INFORMATORIO

BASE DE DATOS

Lenguaje SQL: Triggers y Store Procedures

Subsecretaría de Empleo
Chaco Gobierno de todos

Ministerio de Producción, Industria y Empleo
Chaco Gobierno de todos

CHACO
Gobierno de todos

1. No se puede ejecutar una operación INSERT/UPDATE/DELETE sobre la misma tabla donde el

TRIGGER se está ejecutando.

SQL Procedural

Posibilita el **uso de código procedural** conjuntamente con **sentencias SQL que son almacenadas dentro de la BD**.
El código procedural es ejecutado por el DBMS cuando es invocado (directa o indirectamente) por el usuario de la BD.

Se puede utilizar SQL Procedural para definir:

- **Trigger**: Es un procedimiento que es invocado automáticamente por el DBMS en respuesta a un evento específico de la BD. Por ejemplo, dada una tabla con información sobre «expedientes», se puede crear un Trigger que controle las modificaciones del «estado del expediente» y actualice algún dato.
- **Stored Procedure:** Es un procedimiento que es invocado explícitamente por el usuario.
- **Función:** definida por el usuario para realizar operaciones específicas sobre los datos, y pueden ser invocadas desde un trigger, stored procedure o explícitamente.

Cuál es la principal diferencia entre Triggers y Store Procedures

La principal diferencia entre los triggers y stored procedures: Es que los triggers son procedimientos que se ejecutan automáticamente, cuando se produce un evento sobre el que se quiere trabajar. Para esto existen tres tipos de eventos que pueden disparar un trigger: INSERT, DELETE y UPDATE. El trigger se programa para realizar una tarea determinada que se debe hacer siempre que se produzca uno de los eventos antes mencionados. No requiere intervención humana o programática y no se puede detener. Tiene algunas características:

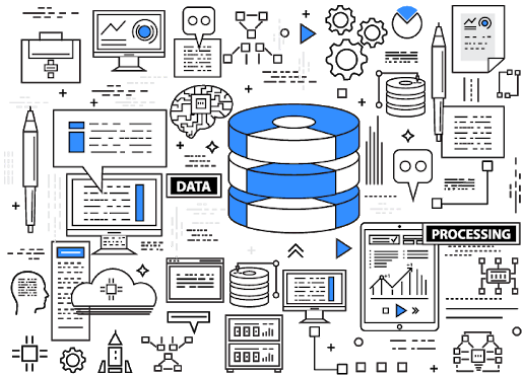
2. No recibe parámetros de entrada o salida.
3. Los únicos valores de entrada son los correspondientes a los de las columnas que se insertan, y sólo son accesibles por medio de ciertas pseudovariables (NEW y OLD).
4. No se puede ejecutar una tarea sobre otra tabla, si la segunda tiene un trigger que afecte a la tabla del primer trigger en ejecución (circularidad).
5. No se puede invocar procedures desde un TRIGGER.
6. No se puede invocar un SELECT que devuelva una tabla resultado en el TRIGGER.

En cambio, un **stored procedure** es un procedimiento almacenado que debe ser invocado para ejecutarse.

1. Puede recibir parámetros y devolver parámetros.
2. Puede manejar cualquier tabla, realizar operaciones con ellas y realizar iteraciones de lectura/escritura.
3. Puede devolver una tabla como resultado. también valores dentro de los parámetros del prototipo si los mismos son también de salida.
4. Existen en la base donde se crean, pero no dependen de ninguna tabla.
5. Pueden aceptar recursividad (pero no es recomendable).

INFORMATORIO

STORE PROCEDURES



Un **store procedure** o procedimiento almacenado es un **programa** dentro de la base de datos que **ejecuta una acción o conjunto de acciones específicas**.

Un procedimiento tiene un *nombre*, un *conjunto de parámetros* (opcional) y un *bloque de código*.

Los procedimientos almacenados pueden devolver **valores (numérico entero) o conjuntos de resultados**.

Tipos de Store Procedures

- **Procedimientos almacenados definidos por el usuario**: son procedimientos definidos por el usuario que se debe llamar explícitamente.
- **Triggers:** Son procedimientos definidos por el usuario que se ejecutan automáticamente cuando se modifica un dato en un tabla.
- **Procedimientos del sistema:** Procedimientos suministrados por el sistema.
- **Procedimientos Extendidos:** Procedimientos que se hacen llamadas al sistema operativo y ejecutan tareas a ese nivel.

