**Clase 17**

**Triggers**

*CONCEPTO GENERAL*

Definimos como Trigger a un conjunto de sentencias o programa almacenado en el servidor (*de DB*) creado para ejecutarse (*dispararse*) de forma automática, cuando uno o más eventos de DML específicos ocurren en la DB.

El Trigger se despierta y ejecuta sus sentencias en el momento en que una operación de DML (INSERT, UPDATE y DELETE) asociada al disparador aparece.

*USOS ESPECÍFICOS*

Los Triggers nacieron integrados a las bases de datos para funcionar de la misma forma en la cual muchos lenguajes de programación activan la detección de eventos específicos cuando el programa se ejecuta.

Pero un Trigger no funciona de forma mágica. Nosotros somos quienes debemos definirlo y escribir su lógica.

La funcionalidad principal que más se les da a los Triggers en el ecosistema de bases de datos, es activarlos para alimentar las tablas de auditoría.

Estas tablas funcionan como complemento dentro de una bb.dd., recopilando información adicional que no es importante dentro de las tablas principales.

La funcionalidad principal que más se les da a los Triggers en el ecosistema de bases de datos, es activarlos para alimentar las tablas de auditoría.

Estas tablas funcionan como complemento dentro de una bb.dd., recopilando información adicional que no es importante dentro de las tablas principales.

Tablas de Auditoría, Log, Bitácora, son algunos nombres con los cuales se definen a estas tablas secundarias, que se ocupan de almacenar información no importante para el negocio en sí, pero clave para el departamento de IT y/o de Seguridad Informática.

Por ejemplo una tabla Productos, almacena información del mismo como ser: *código*, *descripción*, *fecha de alta* o fabricación, *precio de costo* y *precio de venta*, entre otros.

Podemos registrar en una tabla de auditoría paralela, quién lo creó, fecha de creación, quién modificó su precio de venta o de costo, cuándo, y quién lo eliminó de la lista de Productos.

Todo este registro o bitácora de cambios, lo podemos realizar activando uno o más Triggers que monitoreen todos estos pasos y cambios sobre uno o más registros.

Las tablas de LOGs o Bitácoras no suelen tener relación alguna con las entidades que monitorean y, muchas veces, almacenan información general de diferentes entidades.

***TIPOS DE TRIGGER***

El uso de Trigger se puede establecer en dos momentos diferentes de cuando se realiza una operación del tipo UPDATE, DELETE, o INSERT.

Ese momento puede ser antes (BEFORE) de que ocurra la operación, o después (AFTER) de que ocurra la misma.

*BEFORE*

Cuando el usuario envía una operación del tipo INSERT, UPDATE o DELETE sobre una tabla, y esta tiene activo el Trigger que detecta la operación, se disparará la acción BEFORE, la cual permitirá por ejemplo registrar en una tabla de auditoría que se realizará la operación xx sobre la tabla yy.

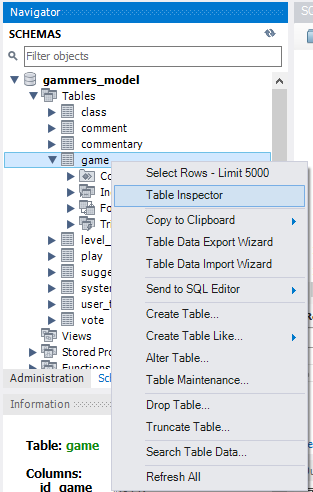
*AFTER*

Cuando el usuario envía una operación del tipo INSERT, UPDATE o DELETE sobre una tabla, y esta tiene activo el Trigger que detecta la operación, se disparará la acción AFTER, la cual registrará en una tabla de auditoría que se realizará la operación xx sobre la tabla yy.

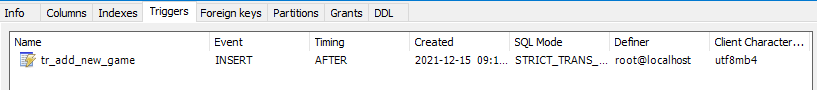
*IDENTIFICAR TRIGGERS EN UNA TABLA*

*AFTER*

Si deseamos ver el o los Trigger(s) asociado(s) a una tabla, desde el SGBD pulsamos sobre la tabla con el botón secundario del mouse, y seleccionamos del menú contextual la opción Table Inspector.

**

Se abre la pestaña de inspección, donde debemos pulsar sobre la opción Triggers. Allí veremos listado cada uno de los Triggers creados sobre esta tabla, el tipo de evento y el momento en el cual se activa, su fecha de creación, usuario, y demás información.

