BD UD6 Triggers



Introducción

Un trigger viene a ser como un procedimiento almacenado que se ejecuta automáticamente cuando sobre una tabla se realiza alguna operación que implique modificar sus datos (DELETE, INSERT, UPDATE).

Por lo tanto un trigger va a estar 'asociado' a una tabla y a un 'tipo de operación' sobre la tabla.

El trigger no tiene que realizar la operación sobre la tabla. La operación sobre la tabla viene a través de una orden SQL (UPDATE, DELETE, INSERT) que se ejecuta. Después o antes de que se ejecute esa orden, **por cada fila modificada** de la tabla se va a ejecutar el conjunto de instrucciones del trigger.

Los usos que le podemos dar a un trigger son:

* Monitorizar y registrar operaciones sobre las tablas (podemos guardar información sobre quien y cuando se realizó alguna modificación).
* Verificar que los datos sean correctos antes de añadirlos o usarlos para una modificación.
* Un mecanismo para implementar columnas calculadas (por ejemplo, la tabla ATRACCIONES tiene una columna 'ganancias' y disponemos de la tabla ATRACCION\_DIA, donde guardamos las ganancias por día. La columna ganancias de la tabla ATRACCIONES, por cada atracción, debe ser la suma de las ganancias de todos los días en los que se celebró esa atracción).
* Copias de seguridad de los datos antes de que sean modificados o borrados.

Debemos de tener en cuenta que un trigger no se va a ejecutar en caso de una operación en cascada debido a las restricciones de integridad referencial (clave primaria-clave foránea).

Un trigger se puede definir para que se ejecute antes (before) o después (after) que la orden SQL que provoca su ejecución.

Normalmente se usa el tipo BEFORE para chequear que los datos son correctos antes de que sean incorporados a la tabla. Recordad que la verificación podemos implementarla con el uso de CHECK's cuando creamos la tabla, pero esta funcionalidad sólo está disponible a partir de la versión 8.0 de Mysql y dependiendo de la complejidad de la condición de verificación podría no ser posible realizarla haciendo uso de check.

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.

Operaciones con triggers

Crear Triggers

La orden sql que crea un trigger es [CREATE TRIGGER](https://dev.mysql.com/doc/refman/8.0/en/create-trigger.html).[Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente](https://wiki.cifprodolfoucha.es/index.php?title=Archivo:Mod_BD_Prog_Trigger_1.jpg)

Imagen obtenida de [https://dev.mysql.com](https://dev.mysql.com/)

Debéis utilizar la **orden DELIMITER** al igual que con los procedimientos almacenados, si los creáis desde una ventana de ejecución de consultas.

Es necesario tener el [permiso TRIGGER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html" \l "priv_trigger) otorgado para poder crear un trigger.

Dicho permiso se encuentra en el nivel de seguridad de tablas, por lo que es necesario indicar sobre qué tabla va a tener permiso para crear el trigger.

Básicamente cuando decidimos crear un trigger necesitamos determinar:

* Sobre qué tabla va a aplicarse el trigger.
* Sobre qué operación se va a aplicar (INSERT / DELETE / UPDATE)
* Queremos que el trigger se ejecute antes o después de que se realice la operación sobre la tabla (BEFORE => antes; AFTER => después).
* Indicar que si la operación SQL que va a provocar la ejecución del trigger afecta a múltiples filas (por ejemplo un borrado de muchas filas), el trigger se ejecutará una vez por cada fila afectada.
* Dentro del trigger vamos a poder acceder a los valores de la fila de la tabla anterior y posterior a la ejecución de la orden sql. Es decir, si realizo una operación de UPDATE sobre una columna, dicha columna tendrá un valor antes de la ejecución de la orden SQL y otro después de la ejecución, ya que le estoy enviando un nuevo valor. Dentro del trigger puedo acceder a dos 'alias' de la tabla a la que afecta el trigger con las mismas columnas que la tabla original:
* NEW: Tabla que posee los datos de cada columna con los nuevos valores.
* OLD: Tabla que posee los datos de cada columna con los valores antiguos.

Por ejemplo:

TABLA ALUMNOS:

---nif---- ---nombre----

---1a----- ---Angel-----

Realiza una operación UPDATE con la orden: UPDATE ALUMNOS SET nombre = 'Pedro' WHERE nif='1a'

Si tengo asociado un trigger UPDATE (de cualquiera de los dos tipos, before/after), dentro del código del trigger:

* NEW.nombre tendrá el valor 'Pedro'.
* OLD.nombre tendrá el valor 'Angel'.

