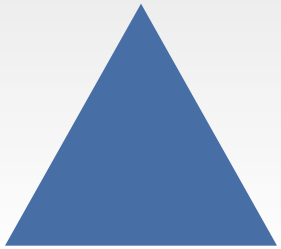
IES Valle del Jerte
PLASENCIA
Bilingual Section

TEMA 9: PROGRAMACIÓN EN BASES DE DATOS (IV)
TRIGGERS



UNION EUROPEA
Fondo Social Europeo
"Una manera de hacer Europa"

"**Una manera de hacer Europa**". Cofinanciación a cargo del Programa Operativo del FSE 2014-2020 para Extremadura gastos de Ciclos Formativos de Grados Medio y Superior.

Índice

1. Introducción.
2. Definición.
3. Utilidad.
4. Sintaxis.
5. Objetos NEW Y OLD.
6. Ejemplos.

INTRODUCCIÓN

- Los TRIGGER se pueden implementarse a partir de la versión 5.0.3 de MySQL.
- También se llaman disparadores, o desencadenadores porque nunca se llaman directamente; se ejecutan de manera automática (“se disparan”) cuando una aplicación o usuario intenta insertar, actualizar, o borrar una fila en una tabla.

DEFINICIÓN

- Son objetos de la base de datos.
- Son procedimientos almacenados.
- Están asociados a una tabla de la BD.
- Se ejecutan automáticamente en respuesta a algún suceso que ocurre en la BD (una instrucción INSERT, UPDATE O DELETE sobre alguna tabla).

UTILIDAD

- Suponen un mecanismo para asegurar la integridad de los datos.
- Se emplean también como un método para realizar operaciones de auditoría sobre la base de datos.
- No hay que abusar de su utilización pues ello puede traer consigo una sobrecarga del sistema y por tanto un bajo rendimiento del mismo.
- No se pueden crear sobre tablas temporales ni sobre vistas, solamente sobre tablas.

VENTAJAS

Las ventajas de los triggers son varias:

1. Se consigue mantener la integridad de los datos, al crear de forma automática restricciones, que hacen que los usuarios introduzcan sólo valores válidos.

2. Reducen las tareas de mantenimiento de la aplicación; los cambios realizados a un disparador se reflejan automáticamente en todas las aplicaciones que tienen que ver con la BD.

TRIGGER EN EL DICCIONARIO DE DATOS

- La vista TRIGGERS del diccionario de datos contendrá toda la información sobre cada trigger. Hay una fila en esta vista por cada trigger en el servidor.

select * from information_schema.triggers;

TRIGGER_CATALOG	TRIGGER_SCHEMA	TRIGGER_NAME	EVENT_MANIPULATION
def	sys	sys_config_insert_set_user	INSERT
def	sys	sys_config_update_set_user	UPDATE
def	notas	t1	DELETE
def	notas	EJERCICIO5	DELETE
def	notas	EJERCICIO6	UPDATE

SINTAXIS (COMPLETA)

```
CREATE
    [DEFINER = { user | CURRENT_USER }]
    TRIGGER trigger_name
    trigger_time trigger_event
    ON tbl_name FOR EACH ROW
    [trigger_order]
    trigger_body

trigger_time: { BEFORE | AFTER }

trigger_event: { INSERT | UPDATE | DELETE }

trigger_order: { FOLLOWS | PRECEDES } other_trigger_name
```

SINTAXIS (ABREVIADA)

```
CREATE TRIGGER nombre_trigger
{BEFORE | AFTER}
{UPDATE | INSERT | DELETE}
ON tabla
FOR EACH ROW
    cuerpo_del_trigger
```

BEFORE|AFTER indica cuando se ejecutará el trigger, antes (before) o después (after) de la instrucción DML.

UPDATE | INSERT | DELETE: define la operación asociada al trigger.

ON tabla : Define la tabla asociada al trigger.

FOR EACH ROW: indica que el trigger se ejecutará por cada fila

cuerpo_del_trigger: instrucciones del procedimiento.

TRIGGERS BEFORE Y AFTER

- Indican si el Trigger se ejecuta BEFORE (antes) o AFTER (después) del evento DML.
- Podemos tener los siguientes tipos de triggers por tabla:

BEFORE INSERT, AFTER INSERT .

BEFORE UPDATE,AFTER UPDATE.

BEFORE DELETE, AFTER DELETE.

