1. T-SQL PROGRAMACIÓN

TRANSACT-SQL (T-SQL) expande el estándar de SQL para incluir programación procedimental, variables locales, varias funciones de soporte para procesamiento de strings, procesamiento de fechas, matemáticas, etc., así cambios en algunas sentencias (DELETE, UPDATE, etc.). Estas características adicionales hacen de T-SQL que cumpla con muchas características propias de un lenguaje de programación.

es un lenguaje muy potente que nos permite definir casi cualquier tarea que queramos efectuar sobre la base de datos; incluye características propias de cualquier lenguaje de programación, características que nos permiten definir la lógica necesaria para el tratamiento de la información:

- Tipos de datos.
- Definición de variables.
- Estructuras de control de flujo.
- Gestión de excepciones.
- Funciones predefinidas.

Sin embargo no nos permite:

- Crear interfaces de usuario.
- Crear aplicaciones ejecutables, sino elementos que en algún momento llegarán al servidor de datos y serán ejecutados.

Debido a estas restricciones se emplea generalmente para crear procedimientos almacenados, triggers y funciones de usuario.

Puede ser utilizado como cualquier SQL como lenguaje embebido en aplicaciones desarrolladas en otros lenguajes de programación como Visual Basic, C, Java, C#, etc.

También lo podremos ejecutar directamente de manera interactiva, por ejemplo desde el editor de consultas de SSMS (SQL Server Management Studio) que usamos en las prácticas.

De momento sólo vamos a ver algunas cosas básicas, que permitan entender los ejemplos sobre disparadores (triggers) del siguiente punto.

Algunas palabras clave generales y del lenguaje de control de flujo de Transact-SQL son:

PRINT: Devuelve al cliente un mensaje definido por el usuario.

```
PRINT msg_str | @local_variable | string_expr
```

EJEMPLO: PRINT 'Fila insertada correctamente'

BEGIN...END: Incluye una serie de instrucciones Transact-SQL de forma que se pueda ejecutar un grupo de instrucciones Transact-SQL. BEGIN y END son palabras clave del lenguaje de control de flujo.

```
BEGIN
    { sql_statement | statement_block }
END
```

EJEMPLO:

```
IF (SELECT COUNT (*) FROM INSERTED, COMPRAS.PRODUCTOS
WHERE INSERTED.NOMPRODUCTO = PRODUCTOS.NOMPRODUCTO) >1
BEGIN
ROLLBACK TRANSACTION
PRINT 'LA DESCRIPCION DEL PRODUCTO SE ENCUENTRA REGISTRADO'
END
```

IF...ELSE: Impone condiciones en la ejecución de una instrucción de Transact-SQL. La instrucción Transact-SQL que sigue a una palabra clave IF y a su condición se ejecuta si la condición se cumple: la expresión booleana devuelve TRUE. La palabra clave opcional ELSE introduce otra instrucción Transact-SQL que se ejecuta cuando la condición IF no se cumple: la expresión booleana devuelve FALSE.

```
IF Boolean_expression
     { sql_statement | statement_block }
[ ELSE
     { sql_statement | statement_block } ]
```

EJEMPLO:

```
IF (select count(*) from aviones,inserted
    where aviones.tipo=inserted.tipo_avion)!=@@rowcount
BEGIN
    ...
END
```

2. TRIGGERS O DISPARADORES (DESENCADENADORES)

Los disparadores pueden usarse para imponer la integridad referencial de los datos en toda la base de datos, o bien para realizar una gestión avanzada sobre los datos.

Los disparadores también permiten realizar cambios "en cascada" en tablas relacionadas, imponer restricciones de columna más complejas que las permitidas por las reglas (Constraint), comparar los resultados de las modificaciones de datos y llevar a cabo una acción resultante.

TABLAS TEMPORALES INSERTED Y DELETED
En las instrucciones de triggers se usan dos tablas temporales especiales: la tabla ***inserted*** y la tabla ***deleted***. SQL Server crea y administra automáticamente ambas tablas.

Puede utilizar estas tablas temporales residentes en memoria para probar los efectos de determinadas modificaciones de datos y para establecer condiciones para las acciones de los triggers.

