**TRIGGERS**

Un trigger o disparador es un tipo especial de rutina almacenada que se activa o ejecuta cuando en una tabla ocurre un evento de tipo INSERT, DELETE o UPDATE. Es decir, los disparadores implementan una funcionalidad asociada a cualquier cambio en una tabla.

**GESTIÓN DE DISPARADORES**

Las instrucciones para gestionar disparadores son CREATE TRIGGER, SHOW TRIGGER y DROP TRIGGER.

**Crear triggers:**

CREATE TRIGGER nombre\_disp momento\_disp evento\_disp

ON nombre\_tabla FOR EACH ROW sentencia\_disp

*Momento disp:* Es el momento en el que el disparador entra en acción. Puede ser BEFORE o AFTER, para indicar que el disparador se ejecute antes o después que la sentencia que lo activa.

*Evento disp*: Indica la clase de sentencia que activa el disparador. Puede ser INSERT, UPDATE o DELETE. Por ejemplo, un disparador BEFORE para sentencias INSERT podría utilizarse para validar los valores a insertar.

**No puede haber dos disparadores en una misma tabla que correspondan al mismo momento y sentencia.** Por ejemplo, no puede habar dos disparadores BEFORE UPDATE. Pero sí es posible tener los disparadores BEFORE UPDATE y BEFORE INSERT o AFTER UPDATE y AFTER INSERT.

*FOR EACH ROW*: Hace referencia a las acciones a llevar a cabo sobre cada fila de la tabla indicada.

*Sentencia disp:* Es la sentencia que se ejecuta cuando se activa el disparador. Si se desean ejecutar múltiples sentencias, deben colocarse entre BEGIN…END.

**Las columnas de la tabla asociada con el disparador pueden referenciarse empleando los alias OLD y NEW.**

OLD.nombre\_col hace referencia a una columna de una fila existente, antes de ser actualizada o borrada.

NEW.nombre\_col hace referencia a una columna en una nueva fila a punto de ser insertada o en una fila existente después de ser actualizada.

**OLD y NEW permiten acceder a columnas en los registros afectados por un disparador. En un disparador para INSERT, solo puede usarse NEW.nombre\_col ya que no hay una versión anterior del registro. En un disparador para DELETE solo puede emplearse OLD.nombre\_col, porque no hay un nuevo registro. En un disparador para UPDATE se puede emplear OLD.nombre\_col para referirse a las columnas de un registro antes de que sea actualizado y NEW.nombre\_col para referirse a las columnas del registro luego de actualizarlo.**

**Borrar triggers:**

Se usa DROP TRIGGER. El nombre del disparador debe incluir el nombre de la tabla.