TRIGGERS BEFORE Y AFTER

- Normalmente se usa el tipo BEFORE para chequear que los datos son correctos antes de que sean incorporados a la tabla.
- El tipo AFTER se suele utilizar para columnas calculadas o para realizar o registrar las operaciones una vez que estas son realizadas en la tabla. (uso de auditorías)

REFERENCIAS A COLUMNAS AFECTADAS POR LA SENTENCIA DML: OBJETOS NEW Y OLD EN TRIGGERS

Cuando se necesita relacionar el trigger con columnas específicas de una tabla debemos usar los identificadores **OLD y NEW.**

REFERENCIAS A COLUMNAS AFECTADAS: OBJETOS NEW Y OLD EN TRIGGERS

- **OLD** indica el valor antiguo de la columna.
- **NEW** el valor nuevo que pudiese tomar.
- Ejemlplo de uso de OLD Y NEW en un trigger.

En un trigger de tipo **BEFORE UPDATE** que afectase a una columna "**id_producto**":

- se utilizará la expresión OLD.id_producto para conocer el valor de la columna antes de ser modificado y "NEW.id_producto" será el nuevo valor de la columna después de la modificación.

REFERENCIAS A COLUMNAS AFECTADAS: OBJETOS NEW Y OLD EN TRIGGERS

- Las referencias al objeto NEW.nombre_columna se usan en los trigger sobre el evento INSERT.
- Las referencias al objeto OLD.nombre_columna se usan en los trigger sobre el evento DELETE.
- Se referencian ambos objetos NEW.nombre_columna y OLD.nombre columna en los trigger sobre el evento UPDATE, ya que en esta operación existirá el valor antes de la modificación (old) y el valor después de la modificación (new).

BORRAR TRIGGERS

Para eliminar un trigger se utiliza el comando DROP.

DROP TRIGGER nombre_trigger

Para eliminar un trigger comprobando su existencia en la BD, se utiliza la estructura DROP.. IF EXISTS.

DROP TRIGGER IF EXISTS nombre_trigger

UTILIZACIÓN DE TRIGGERS

Los trigger se usan para mantener la integridad de la base de datos. Podemos tener estos tipos de triggers:

- Triggers para mantener automáticamente réplicas o copias de datos.
- Triggers para crear tablas de auditorías de operaciones realizadas en la BD.
- Triggers para validar los datos que se van a almacenar en una tabla y así mantener la consistencia de los mismos.
- Triggers para calcular atributos derivados.

EJEMPLOS DE UTILIZACIÓN DE TRIGGERS

SEGUIMOS TRABAJANDO CON LA BASE DE DATOS BDTEMA9.

```
CREATE TABLE REPLICA_ALUMNOS
(ID INT PRIMARY KEY,
ALUMNO VARCHAR(50));
```

EJEMPLOS DE UTILIZACIÓN DE TRIGGERS-EJEMPLO1

TRIGGER PARA REALIZAR COPIAS O RÉPLICAS DE DATOS:

Sirven para mantener sincronizada una copia de seguridad de unos datos.

- En el ejemplo siguiente, seguimos utilizando la BD BDTEMA9.
- Creamos una tabla alumnos.

```
CREATE TABLE ALUMNOS
(ID INT PRIMARY KEY,
NOMBRE VARCHAR(50));
```

- Creamos una tabla réplica de la tabla alumnos

```
CREATE TABLE REPLICA_ALUMNOS
(ID INT PRIMARY KEY,
NOMBRE VARCHAR(50));
```

EJEMPLOS DE UTILIZACIÓN DE TRIGGERS-EJEMPLO1

Crear un trigger que cada vez que se inserte un registro en la tabla ALUMNOS, inserte también en la tabla REPLICA_ALUMNOS.

```
DELIMITER //
DROP TRIGGER IF EXISTS TRIGGER1//
CREATE TRIGGER TRIGGER1
    BEFORE INSERT ON ALUMNOS
    FOR EACH ROW
BEGIN
    INSERT INTO REPLICA_ALUMNOS VALUES (NEW.ID,NEW.NOMBRE);
END //
```

EJEMPLOS DE UTILIZACIÓN DE TRIGGERS-EJEMPLO1

Ejecutamos INSERT en alumnos y comprobamos el resultado en las dos tablas:

```
INSERT INTO ALUMNOS VALUES (1,'ALUMNO1'),(2,'ALUMNO2'),(3,'ALUMNO3');
```

```
40 •  SELECT * FROM ALUMNOS;
```

Result Grid | Filter Rows: | Edit:

ID	NOMBRE
1	ALUMNO1
2	ALUMNO2
3	ALUMNO3
NULL	NULL

```
41 •  SELECT * FROM REPLICA ALUMNOS;
```

Result Grid | Filter Rows: | Edit: | Export/Imp

ID	NOMBRE
1	ALUMNO1
2	ALUMNO2
3	ALUMNO3
NULL	NULL

EJEMPLOS DE UTILIZACIÓN DE TRIGGERS-EJEMPLO1

De forma automática, al realizar el INSERT en la tabla ALUMOS, el trigger se dispara y realiza la inserción en la tabla REPLICA__ALUMNOS.