No puede modificar directamente los datos de estas tablas ni realizar en ellas operaciones de lenguaje de definición de datos (DDL), como CREATE INDEX.

En los triggers, las tablas ***inserted*** y ***deleted*** se utilizan principalmente para realizar las siguientes tareas:

- Ampliar la integridad referencial entre tablas.
- Insertar o actualizar datos de tablas base subyacentes a una vista.
- Comprobar errores y realizar acciones en función del error.
- Conocer la diferencia entre el estado de una tabla antes y después de realizar una modificación en los datos, y actuar en función de dicha diferencia.

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 del trigger y se transfieren a la tabla deleted. La tabla deleted y la tabla del trigger no suelen tener filas en común.

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 trigger. Las filas de la tabla inserted son copias de las nuevas filas de la tabla del trigger.

Una transacción de actualización (UPDATE) 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 trigger y en la tabla inserted.
Cuando ser trabaja con triggers, se utilizan las tablas inserted y deleted correspondientes a la acción que lo activó.
Aunque no se produce ningún error al hacer referencia a la tabla deleted cuando se prueba una instrucción INSERT, o bien al hacer referencia a la tabla inserted cuando se prueba una instrucción DELETE, estas tablas de prueba del trigger no contendrán filas en estos casos.

DEFINICIÓN DEL DISPARADOR

Un disparador (DML) es un tipo especial de procedimiento almacenado que **se ejecuta cuando se insertan, eliminan o actualizan datos** de una tabla especificada, en general cuando tiene lugar un evento de lenguaje de manipulación de datos (DML) que afecta a la tabla o la vista definida en el desencadenador.

Los disparadores permiten aplicar reglas y restricciones, consultar otras tablas e incluir instrucciones Transact-SQL complejas, pero también ayudan a mantener la integridad referencial de los datos conservando la consistencia entre los datos relacionados lógicamente entre distintas tablas. Recordemos que la integridad referencial significa que los valores de las llaves primarias y los valores correspondientes de las llaves foráneas deben coincidir de forma exacta. Los desencadenadores DML se parecen a las restricciones en que pueden aplicar la integridad de entidad o de dominio. En general, la integridad de entidad se debe exigir siempre en el nivel más bajo mediante índices que formen parte de las restricciones PRIMARY KEY y UNIQUE o que se creen independientemente de las restricciones. La integridad de dominio se debe aplicar con restricciones CHECK y la integridad referencial (RI) se debe aplicar a las restricciones de FOREIGN KEY.

El desencadenador (trigger) y la instrucción (INSERT, UPDATE o DELETE) que lo activa se tratan como una sola transacción, que puede revertirse desde el propio desencadenador. Si se detecta un error grave (por ejemplo, no hay suficiente espacio en disco), se revierte automáticamente toda la transacción.

Los desencadenadores DML resultan de especial utilidad cuando las características permitidas por las restricciones no cubren las necesidades funcionales de la aplicación.

La principal ventaja de los disparadores es que son automáticos: funcionan cualquiera sea el origen de la modificación de los datos.

Cada disparador es específico de una o más operaciones de modificación de datos, ***UPDATE, INSERT o DELETE***. El disparador se ejecuta una vez por cada instrucción.

CREACIÓN DE DISPARADORES

Un disparador es un objeto de la base de datos. Cuando se crea un disparador, se especifica la tabla y los comandos de modificación de datos que deben "disparar" o activar el disparador. Luego, se indica la acción o acciones que debe llevar a cabo un disparador.

```
CREATE [ OR ALTER ] TRIGGER [ esquema. ]nombre_trigger
ON { Tabla | Vista }
{ FOR | AFTER | INSTEAD OF }
{ [ INSERT ] [ , ] [ UPDATE ] [ , ] [ DELETE ] }
AS
sentencias sql [ ; ]
```

SINTAXIS COMPLETA T-SQL

https://docs.microsoft.com/es-es/sql/t-sql/statements/create-trigger-transact-sql?view=sql-server-ver15

A continuación, se muestra un ejemplo sencillo. Este disparador imprime un mensaje cada vez que alguien trata de insertar, actualizar o eliminar datos de la tabla Productos.

```
CREATE TRIGGER TX_Productos
ON Compras.Productos
FOR INSERT, UPDATE, DELETE
AS
PRINT 'Actualización de los registros de Productos'
```

Una vez creado un trigger se puede ver la dependencia del mismo en relación a la tabla.