DROP TRIGGER [IF EXISTS] [nombre\_esquema].trigger\_name

**Consulta de triggers:**

Podemos obtener información de los triggers creados con SHOW TRIGGERS

SHOW TRIGGERS FROM nombre\_esquema

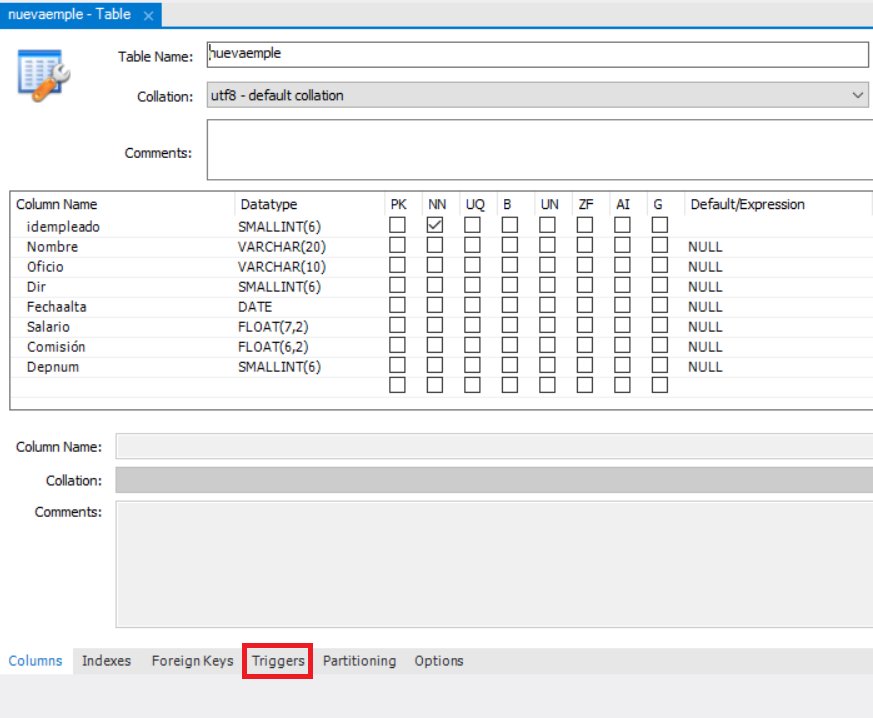
**USOS DE DISPARADORES**

**Control de sesiones**

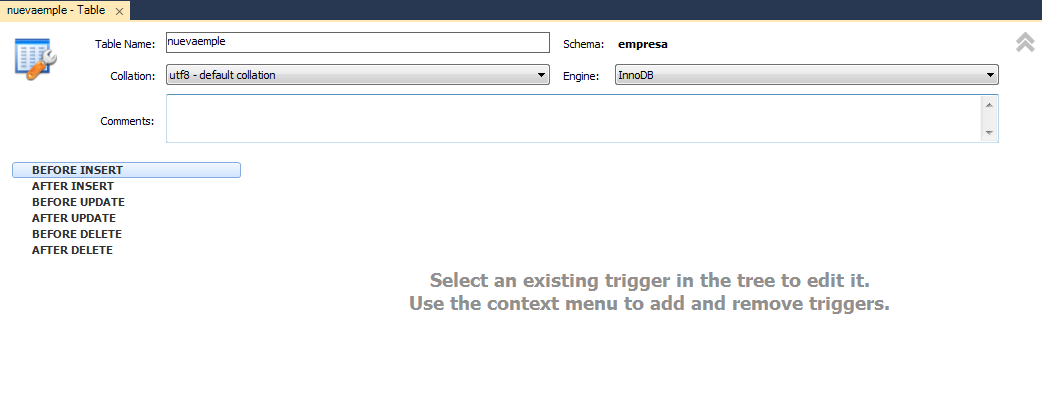
En ocasiones puede ser interesante recoger ciertos valores que nos permitan ver un resumen de lo realizado en dicha sesión.

**EJEMPLO1**

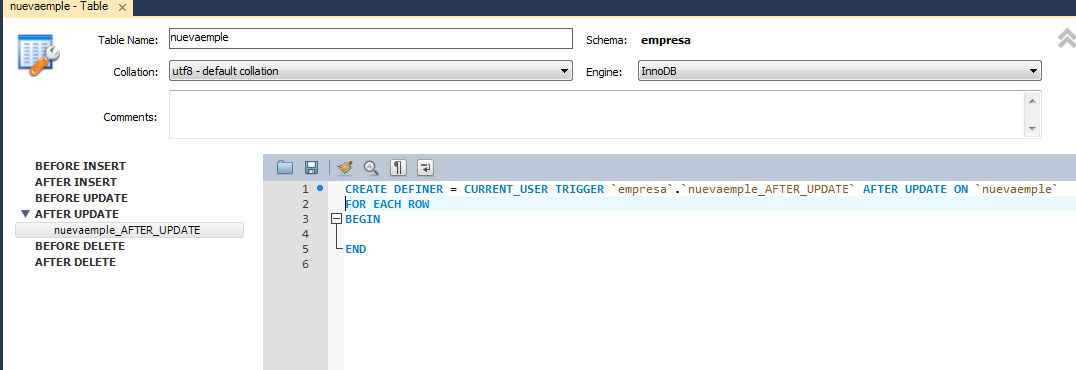
Vamos a crear un trigger llamado SUBIDA\_SAL, que se disparará después de cada modificación de la columna salario de la tabla nuevaemple del esquema empresa. Este trigger tendremos que crearlo dentro de la tabla que va a producir la ejecución del mismo. Para ello, vamos a crear una nueva tabla dentro de empresa que se llamará AUDITAREMPLE y que va a contener un único campo de tipo INT(11). Una vez creada, nos colocamos sobre la tabla nuevaemple de empresa, pulso el botón derecho y selecciono la opción Triggers que aparece a la derecha de Foreign Keys.