Operaciones

* UPDATE:
* NEW: Valores nuevos
* OLD: Valores antiguos
* INSERT
* NEW: Valores nuevos
* OLD: NO EXISTE
* DELETE
* NEW: NO EXISTE
* OLD: Valores antiguos

La orden [LOAD DATA](https://dev.mysql.com/doc/refman/8.0/en/load-data.html) también provoca la ejecución del trigger ya que realiza operaciones de INSERT sobre las tablas.

Es posible tener más de un trigger sobre la misma tabla y con el mismo evento.

En estos casos, la ejecución se hará uno después de otro en el orden en que fueron creados.

Si queremos modificar dicho orden, a la hora de crear el trigger tendremos que utilizar la opción **trigger\_order**: { FOLLOWS | PRECEDES } other\_trigger\_name

De tal forma que el trigger que se está creando se ejecute antes (PRECEDES) o después (FOLLOWS) que el trigger que ya existe (other\_trigger\_name)

Cuando se define un trigger, este estás asociado al usuario que lo creó (CREATE **DEFINER=**`root`@`localhost` TRIGGER .....)

Dicho usuario debe tener el [permiso TRIGGER](https://dev.mysql.com/doc/refman/5.7/en/privileges-provided.html" \l "priv_trigger) otorgado con la [orden grant](https://dev.mysql.com/doc/refman/8.0/en/grant.html) sobre la tabla, lo que le da derecho a crear, borrar, mostrar y ejecutar un trigger.

Cuando otro usuario realiza una operación SQL sobre una tabla en la que está asociado el trigger, el usuario que creó el trigger debe conservar el permiso TRIGGER para que se pueda ejecutar.

Dentro del cuerpo del trigger vamos a poder acceder a los valores antiguos y nuevos como hemos visto.

Para poder modificar el valor nuevo (SET NEW.col=valor) el usuario que creó el trigger necesita tener permiso UPDATE sobre la tabla.

Para poder consultar el valor nuevo o antiguo (OLD.col o NEW.col) el usuario que creó el trigger necesita tener el permiso SELECT sobre la tabla.

**Gráficamente**

**Creando triggers desde MysqlWorkBench y phpmyadmin**[Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente](https://wiki.cifprodolfoucha.es/index.php?title=Archivo:Mysql_triggers_1.jpg)

Desde el MysqlWorkBench podemos ver los triggers creados y asociados a cada una de las tablas, **pero no se pueden crear desde aquí**.

[Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente](https://wiki.cifprodolfoucha.es/index.php?title=Archivo:Mysql_triggers_2.jpg)

Debemos ir a la opción de cambiar la estructura de la tabla (botón derecho sobre la tabla sobre la que vamos a querer crear un trigger y escoger Alter Table).

[Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente](https://wiki.cifprodolfoucha.es/index.php?title=Archivo:Mysql_triggers_3.jpg)

En la parte baja debemos de presionar sobre la pestaña Triggers. Una vez presionada ya podemos ver, crear y eliminar triggers de cada tipo. Sólo tenemos que pasar el ratón por encima del tipo correspondiente y aparecerán unos botones para poder crear (+) o eliminar (-) un trigger.

[Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente](https://wiki.cifprodolfoucha.es/index.php?title=Archivo:Mysql_triggers_4.jpg)

En PhpMyAdmin debemos de seleccionar una base de datos y escoger la opción superior que indica **Disparadores**.

[Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico, Sitio web

Descripción generada automáticamente](https://wiki.cifprodolfoucha.es/index.php?title=Archivo:Mysql_triggers_5.jpg)

Aparece la lista de triggers creados junto a la tabla asociada y el tipo de trigger. Podremos crear, borrar o modificar uno existente.

Visualizar triggers

La orden SQL para ver los triggers asociados a una base de datos o a una tabla es [SHOW TRIGGERS](https://dev.mysql.com/doc/refman/8.0/en/show-triggers.html).

Por defecto muestra los triggers de la base de datos activa. Si queremos mostrar los de otra debemos de utilizar la opción **FROM nombre\_BD** o **IN nombre\_BD**.

Si queremos buscar por los triggers asociados a una determinada tabla podemos hacer uso de la cláusula **LIKE** que busca por patrón (símbolo comodín el %) nombres de tablas.

Si queremos buscar dentro de los resultados aquellas filas que cumplan alguna condición asociada al valor de alguna columna debemos hacer uso la cláusula **WHERE**. Por ejemplo: show triggers where `Definer` like 'root%'

Hay que fijarse que en este caso, en la parte where se pueden emplear todas las 'columnas' que aparecen cuando ejecutamos la orden SHOW TRIGGER. Dichas columnas deben ir entre ` (acento grave).