Para modificar el *TRIGGER*, se utiliza la siguiente sintaxis:

```
ALTER TRIGGER TX_PRODUCTOS
ON COMPRAS.PRODUCTOS
FOR INSERT, UPDATE, DELETE
AS
PRINT 'ACTUALIZACION DE LOS REGISTROS DE PRODUCTOS'
```

Para borrar un *TRIGGER*, se utiliza la siguiente sintaxis:

```
DROP TRIGGER TX_PRODUCTOS
```

FUNCIONAMIENTO DE LOS DISPARADORES

A. DISPARADOR DE INSERCIÓN

Cuando se inserta una nueva fila en una tabla, *SQL Server* inserta los nuevos valores en la tabla ***INSERTED*** el cual es una tabla del sistema. Está tabla toma la misma estructura de la cual se originó el *TRIGGER*, de tal manera que se pueda verificar los datos y ante un error podría revertirse los cambios.

EJEMPLO: Cree un *TRIGGER* que permita insertar los datos de un Producto siempre y cuando la descripción o nombre del producto sea único.

```
CREATE TRIGGER TX_PRODUCTO_INSERTA
ON COMPRAS.PRODUCTOS
FOR INSERT
AS
IF (SELECT COUNT (*) FROM INSERTED, COMPRAS.PRODUCTOS
WHERE INSERTED.NOMPRODUCTO = PRODUCTOS.NOMPRODUCTO) >1
BEGIN
ROLLBACK TRANSACTION
PRINT 'LA DESCRIPCION DEL PRODUCTO SE ENCUENTRA REGISTRADO'
END
ELSE
PRINT 'EL PRODUCTO FUE INGRESADO EN LA BASE DE DATOS'
GO
```

En este ejemplo, verificamos el número de productos que tienen la misma descripción y de encontrarse más de un registro de productos no se deberá permitir ingresar los datos del producto. Este disparador imprime un mensaje si la inserción se revierte y otro si se acepta.

B. DISPARADOR DE ELIMINACIÓN

Cuando se elimina una fila de una tabla, *SQL Server* inserta los valores que fueron eliminados en la tabla ***DELETED*** el cual es una tabla del sistema. Está tabla toma la misma estructura de la cual se originó el *TRIGGER*, de tal manera que se pueda verificar los datos y ante un error podría revertirse los cambios. En este caso, la reversión de los cambios significará restaurar los datos eliminados.

EJEMPLO: Cree un *TRIGGER* el cual permita eliminar Clientes sólo si no tienen registrado algún pedido. De eliminarse algún Cliente que no cumpla con dicha condición la operación no deberá ejecutarse.

```
CREATE TRIGGER TX_ELIMINA_ELIMINA
ON VENTAS.CLIENTES
FOR DELETE
AS
IF EXISTS (SELECT * FROM VENTAS.PEDIDOSCABE
WHERE PEDIDOSCABE.IDCLIENTE = (SELECT IDCLIENTE FROM DELETED) )
BEGIN
ROLLBACK TRANSACTION
PRINT 'EL CLIENTE TIENE REGISTRADO POR LO MENOS 1 PEDIDOS'
END
```

En este ejemplo, verificamos si el cliente tiene pedidos registrados, de ser así la operación deberá ser cancelada.

C. DISPARADOR DE ACTUALIZACIÓN

Cuando se actualiza una fila de una tabla, *SQL Server* inserta los valores que antiguos en la tabla ***DELETED*** y los nuevos valores los inserta en la tabla ***INSERTED***. Usando estas dos tablas se podrá verificar los datos y ante un error podrían revertirse los cambios.