Me aparecerá la siguiente pantalla:



Tengo que seleccionar AFTER UPDATE y pulso el +. Me aparece lo siguiente:



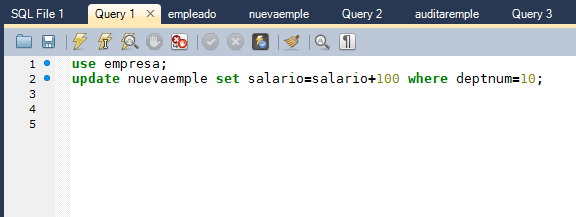
Dentro del BEGIN-END tengo que escribir:

INSERT INTO AUDITAREMPLE VALUES (old.idempleado);

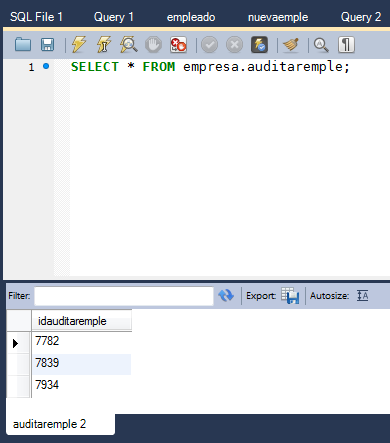
Cuando actualice mediante un UPDATE la tabla nuevaemple, se va a guardar en la tabla auditaremple los números de los empleados a los que se ha aplicado la actualización.

Por ejemplo, si ejecuto la sentencia siguiente, dentro del esquema empresa:

UPDATE NUEVAEMPLE SET SALARIO=SALARIO+100 WHERE DEPNUM=10;



Si muestro ahora el contenido de la tabla auditaremple, aparecerán los tres números de los empleados a los que se les ha actualizado el salario:



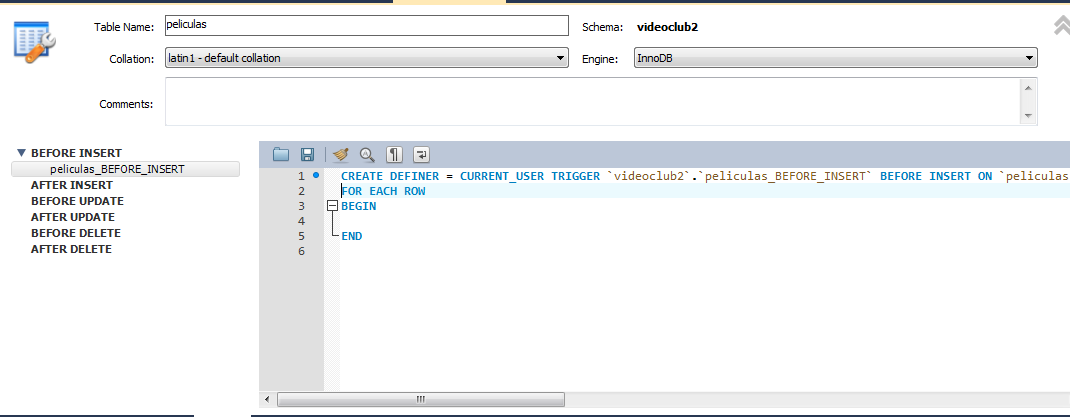
**Control de valores de entrada:**

Un uso posible de los disparadores es el control de valores insertados o actualizados en tablas.