**

*SINTAXIS*

Antes de crear un trigger, debemos tener previamente definidas:

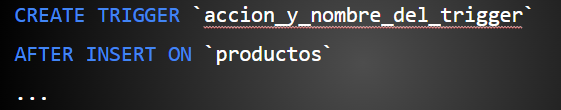
* La tabla a la cual le asociaremos el trigger
* La tabla donde se realizarán operaciones relacionadas al Trigger

Resuelto estos dos puntos, queda definir si el Trigger se ejecutará antes o después de la acción a evaluar. Esto impactará en su sintaxis.

En una ventana de script, utilizaremos la sentencia CREATE TRIGGER, seguida al “*nombre amigable*” que nos permitirá identificar su función.

Seguido a dicha sentencia, debemos agregar el comando que indica si se ejecuta antes o después de la acción a evaluar, el tipo de acción que controlará, y sobre qué tabla trabajará.

En el siguiente ejemplo, utilizamos la sentencia (CREATE TRIGGER) para definir una acción posterior al alta (AFTER INSERT) de un nuevo producto en la tabla homónima (ON productos).



Nos quedan dos incógnitas a resolver para finalizar el armado del Trigger (*manipular la inserción de múltiples registros e identificar los nuevos registros*).

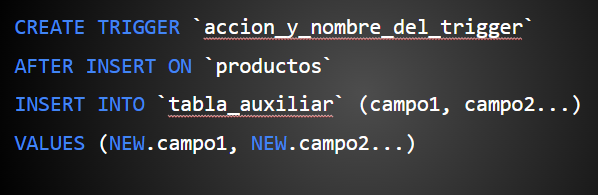
Lo primero lo resolveremos con la sentencia FOR EACH ROW, mientras que, lo segundo, con el comando NEW.

*NEW*

La palabra reservada NEW se ocupará de detectar o ubicar a cada nuevo registro agregado. No la utilizamos para un registro (*o fila*) completo en sí, sino que debemos integrar con cada uno de los datos que conforma un nuevo registro.

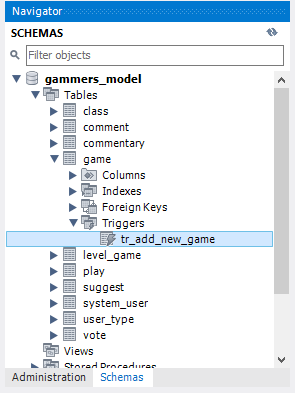
De esta forma garantizamos que se llevará una bitácora de cada uno de los nuevos registros que se inserten en la tabla principal.

Ejemplo de cómo integrar la sentencia NEW en combinación con FOR EACH ROW.



*TRIGGERS ASOCIADOS*

Cada tabla podrá contener más de un trigger asociado. Si deseamos visualizar los mismos, debemos ubicar en el SGBD, el apartado Tables, luego la tabla a la cual le asociamos los triggers, y finalmente el apartado Triggers.



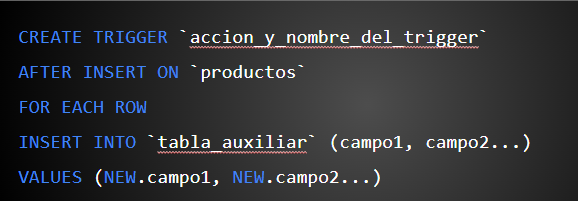
*FOR EACH ROW*

*RECORRER MÚLTIPLES FILAS INSERTADAS*

*FOR EACH ROW*

Un Trigger es un evento que se dispara una vez por cada sentencia INSERT que se realice sobre la tabla asociada. Pero, como mencionamos anteriormente, para aquellos casos donde se inserten registros de forma masiva, necesitamos incluir la cláusula FOR EACH ROW.

De esta forma nos aseguramos que el Trigger se disparará una vez por cada nuevo registro agregado de forma masiva.



*TRIGGERS POR TABLA*

Cada tabla que requiera tener un trigger, podrá asociarse a la misma un Trigger por cada acción que necesitemos cubrir.

Si desde una tabla debemos alimentar dos o más tablas del tipo bitácora, entonces debemos crear un Trigger para cada una de ellas, de acuerdo a la acción correspondiente que debamos ejecutar.