EJEMPLO: Cree un *TRIGGER* que valide el precio unitario y su Stock de un producto, donde dichos datos sean mayores a cero.

```
CREATE TRIGGER TX_PRODUCTO_ACTUALIZA
ON COMPRAS.PRODUCTOS
FOR UPDATE
AS
IF (SELECT PRECIOUNIDAD FROM INSERTED) <=0 OR
(SELECT UNIDADESENEXISTENCIA FROM INSERTED)<=0
BEGIN
PRINT 'EL PRECIO O UNIDADESENEXISTENCIA DEBEN SER MAYOR A CERO'
ROLLBACK TRANSACTION
END
```

EJEMPLO: Cree un *TRIGGER* que bloquee actualizar el id del producto en el proceso de actualización.

```
CREATE TRIGGER TX_PRODUCTO_ACTUALIZA_ID
ON COMPRAS.PRODUCTOS
FOR UPDATE
AS
IF UPDATE(IDPRODUCTO)
BEGIN
PRINT 'NO SE PUEDE ACTUALIZAR EL ID DEL PRODUCTO'
ROLLBACK TRANSACTION
END
```

USO DE INSTEAD OF

Los trigger de tipo INSTEAD OF pasan por alto las acciones estándar de la instrucción de desencadenamiento: INSERT, UPDATE o DELETE. Se puede definir un trigger INSTEAD OF para realizar comprobación de errores o valores en una o más columnas y, a continuación, realizar acciones adicionales antes de insertar el registro.

Por ejemplo, cuando el valor que se actualiza en un campo de tarifa de una hora de trabajo de una tabla de planilla excede un valor específico, se puede definir un trigger para producir un error y revertir la transacción o insertar un nuevo registro en un registro de auditoría antes de insertar el registro en la tabla de planilla.

A. INSTEAD OF INSERT

Se pueden definir trigger INSTEAD OF INSERT en una vista o tabla para reemplazar la acción estándar de la instrucción INSERT. Normalmente, el trigger INSTEAD OF INSERT se define en una vista para insertar datos en una o más tablas base. Las columnas de la lista de selección de la vista pueden o no admitir valores NULL. Si una columna de la vista no admite valores NULL, una instrucción INSERT debe proporcionar los valores para la columna.

```
-- CREAR UNA VISTA QUE LISTE LAS COLUMNAS DE CLIENTES.
CREATE VIEW INSTEADVIEW
AS
SELECT IDCLIENTE,
NOMCLIENTE,
DIRCLIENTE,
IDPAIS,
FONOCLIENTE
FROM VENTAS.CLIENTES
GO
-- CREAR UN TRIGGER INSTEAD OF INSERT PARA LA VISTA.
CREATE TRIGGER INSTEADTRIGGER ON INSTEADVIEW
INSTEAD OF INSERT
AS
BEGIN
INSERT INTO VENTAS.CLIENTESBAK
SELECT IDCLIENTE, NOMCLIENTE, DIRCLIENTE, IDPAIS, FONOCLIENTE
FROM INSERTED
END
GO
```

B. INSTEAD OF UPDATE

Se pueden definir triggers INSTEAD OF UPDATE en una vista o tabla para reemplazar la acción estándar de la instrucción UPDATE. Normalmente, el trigger INSTEAD OF UPDATE se define en una vista para modificar datos en una o más tablas. Las instrucciones UPDATE que hacen referencia a vistas que contienen triggers INSTEAD OF UPDATE deben suministrar valores para todas las columnas que no admiten valores NULL a las que hace referencia la cláusula SET.

C. INSTEAD OF DELETE

Se pueden definir triggers INSTEAD OF DELETE en una vista o una tabla para reemplazar la acción estándar de la instrucción DELETE. Normalmente, el trigger INSTEAD OF DELETE se define en una vista para modificar datos en una o más tablas. Las instrucciones DELETE no especifican modificaciones de los valores de datos existentes. Las instrucciones DELETE sólo especifican las filas que se van a eliminar.

La tabla **inserted** pasada a un trigger DELETE siempre está vacía. La tabla **deleted** enviada a un trigger DELETE contiene una imagen de las filas en el estado que tenían antes de emitir la instrucción DELETE.

ÚTILES PARA USAR EN DISPARADORES

- @@ROWCOUNT: Variable del sistema que devuelve el número de filas afectadas por la última instrucción. Se suele usar en condiciones detrás de AS, dentro de un trigger.