Insertamos otro registro más y comprobamos que ambas tablas siguen sincronizadas:

```
INSERT INTO ALUMNOS VALUES (10,'ALUMNO10')
```

	ID	NOMBRE
▶	1	ALUMNO1
	2	ALUMNO2
	3	ALUMNO3
	10	ALUMNO10

ID	NOMBRE
1	ALUMNO1
2	ALUMNO2
3	ALUMNO3
10	ALUMNO10
NULL	NULL

EJEMPLOS DE UTILIZACIÓN DE TRIGGERS-EJEMPLO2

TRIGGERS DE AUDITORÍA:

- Se utilizan para auditar las operaciones que se realizan sobre una tabla, manteniendo una tabla auxiliar con los cambios que registra la tabla afectada.
- Es una manera fácil y automática de registrar qué cambios se dan en una tabla.
- Tienen utilidad para almacenar datos de borrados, modificaciones, inserciones, fechas y autores de los cambios.

EJEMPLOS DE UTILIZACIÓN DE TRIGGERS-EJEMPLO2

- En el ejemplo siguiente, vamos a crear una tabla CAMBIOS donde guardaremos los cambios realizados en las tablas.

```
/* CREAR TABLA CAMBIOS*/
CREATE TABLE CAMBIOS(MENSAJE VARCHAR(200));
```

- El trigger se dispara DESPUÉS de cada modificación en la tabla ALUMNOS.

EJEMPLOS DE UTILIZACIÓN DE TRIGGERS-EJEMPLO2

- En el ejemplo siguiente, vamos a crear una tabla CAMBIOS donde guardaremos los cambios realizados en las tablas.

```
/* CREAR TABLA CAMBIOS*/
CREATE TABLE CAMBIOS(MENSAJE VARCHAR(200));
```

- La tabla tiene un único campo varchar donde vamos a guardar concatenada la información que nos interesa, en este caso: el usuario que ha modificado el registro, la fecha, los valores de los datos antes de la modificación y los valores de los datos después de la modificación.

EJEMPLOS DE UTILIZACIÓN DE TRIGGERS-EJEMPLO2

```
DELIMITER //
DROP TRIGGER IF EXISTS TRIGGER2//
CREATE TRIGGER TRIGEER2
AFTER UPDATE ON ALUMNOS
FOR EACH ROW
BEGIN
INSERT INTO CAMBIOS
VALUES (CONCAT_WS('  ','MODIFICACION REALIZADA POR:',USER(),'FECHA:', current_Date(),
'VALORES ANTIGUOS' , OLD.ID, OLD.NOMBRE,
'VALORES NUEVOS:',NEW.ID,NEW.NOMBRE));
END //
```

EJEMPLOS DE UTILIZACIÓN DE TRIGGERS-EJEMPLO2

De forma automática, al realizar el UPDATE en la tabla ALUMOS, el trigger se dispara y realiza la inserción en la tabla CAMBIOS.

```
UPDATE ALUMNOS SET ALUMNO='MARIA MONTERO' WHERE ID=10;
```

ID	NOMBRE
1	ALUMNO1
2	ALUMNO2
3	ALUMNO3
10	MARIA MONTERO

```
64    SELECT * FROM CAMBIOS;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

MENSAJE
MODIFICACION REALIZADA POR: root@localhost FECHA: 2020-05-03 VALORES ANTIGUOS 10 ALUMNO10 VALORES NUEVOS: 10 MARIA MONTERO

EJEMPLOS DE UTILIZACIÓN DE TRIGGERS-EJEMPLO3

VALIDACIÓN DE DATOS:

- Los triggers se pueden utilizar para validar los datos que se van a almacenar en una tabla y así mantener la consistencia de los mismos.
- Son parecidos a los constraint CHECK, pero pueden afectar a campos que están en tablas diferentes.
- Debemos tener en cuenta que si el trigger falla y está asociado a una instrucción DML también fallará la operación DML que intenta modificar la BD.
- Se utiliza este mecanismo para evitar entradas no deseadas en las tablas.

EJEMPLOS DE UTILIZACIÓN DE TRIGGERS-EJEMPLO3

VALIDACIÓN DE DATOS:

- Los triggers se pueden utilizar para validar los datos que se van a almacenar en una tabla y así mantener la consistencia de los mismos.
- Son parecidos a los constraint CHECK, pero pueden afectar a campos que están en tablas diferentes.
- Debemos tener en cuenta que si el trigger falla y está asociado a una instrucción DML también fallará la operación DML que intenta modificar la BD.
- Se utiliza este mecanismo para evitar entradas no deseadas en las tablas.

EJEMPLOS DE UTILIZACIÓN DE TRIGGERS-EJEMPLO3

En este ejemplo, vamos a crear un trigger para impedir que al insertar un nuevo alumno, se pueda introducir un valor <=0.

- Normalmente un TRIGGER en MySQL no devuelve datos de ningún tipo, ni siquiera mensajes. El objeto de SIGNAL es administrar las excepciones de usuario, pero no emitir mensajes.
- El uso de SIGNAL es válido a partir de MySQL 5.

EJEMPLOS DE UTILIZACIÓN DE TRIGGERS-EJEMPLO3

- Crear un trigger para impedir que al insertar un nuevo alumno, se pueda introducir un valor <=0.
- Normalmente un TRIGGER en MySQL no devuelve datos de ningún tipo. Ni siquiera mensajes. El objeto de SIGNAL es administrar las excepciones de usuario,