**EJEMPLO2**

En el siguiente ejemplo, se crea un trigger para INSERT que verifica los valores de entrada para la tabla PELÍCULAS del videoclub y comprueba que el valor del precio de alquiler es mayor que 0. Si no es así, pondrá un 2. Esto debe hacerse en un disparador BEFORE porque los valores deben verificarse antes de insertar el registro.

Haré lo que se describe en la siguiente pantalla:



Dentro del BEGIN-END escribiré:

IF new.precioalquiler<=0 THEN

SET new.precioalquiler=2;

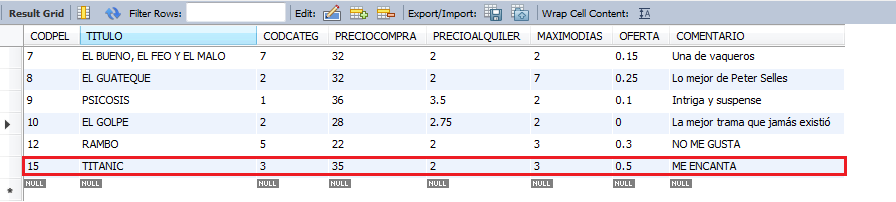
END IF;

A continuación, vamos a introducir la siguiente sentencia:

USE VIDEOCLUB;

INSERT INTO nuevapelis VALUES (15,"TITANIC",3,35,0,3,0.5,"ME ENCANTA");

Observamos que hemos introducido un precio de alquiler igual a 0. Vamos a ver lo que se ha guardado en la tabla películas y comprobamos que en lugar del 0 ha guardado un 2.



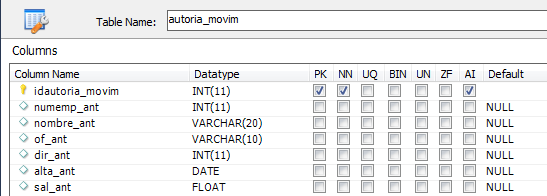
**Registro y auditoría**

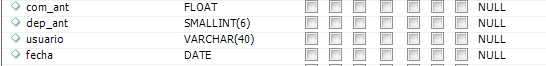
Cuando muchos usuarios acceden a las bases de datos puede ser que el registro de log no sea suficiente o simplemente dificulte la revisión de la actividad en el servidor en el sentido de saber quién ha hecho qué operación y a qué hora. Podemos usar triggers que me faciliten esta tarea. Podemos asignar un trigger a una tabla que se dispare después de una sentencia DELETE o UPDATE, que guarde los valores del registro, así como alguna otra información de utilidad en una tabla de log.

**EJEMPLO 3**

Queremos saber quién y a qué hora modificó la tabla nuevaemple del esquema empresa. Para ello creamos un trigger que registre dichas actualizaciones incluyendo los datos antiguos para cada registro modificado.

