

INSTITUTO TECNOLÓGICO SUPERIOR DE JEREZ

JEREZ DE GARCÍA SALINAS A 08 DE NOVIEMBRE DEL 2019

NOMBRE:

GUADALUPE VÁZQUEZ DE LA TORRE

NUMERO DE CONTROL:

S17070158

CORREO:

guvadlt@Outlook.com

CARRERA:

INGENIERÍA EN SISTEMAS COMPUTACIONALES

NOMBRE DE LA MATERIA:

TALLER DE BASES DE DATOS

QUINTO SEMESTRE

TEMA 5 - SQL PROCEDURAL

**“ACTIVIDAD 1 - MAPA CONCEPTUAL”**

DOCENTE:

SALVADOR ACEVEDO SANDOVAL

**1)¿Qué es un procedimiento almacenado en una base de datos y para qué sirve?**

Un procedimiento almacenado es un grupo de una o varias instrucciones Transact-SQL o una referencia a un método de Common Runtime Language. Los procedimientos se asemejan a las construcciones de otros lenguajes de programación.

* Aceptar parámetros de entrada y devolver varios valores en forma de parámetros de salida al programa que realiza la llamada.
* Contener instrucciones de programación que realicen operaciones en la base de datos. Entre otras, pueden contener llamadas a otros procedimientos.
* Devolver un valor de estado a un programa que realiza una llamada para indicar si la operación se ha realizado correctamente o se han producido errores, y el motivo de estos.

Definidos por el usuario

Se puede crear en una base de datos definida por el usuario o en todas las bases de datos del sistema excepto en la base de datos Resource

Temporales

Los procedimientos temporales son iguales que los procedimientos permanentes salvo porque se almacenan en tempdb.

Los procedimientos temporales locales tienen como primer carácter de sus nombres un solo signo de número (#); solo son visibles en la conexión actual del usuario y se eliminan cuando se cierra la conexión.

Los procedimientos temporales globales presentan dos signos de número (##) antes del nombre; son visibles para cualquier usuario después de su creación y se eliminan al final de la última sesión en la que se usa el procedimiento.

Sistema

Están almacenados físicamente en la base de datos interna y oculta Resource y se muestran de forma lógica en el esquema sys de cada base de datos definida por el sistema y por el usuario.

Extendidos definidos por el usuario

Le permiten crear sus propias rutinas externas en un lenguaje de programación.

**2) ¿Qué es una FUNCIÓN en una base de datos y para qué sirve?**

Es una rutina creada para tomar unos parámetros, procesarlos y retornar en un salida.

**3) ¿Cuál es la diferencia entre procedimiento y función?**

Las funciones se diferencian de los procedimientos en las siguientes características:

* Solamente pueden tener parámetros de entrada IN y no parámetros de salida OUT oINOUT
* Deben retornar en un valor con algún tipo de dato definido
* Pueden usarse en el contexto de una sentencia SQL
* Solo retornan un valor individual, no un conjunto de registros.

**4) ¿Qué es un disparador (trigger) en una base de datos y para qué sirve?**

Es un programa almacenado (stored program), creado para ejecutarse automáticamente cuando ocurra un evento en nuestra base de datos. Dichos eventos son generados por los comandos INSERT, UPDATE y DELETE, los cuales hacen parte del DML(Data Modeling Languaje) de SQL.

**5) ¿Cuáles son los tiempos o momentos en los que puede activarse un trigger?**

Una misma tabla puede tener varios triggers asociados. En tal caso es necesario conocer el orden en el que se van a ejecutar.

Los disparadores se activan al ejecutarse la sentencia SQL.

* Si existe, se ejecuta el disparador de tipo BEFORE (disparador previo) con nivel de orden.
* Para cada fila a la que afecte la orden:
* Se ejecuta si existe, el disparador de tipo BEFORE con nivel de fila.
* Se ejecuta la propia orden.
* Se ejecuta si existe, el disparador de tipo AFTER (disparador posterior) con nivel de fila.
* Se ejecuta, si existe, el disparador de tipo AFTER con nivel de orden.

**6) ¿Cuáles son los eventos que puede generar un trigger?**

* Antes de: son los que se ejecutan antes de que se produzca la actualización de los datos: antes de insertar un registro, antes de actualizar un registro y antes de que se elimine un registro.

Estos desencadenantes tienen asociadas dos Macros de Datos predefinidas:

* + Antes del cambio: para los eventos de inserción y actualización. Podemos aprovecharlos por ejemplo para impedir que se realice una venta si no hay suficientes existencias del artículo.
  + Validar eliminación: para el evento de eliminación.
* Después de: son los que se ejecutan después de que se haya realizado la actualización de los datos: después de insertar el registro, después de haber actualizado el registro y después de eliminar el registro.