Sintaxis

1. Crear el procedimiento almacenado con el comando CREATE PROCEDURE seguido del nombre que le quieras asignar.
2. Instrucción BEGIN para indicar que empieza el código SQL del procedimiento en SQL.
3. Código SQL que queramos que se ejecute cuando se llame a la rutina.
4. Cerrar el código con la cláusula END seguida de los caracteres definidos con el comando DECLARE.

```
CREATE PROCEDURE nombre_procedimiento (IN parametros)
BEGIN
    Sentencias.
END
```

Ejemplo 1

```
1. CREATE PROCEDURE total_paises
2. BEGIN
3.  SELECT COUNT(*)
4.  FROM pais
5. END //
```

Este sencillo ejemplo permite mostrar el **total de paises** almacenados en una **tabla pais** con una simple llamada. Para llamarlo usamos la siguiente línea SQL:

```
CALL total_paises();
```

Ejemplo 2

```
1. CREATE PROCEDURE total_paises_nombrados_como
2. (IN palabra CHAR(20))
3. BEGIN
4.  SELECT COUNT(*) FROM pais
5.  WHERE nombre LIKE palabra;
6. END //
```

Este "stored procedure" es una rutina SQL que permite calcular el total de paises en una tabla pais que coinciden con una "palabra" pasada como parámetro de entrada. Hay que tener en cuenta que la palabra solo puede tener cómo máximo 20 caracteres ya que el parámetro de entrada "palabra" está definido como CHAR(20).

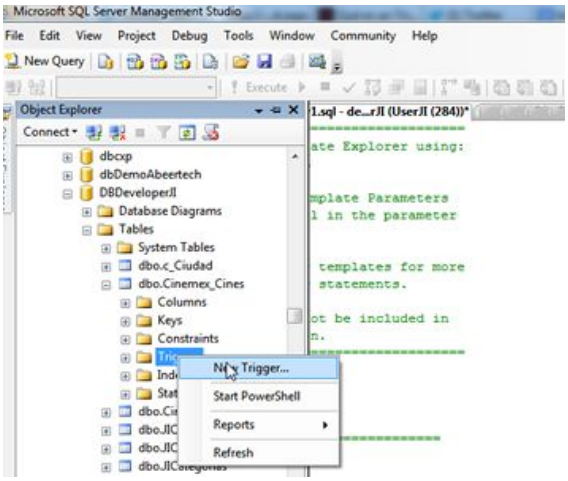
Ejemplo 3

```
7. CREATE PROCEDURE total_paises (OUT total INTEGER)
8. BEGIN
9.  SELECT COUNT(*) INTO total FROM pais;
10. END //
```

Este procedimiento calcula el total de registros de la tabla pais y los inserta con INTO en la variable de salida OUT.

Los Triggers

Un **trigger** es una clase especial de procedimiento almacenado que se ejecuta automáticamente cuando se produce un evento en el servidor de bases de datos. Los trigger 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. Estos triggers se activan cuando se desencadena cualquier evento válido, con independencia de que las filas de la tabla se vean o no afectadas.



Los componentes de un trigger son:

- Un nombre único que identifica al trigger dentro de la base.
- Un evento de disparo asociado (INSERT, UPDATE o DELETE).
- Una condición que puede ser cualquier condición válida
- Una acción que puede ser un conjunto de sentencias SQL

La sentencia que se utiliza para añadir triggers al esquema de base de datos es CREATE TRIGGER. La sintaxis general de un trigger es la siguiente:

```
CREATE TRIGGER <Schema_Name, sysname, Schema_Name>.<Trigger_Name,
sysname, Trigger_Name>
   ON  <Schema_Name, sysname, Schema_Name>.<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
```

Los triggers utilizan **dos tablas virtuales** denominadas **inserted** y **deleted**.
SQL Server 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 tabla virtual Inserted solo está disponible en las operaciones INSERT y UPDATE y en ella están los valores resultantes después de la inserción o actualización.
La tabla Deleted está disponible en las operaciones UPDATE y DELETE, los valores que tiene esta tabla son los anteriores a la ejecución de la actualización o borrado. Es decir, los datos que serán borrados.

Veamos un ejemplo:

```
--Crea un trigger llamado AddCines
CREATE TRIGGER AddCines
--Se ejecutara en la tabla Cinemex_Ciudades
   ON  Cinemex_Ciudades
--Se ejecutara despues de un Insert o un Update a la tabla
   AFTER INSERT,UPDATE
AS
BEGIN
      -- SET NOCOUNT ON impide que se generen mensajes de texto con cada
instrucción
      SET NOCOUNT ON;
    -- Se crea un Insert: cuando se inserten valores en la tabla
Cinemex_Ciudades, el trigger insertará un registro en la tabla Cinemex_Cines
    INSERT INTO Cinemex_Cines
    (ID, IDCiudad, Cine, Direccion)
    SELECT '500', ID, 'Cinemex ' + Ciudad, 'Prueba'
    FROM INSERTED
--Los valores que se insertaran, seran los que estén almacenados en la tabla
virtual Inserted
END
GO
```