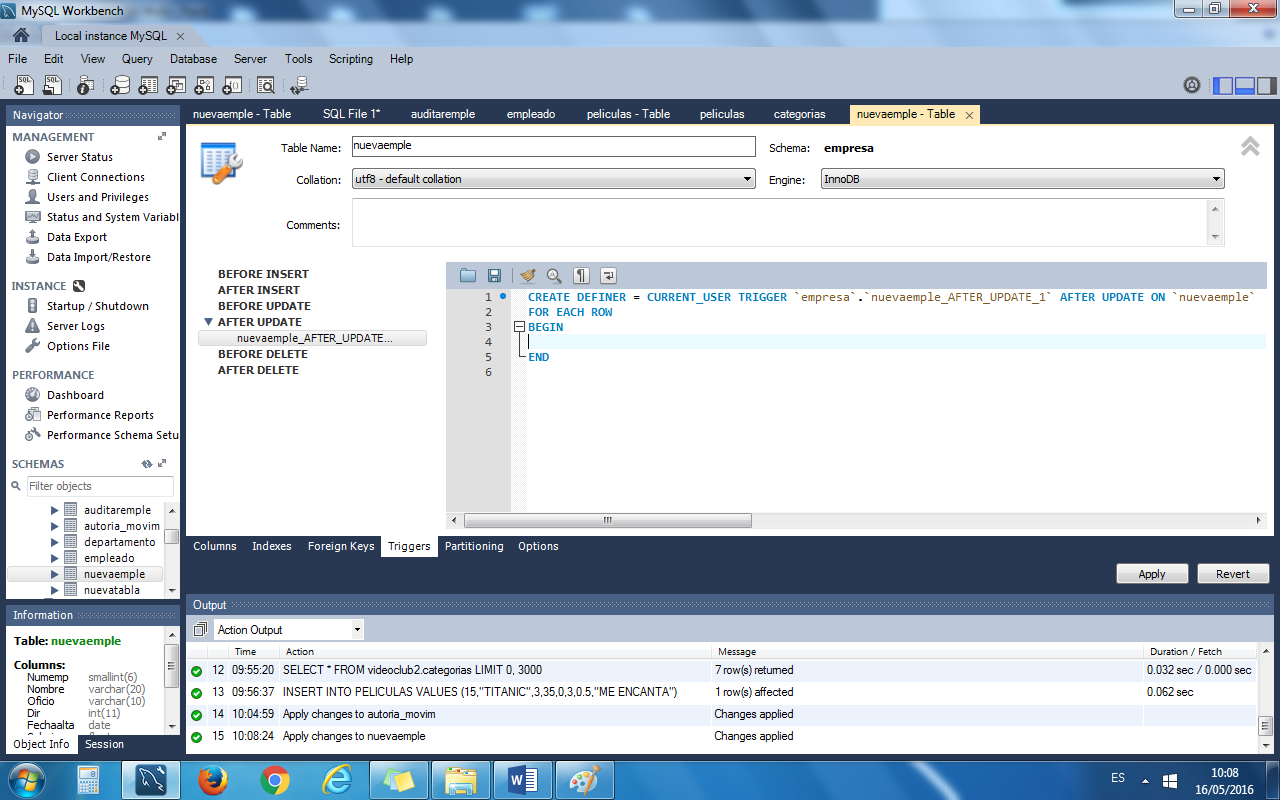
Lo primero que vamos a hacer es crear una tabla de auditoría:

****

****

**El campo fecha que sea DATETIME**

Y ahora crearemos un trigger para que vaya llenando los registros de esta tabla cada vez que alguien ejecute una actualización sobre la tabla nuevaemple.



INSERT INTO auditoria (numemp\_ant,nombre\_ant,of\_ant,dir\_ant,alta\_ant,sal\_ant,com\_ant,dep\_ant,usuario,fecha) VALUES (old.idempleado,old.nombre,old.oficio,old.dir,old.fechaalta,old.salario,old.comisión,old.depnum, current\_user(), now());

Las funciones CURRENT\_USER() y NOW() devuelven el usuario y la fecha y hora actuales.

A continuación, ejecuto la siguiente instrucción:

UPDATE NUEVAEMPLE SET SALARIO=SALARIO+500 WHERE DEPNUM=20

Se actualizarán los salarios correspondientes a los empleados de la tabla nuevaemple que pertenezcan al departamento 20, incrementándolos en 500 €.

Además, el trigger que hemos creado anteriormente se activará con la ejecución de la actualización (UPDATE) y agregará varios registros a la tabla autoria\_movim cada vez que se actualice la tabla nuevaemple.