Estos desencadenantes tienen asociadas tres Macros de Datos predefinidas:

* + Después de insertar: para el evento de inserción. Podemos aprovecharlo para actualizar el stock de un artículo cuando se realiza una venta.
  + Después de actualizar: para el evento de actualización.
  + Después de eliminar: para el evento de eliminación. Podemos aprovecharlo para llevar un registro de quién y cuándo se ha eliminado un registro.

**7) ¿Cuál es el orden que puede que puede activar un trigger?**

Puede especificar que uno de los desencadenadores AFTER asociados a una tabla sea el primero o el último que se ejecute para cada una de las acciones desencadenadoras INSERT, DELETE y UPDATE. Los desencadenadores AFTER que se activan entre el primero y el último se ejecutan en un orden indefinido.

**8) ¿Cuál es la sintaxis para la creación de FUNCIONES en MySQL?**

CREATE FUNCTION nombre\_función (parametro1,parametro2,...)

RETURNS tipoDato

[atributos de la rutina]

<bloque de instruccciones>

**9) ¿Cuál es la sintaxis para la creación de PROCEDIMIENTOS ALMACENADOS en MySQL?**

CREATE PROCEDURE nombre (parámetros)

[características] definición

**10) Ejemplo REAL del uso de TRIGGERS y PROCEDIMIENTOS ALMACENADOS**

-- TRIGGER PARA INSERT --

DELIMITER //

CREATE TRIGGER PEDIDO\_BI\_TRIGGER

BEFORE INSERT ON PEDIDO

FOR EACH ROW

BEGIN

DECLARE cantidad\_filas INT;

SELECT COUNT(\*)

INTO cantidad\_filas

FROM TOTAL\_VENTAS

WHERE idcliente=NEW.idcliente;

IF cantidad\_filas > 0 THEN

UPDATE TOTAL\_VENTAS

SET total=total+NEW.total

WHERE idcliente=NEW.idcliente;

ELSE

INSERT INTO TOTAL\_VENTAS

(idcliente,total)

VALUES(NEW.idcliente,NEW.total);

END IF;

END//

-- TRIGGER PARA UPDATE

CREATE TRIGGER PEDIDO\_BU\_TRIGGER

BEFORE UPDATE ON PEDIDO

FOR EACH ROW

BEGIN

UPDATE TOTAL\_VENTAS

SET total=total+(NEW.total-OLD.total)

WHERE idcliente=NEW.idcliente;

END//

-- TRIGGER PARA DELETE—

CREATE TRIGGER PEDIDO\_BD\_TRIGGER

BEFORE DELETE ON PEDIDO

FOR EACH ROW

BEGIN

UPDATE TOTAL\_VENTAS

SET total=total-OLD.total

WHERE idcliente=OLD.idcliente;

END//

CREATE PROCEDURE proc\_sacar\_clientes\_tipo (IN tipoCliente INT)

-> BEGIN

-> SELECT \* FROM clientes WHERE tipo = tipoCliente;

-> END

**Fuentes:**

* Procedimientos almacenados (motor de base de datos) (13/03/2017), recuperado de: https://docs.microsoft.com/es-es/sql/relational-databases/stored-procedures/stored-procedures-database-engine?view=sql-server-ver15
* Alejandro Alcalde (Oct 7, 2017), PL/SQL. Disparadores o Triggers, recuperado de: https://elbauldelprogramador.com/plsql-disparadores-o-triggers/
* Triggers en Access. Eventos de tablas (26-Abr-2017), recuperado de: http://minubeinformatica.com/cursos/access-por-modulos/triggers-en-access-disparadores-de-eventos#:~:targetText=Triggers%20en%20Access.,en%20la%20base%20de%20datos.
* Especificar el primer y el último desencadenador (03/03/2017), recuperado de: https://docs.microsoft.com/es-es/sql/relational-databases/triggers/specify-first-and-last-triggers?view=sql-server-ver15
* Ing. Tomas Urbina (s.f), Funciones en MySql, recuperado de: http://appl.transexpress.com.sv/misdocs/dsiw1/Guia%20Funciones%20y%20Triggers.pdf
* Aner Barrena (29 agosto, 2016), MySQL CREATE PROCEDURE: Crear procedimientos almacenados, recuperado de: https://www.anerbarrena.com/mysql-create-procedure-5346/