, o bien volver a definirlo en un solo paso.

Si cambia el nombre de un objeto al que hace referencia un desencadenante DDL, debe modificar el desencadenante para que el texto refleje el nuevo nombre. Por lo tanto, antes de cambiar el nombre de un objeto, vea primero las dependencias del mismo para determinar si algún desencadenante va a verse afectado por el cambio propuesto.

Para modificar un desencadenante:

```
ALTER TRIGGER . . .
```

Para ver las dependencias de un desencadenante:

- sys.sql_expression_dependencies
- sys.dm_sql_referenced_entities
- sys.dm_sql_referencing_entities

DESHABILITAR Y ELIMINAR DESENCADENANTES DDL

Se debe eliminar o deshabilitar un desencadenante DDL cuando ya no se necesite.

Al deshabilitar un desencadenante DDL, éste no se elimina de la base de datos. Sigue siendo un objeto de la base de datos actual. Sin embargo, el desencadenante no se activa cuando se ejecuta una instrucción Transact-SQL en la que se programó. Los desencadenantes DDL deshabilitados se pueden volver a habilitar. La habilitación de un desencadenante DDL hace que se active tal como lo hizo cuando se creó originalmente. Cuando se crean desencadenantes DDL, se habilitan de forma predeterminada.

Cuando el desencadenante DDL se elimina, se quita de la base de datos actual. Los objetos o datos incluidos en el ámbito del desencadenante DDL no se ven afectados.

Para deshabilitar un desencadenante DDL tenemos:

- DISABLE TRIGGER
- ALTER TABLE

Para habilitar un desencadenador DDL:

- ENABLE TRIGGER
- ALTER TABLE

Para eliminar un desencadenador DDL:

- DROP TRIGGER

EJEMPLOS

Ejemplo 3: Log de cambios en el sistema

El siguiente desencadenante crea un los para registrar los cambios que se realizan en la base de datos EDUTEC.

Creación de la base de datos de auditoría.

```
-- Creación de la base de datos
CREATE DATABASE AuditDB;
GO

-- Seleccionamos la base de datos
USE AuditDB;
GO

-- Tabla de auditoria.
CREATE TABLE dbo.DDLEvents
```

```
(
  EventDate    DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
  EventType    NVARCHAR(64),
  EventDDL     NVARCHAR(MAX),
  EventXML     XML,
  DatabaseName NVARCHAR(255),
  SchemaName   NVARCHAR(255),
  ObjectName   NVARCHAR(255),
  HostName     VARCHAR(64),
  IPAddress    VARCHAR(32),
  ProgramName  NVARCHAR(255),
  LoginName    NVARCHAR(255)
);
```

Creación del desencadenante en la base de datos EDUTEC.

```
-- Seleccionar la Base de Datos
USE EDUTEC;
GO

-- Creación del desencadenante
CREATE TRIGGER DDL_Change_Log
    ON DATABASE
    FOR  CREATE_PROCEDURE, ALTER_PROCEDURE, DROP_PROCEDURE,
         CREATE_TABLE, ALTER_TABLE, DROP_TABLE,
         CREATE_FUNCTION, ALTER_FUNCTION, DROP_FUNCTION
AS
BEGIN
  SET NOCOUNT ON;
  DECLARE
     @EventData XML = EVENTDATA(),
     @ip VARCHAR(32) =
         (
           SELECT client_net_address
           FROM sys.dm_exec_connections
           WHERE session_id = @@SPID
         );

  INSERT AuditDB.dbo.DDLEvents
  (
     EventType,
     EventDDL,
     EventXML,
     DatabaseName,
     SchemaName,
     ObjectName,
```

```
    HostName,
    IPAddress,
    ProgramName,
    LoginName
   )
  SELECT
    @EventData.value('(/EVENT_INSTANCE/EventType)[1]',   'NVARCHAR(100)'),
    @EventData.value('(/EVENT_INSTANCE/TSQLCommand)[1]', 'NVARCHAR(MAX)'),
    @EventData,
    DB_NAME(),
    @EventData.value('(/EVENT_INSTANCE/SchemaName)[1]',  'NVARCHAR(255)'),
    @EventData.value('(/EVENT_INSTANCE/ObjectName)[1]',  'NVARCHAR(255)'),
    HOST_NAME(),
    @ip,
    PROGRAM_NAME(),
    SUSER_SNAME();
END;
GO
```