Si queremos obtener información sobre un trigger específico por su nombre, debemos hacer uso de la orden SQL [SHOW CREATE TRIGGER](https://dev.mysql.com/doc/refman/8.0/en/show-create-trigger.html).

La base de datos y tabla donde se guarda toda la información de los triggers es: **information\_schema.triggers**

Modificar triggers

La **modificación de triggers NO es posible**: es **necesario borrar y volver a crear** el trigger.

Borrar triggers

La orden SQL para borrar un trigger es [DROP TRIGGER](https://dev.mysql.com/doc/refman/5.5/en/drop-trigger.html).

Aspectos varios

**Nombre de usuario que realiza la operación**

Como comentamos al principio, uno de los objetivos de utilizar triggers era el de registrar las operaciones sobre las tablas.

Uno de los datos que puede que necesitemos es el de obtener el nombre del usuario que realiza la operación.

Para obtenerlo podemos hacer uso de la [función USER()](https://dev.mysql.com/doc/refman/8.0/en/information-functions.html" \l "function_user).

Cuidado con utilizar la [función CURRENT\_USER()](https://dev.mysql.com/doc/refman/8.0/en/information-functions.html" \l "function_current-user) ya que esta devuelve el usuario que creó el trigger

**Funciones interesantes**

* NOW(): obtiene el DATETIME actual
* SYSDATE(): obtiene el DATETIME actual
* [CURDATE()](https://dev.mysql.com/doc/refman/5.5/en/date-and-time-functions.html" \l "function_curdate): Obtiene la fecha actual.
* DATE(SYSDATE()): obtiene el DATE actual
* TIME(SYSDATE()): obtiene el TIME actual
* [SIGNAL](https://dev.mysql.com/doc/refman/5.7/en/signal.html) SQLSTATE '45000' SET MESSAGE\_TEXT = 'mensaje a mostrar' (la veremos en control de errores)

Cancela la operación que se estaba realizando y muestra un mensaje.

Lleva el valor 45000 ya que este valor significa 'excepción definida por el usuario' (“unhandled user-defined exception”)

[DATE\_FORMAT(date,format)](https://dev.mysql.com/doc/refman/8.0/en/date-and-time-functions.html" \l "function_date-format): Formatea una fecha (por ejemplo, poner el resultado de la forma:día-mes-año).

Si queremos que los días-meses aparezcan en español, lo podemos cambiar con la variable del [sistema lc\_time\_names](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html" \l "sysvar_lc_time_names) o localmente con la orden: SET lc\_time\_names = 'es\_ES';

Ejercicios propuestos Triggers

Ejercicio 1

Haz que no se pueda añadir un nuevo animal si el tipo es 'León' y el número de años es mayor que 20.

En caso de no cumplirse la condición lanzará una excepción.

**Nota:** Fijarse que si el alta la realizamos a través de un procedimiento almacenado, la comprobación ya puede ir allí y no necesitaríamos el uso de triggers.

Siempre es mejor emplear procedimientos que triggers sobre los cuales no tenemos control.

Ejercicio 2

Cuando se añada un nuevo animal, hacer que dicho animal esté cuidado por el artista que cuida a menos animales.

Deberás de tener en cuenta el caso en el que no haya ningún artista cuidando animales. En ese caso debes coger el primer artista que no sea jefe.

Fijarse que este trigger va a suponer que se añaden dos filas: una fila a la tabla ANIMALES y otra a la tabla ANIMALES\_ARTISTAS. Por lo tanto, si queremos que ambas operaciones se hagan juntas, debería ir dentro de una transacción y controlar si el trigger provoca un error para hacer el ROLLBACK. El ROLLBACK no se puede hacer dentro del trigger. Habría que empezar la transacción en el código donde se realizar la orden SQL INSERT (por ejemplo en el programa cliente que esté realizando el programador o dentro de un procedimiento almacenado).

Ejercicio 3

Haz todo lo necesario para que el campo ganancias de la tabla ATRACCIONES se actualice cuando se añadan, borren o modifiquen datos en la tabla ATRACCION\_DIA.

Si al añadir una celebración nueva (ATRACCION\_DIA) la fecha\_inicio en ATRACCIONES es NULL, se debe de actualizar con la fecha actual.

**Nota:** Debéis de tener en cuenta que tanto la fecha como las ganancias en ATRACCIONES pueden tener valores nulos. Hacer uso de la [función IFNULL](https://dev.mysql.com/doc/refman/8.0/en/control-flow-functions.html" \l "function_ifnull) para dar un valor a esas columnas cuando sean null (recordar que si sumamos una columna con valor null siempre devolverá null).