*INTEGRACIÓN DE FUNCIONES*

*FUNCIONES DEL SISTEMA*

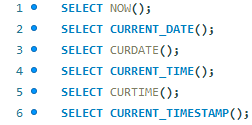
En casi todos los casos donde los Trigger se usan para registrar un LOG o Bitácora de cambios sobre diferentes tablas, debemos integrar datos adicionales a estas últimas, como ser el usuario que realiza el cambio, el ambiente de base de datos donde esto ocurre y/o la fecha y hora.

Y para realizar esto de forma efectiva, Mysql cuenta con una serie de funciones las cuales nos facilitan el trabajo. Podemos dividir a las mismas en tres categorías diferentes:

* de Fecha y Hora
* de Usuario
* de Plataforma

*FUNCIONES DE FECHA Y HORA*

Algunas de las funciones de fecha y hora disponibles en Mysql, son:

* NOW()
* CURRENT\_DATE()
* CURDATE()
* CURRENT\_TIME()
* CURTIME()
* CURRENT\_TIMESTAMP()

*FUNCIONES DE USUARIO*

Algunas de las funciones de usuario, disponibles en Mysql, son:

* SESSION\_USER()
* SYSTEM\_USER()
* USER()



*FUNCIONES DE PLATAFORMA*

Algunas de las funciones de Plataforma, disponibles en Mysql, son:

* DATABASE()
* VERSION()



*TRIGGERS POR TABLA*

Las funciones de Plataforma son muy útiles para cuando trabajamos con sistemas distribuidos en varios servidores y luego se concentran en una base de datos general.

Y las funciones de Usuario son sumamente útiles cuando trabajamos con bases de datos correctamente estructuradas a nivel de seguridad, estableciendo diferentes usuarios y permisos para cada objeto que la integra.

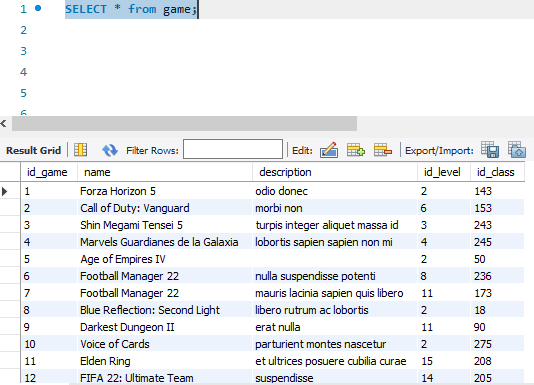
*IMPLEMENTAR TRIGGER*

Crearemos una tabla simple y una tabla similar que sea un espejo de la primera. Luego, le agregaremos un trigger que detecte la inserción de un registro en la tabla simple, y replique el mismo en la tabla espejo.

El trigger se disparará justo antes de que se inserte el registro en la tabla simple.

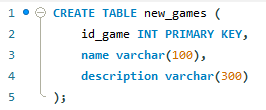
*TABLA GAME*

Trabajaremos con la tabla game. Detectaremos la inserción de nuevos registros, creando un LOG de los nuevos juegos ingresados dentro del sistema que registre el id\_game, name y description



*TABLA NEW\_GAMES*

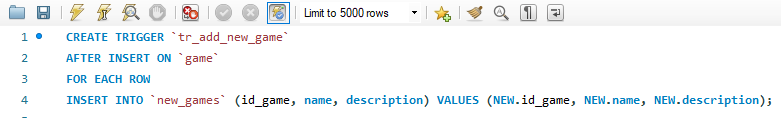
Creamos el script para la tabla que usaremos de new\_games. Los únicos campos que registrará desde la tabla game, serán (id\_game, name y description).

**

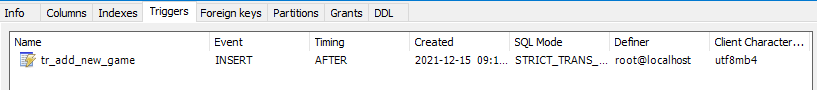
*SCRIPT DEL TRIGGER*

Armamos las bases del SCRIPT de generación del trigger, donde contemplamos la inserción simultánea de múltiples registros.

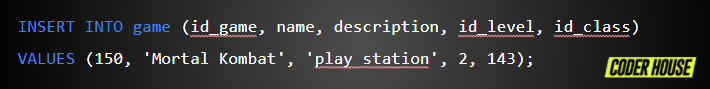
Definamos, a continuación, los campos.



Verifiquemos si el Trigger fue creado exitosamente.



Luego, agreguemos un nuevo registro a la tabla game para verificar su correcto funcionamiento:



Finalmente verás que el Trigger almacenó la operación realizada, copiando los campos indicados en esta tabla de new\_games.