```
DELIMITER //
DROP TRIGGER IF EXISTS TRIGGER3//
CREATE TRIGGER trigger3
BEFORE INSERT ON ALUMNOS
FOR EACH ROW
BEGIN
DECLARE ERR_IDENTIFICADOR_NO_VALIDO CONDITION FOR SQLSTATE '45000';
IF NEW.id <=0 THEN
SIGNAL ERR_IDENTIFICADOR_NO_VALIDO
SET MESSAGE_TEXT = 'El identificador del alumno no puede ser menor o igual que cero';
END IF;
END //
```

EJEMPLOS DE UTILIZACIÓN DE TRIGGERS-EJEMPLO3

- Comprobamos que al insertar un valor válido, no hay problemas.

```
INSERT INTO ALUMNOS VALUES(5,'ALUMNO5');
```

ID	NOMBRE
1	ALUMNO1
2	ALUMNO2
3	ALUMNO3
5	ALUMNO5
10	MARIA MONTERO
NULL	NULL

EJEMPLOS DE UTILIZACIÓN DE TRIGGERS-EJEMPLO3

- Pero al intentar insertar con un valor no válido, el trigger impide la inserción de los datos.

```
INSERT INTO ALUMNOS VALUES(0,'ALUMNOPRUEBA');
```

- Retorna este error:

```
Error Code: 1644. El identificador del alumno no puede ser menor o igual que cero
```

- Y el insert no se ejecuta:

ID	NOMBRE
1	ALUMNO1
2	ALUMNO2
3	ALUMNO3
5	ALUMNO5
10	MARIA MONTERO

EJEMPLOS DE UTILIZACIÓN DE TRIGGERS-EJEMPLO4

ACTUALIZAR CAMPOS CALCULADOS:

En este ejemplo, vamos a crear un trigger para calcular el campo edad a partir de la fecha de nacimiento cuando se inserta un nuevo alumno.

- En primer lugar, modificaremos la tabla ALUMNOS para añadir el campo FECHA_NAC y EDAD

```
ALTER TABLE ALUMNOS
ADD FECHA_NAC DATE,
ADD EDAD INT;
```

EJEMPLOS DE UTILIZACIÓN DE TRIGGERS-EJEMPLO4

```
DELIMITER //
DROP TRIGGER IF EXISTS TRIGGER4//
CREATE TRIGGER TRIGGER4 BEFORE INSERT ON ALUMNOS
FOR EACH ROW
BEGIN
    IF NEW.fecha_nac IS NOT NULL THEN
        SET NEW.edad = YEAR(CURRENT_DATE()) - YEAR(NEW.fecha_nac);
    END IF;
END;//
```

EJEMPLOS DE UTILIZACIÓN DE TRIGGERS-EJEMPLO4

- Insertamos una nueva fila en la tabla ALUMNOS.

```
INSERT INTO ALUMNOS (ID,NOMBRE,FECHA_NAC)
VALUES (20,'ALUMNO20','2000-03-01')
```

- Comprobamos que después de la ejecución el campo edad se ha calculado automáticamente al disparar el trigger

ID	NOMBRE	FECHA_NAC	EDAD
1	ALUMNO1	NULL	NULL
2	ALUMNO2	NULL	NULL
3	ALUMNO3	NULL	NULL
5	ALUMNO5	NULL	NULL
10	MARIA MONTERO	NULL	NULL
20	ALUMNO20	2000-03-01	20
NULL	NULL	NULL	NULL