Provocando el disparo del desencadenante.

```
-- Seleccionando la base de datos
USE EDUTEC;
GO

-- Creando un procedimiento almacenado
CREATE PROC dbo.usp_GetDBVersion
AS
SELECT @@version AS Vrsion;
GO
```

Consultando el log.

```
SELECT * FROM AuditDB.dbo.DDLEvents;
GO
```

Ejemplo 4: Log de cambios de inicios de sesión u usuarios

Creando la tabla en la base de datos master.

```
-- Creando la tabla

USE master;
GO

CREATE TABLE DDLSecurityLog
(
    DDLSecurityLog_ID int IDENTITY(1, 1) NOT NULL,
    InsertionDate datetime NOT NULL
           CONSTRAINT DF_ddl_log_InsertionDate
           DEFAULT ( GETDATE() ),
    CurrentUser nvarchar(50) NOT NULL
           CONSTRAINT DF_ddl_log_CurrentUser
           DEFAULT ( CONVERT(nvarchar(50), USER_NAME(), 0 ) ),
    LoginName nvarchar(50) NOT NULL
           CONSTRAINT DF_DDLSecurityLog_LoginName
           DEFAULT ( CONVERT(nvarchar(50), SUSER_SNAME(), ( 0 )) ),
    Username nvarchar(50) NOT NULL
           CONSTRAINT DF_DDLSecurityLog_Username
           DEFAULT ( CONVERT(nvarchar(50), original_login(), ( 0 )) ),
    EventType nvarchar(100) NULL,
    objectName nvarchar(100) NULL,
    objectType nvarchar(100) NULL,
    DatabaseName nvarchar(100) NULL,
    tsql nvarchar(MAX) NULL
) ;
GO
```

Creando el desencadenante de seguridad en el servidor.

```
-- Creando el trigger de seguridad para el servidor

USE master;
GO

IF EXISTS ( SELECT  *
            FROM    sys.server_triggers
            WHERE   name = 'trgLogServerSecurityEvents' )
    DROP TRIGGER trgLogServerSecurityEvents ON ALL SERVER;
GO
```

```
CREATE TRIGGER trgLogServerSecurityEvents ON ALL SERVER
    FOR CREATE_LOGIN, ALTER_LOGIN, DROP_LOGIN, GRANT_SERVER, DENY_SERVER,
        REVOKE_SERVER, ALTER_AUTHORIZATION_SERVER
AS
BEGIN
  DECLARE @data XML
  SET @data = EVENTDATA()
  INSERT  INTO master..DDLSecurityLog
  (
     EventType,
     ObjectName,
     ObjectType,
     DatabaseName,
     tsql
  )
  VALUES
  (
     @data.value('(/EVENT_INSTANCE/EventType)[1]', 'nvarchar(100)'),
     @data.value('(/EVENT_INSTANCE/ObjectName)[1]', 'nvarchar(100)'),
     @data.value('(/EVENT_INSTANCE/ObjectType)[1]', 'nvarchar(100)'),
     'Server',
     @data.value('(/EVENT_INSTANCE/TSQLCommand)[1]', 'nvarchar(max)')
  );
END;
GO
```

Desencadenante de seguridad en cada base de datos.

```
USE RH;
GO

CREATE TRIGGER trgLogDatabaseSecurityEvents
ON DATABASE
FOR DDL_DATABASE_SECURITY_EVENTS
AS
BEGIN
  DECLARE @data XML;
  SET @data = EVENTDATA();
  INSERT  INTO master..DDLSecurityLog
  (
     EventType,
     ObjectName,
     ObjectType,
     DatabaseName,
     tsql
  )
```

```
    VALUES
  (
    @data.value('(/EVENT_INSTANCE/EventType)[1]', 'nvarchar(100)'),
    @data.value('(/EVENT_INSTANCE/ObjectName)[1]', 'nvarchar(100)'),
    @data.value('(/EVENT_INSTANCE/ObjectType)[1]', 'nvarchar(100)'),
    @data.value('(/EVENT_INSTANCE/DatabaseName)[1]', 'nvarchar(max)'),
    @data.value('(/EVENT_INSTANCE/TSQLCommand)[1]', 'nvarchar(max)')
  );
END;
GO
```

Forzando el disparo de los desencadenantes.

```
USE master;
GO

CREATE LOGIN intruso
    WITH
          PASSWORD= N'123456',
          DEFAULT_DATABASE = RH;
GO

EXEC master..sp_addsrvrolemember
    @loginame = N'intruso',
    @rolename = N'sysadmin';
GO

USE RH;
GO

CREATE USER intruso FOR LOGIN intruso
GO
```