EJEMPLO: Cree un *TRIGGER* que controle los aviones a insertar en la tabla VUELOS.

```
CREATE TRIGGER forinsert1
ON vuelos
FOR insert
AS
IF (select count(*) from aviones,inserted
    where aviones.tipo=inserted.tipo_avion)!=@@rowcount
BEGIN
    ROLLBACK TRANSACTION
    PRINT 'No se puede insertar ese vuelo, no existe ese tipo de avión.'
END
ELSE
    PRINT 'Nuevo vuelo insertado'
```

TPL (Transaction Procesing Language):

- ROLLBACK TRANSACTION: Revierte una transacción explícita o implícita hasta el inicio de la transacción o hasta un punto de retorno dentro de la transacción.
- COMMIT TRANSACTION: Marca el final de una transacción correcta, implícita o explícita.

RESTRICCIONES DE LOS DISPARADORES

A continuación, se describen algunas limitaciones o restricciones impuestas a los disparadores por SQL Server:

- Una tabla puede tener un máximo de tres disparadores: uno de actualización, uno de inserción y uno de eliminación.
- Cada disparador puede aplicarse a una sola tabla. Sin embargo, un mismo disparador se puede aplicar a las tres acciones del usuario: *UPDATE, INSERT* y *DELETE*.
- No se puede crear un disparador en una vista ni en una tabla temporal, aunque los disparadores pueden hacer referencia a las vistas o tablas temporales.
- Los disparadores no se permiten en las tablas del sistema. Aunque no aparece ningún mensaje de error si crea un disparador en una tabla del sistema, el disparador no se utilizará.

RESUMEN

- Una transacción es un conjunto de operaciones **TRANSACT SQL** que se ejecutan como un único bloque, es decir, si falla una operación **TRANSACT SQL** fallan todas. Si una transacción tiene éxito, todas las modificaciones de los datos realizadas durante la transacción se confirman y se convierten en una parte permanente de la base de datos. Si una transacción encuentra errores y debe cancelarse o revertirse, se borran todas las modificaciones de los datos.
- Los disparadores pueden usarse para imponer la integridad de referencia de los datos en toda la base de datos. Los disparadores también permiten realizar cambios “en cascada” en tablas relacionadas, imponer restricciones de columna más complejas que las permitidas por las reglas, compara los resultados de las modificaciones de datos y llevar a cabo una acción resultante.
- Cuando se inserta una nueva fila en una tabla, *SQL Server* inserta los nuevos valores en la tabla ***INSERTED*** el cual es una tabla del sistema. Está tabla toma la misma estructura del cual se originó el *TRIGGER*, de tal manera que se pueda verificar los datos y ante un error podría revertirse los cambios
- Cuando se elimina una fila de una tabla, *SQL Server* inserta los valores que fueron eliminados en la tabla ***DELETED*** el cual es una tabla del sistema. Está tabla toma la misma estructura del cual se originó el *TRIGGER*, de tal manera que se pueda verificar los datos y ante un error podría revertirse los cambios. En este caso la reversión de los cambios significará restaurar los datos eliminados.
- Cuando se actualiza una fila de una tabla, *SQL Server* inserta los valores antiguos en la tabla ***DELETED*** y los valores nuevos en la tabla ***INSERTED***. Usando estas dos tablas se podrá verificar los datos y ante un error podrían revertirse los cambios.
- Los triggers usan las tablas lógicas (conceptuales) *inserted* y *deleted*.
- Son de estructura similar a la tabla en que se define el trigger, es decir, la tabla en que se intenta la acción del usuario.
- Las tablas *inserted* y *deleted* contienen los valores antiguos o nuevos de las filas que pueden modificarse mediante la acción del usuario. Por ejemplo, para recuperar todos los valores de la tabla deleted, utilice: `SELECT * FROM deleted`
- Para obtener más información, vea Usar las tablas inserted y deleted.

EJEMPLO: Actualizar un atributo derivado

```
CREATE TRIGGER N_Empleados_Dpto
ON Empleados
FOR INSERT
AS
IF (@@rowcount=1)
BEGIN
	UPDATE Departamento
	SET N_Empleados=N_Empleados+1
	WHERE Cod_Dpto = (SELECT Cod_Dpto FROM inserted)

	PRINT 'Atributo derivado NEmpleados actualizado'
END
```