



Clase 17. Curso SQL

TRIGGERS

***RECUERDA PONER A GRABAR LA
CLASE***





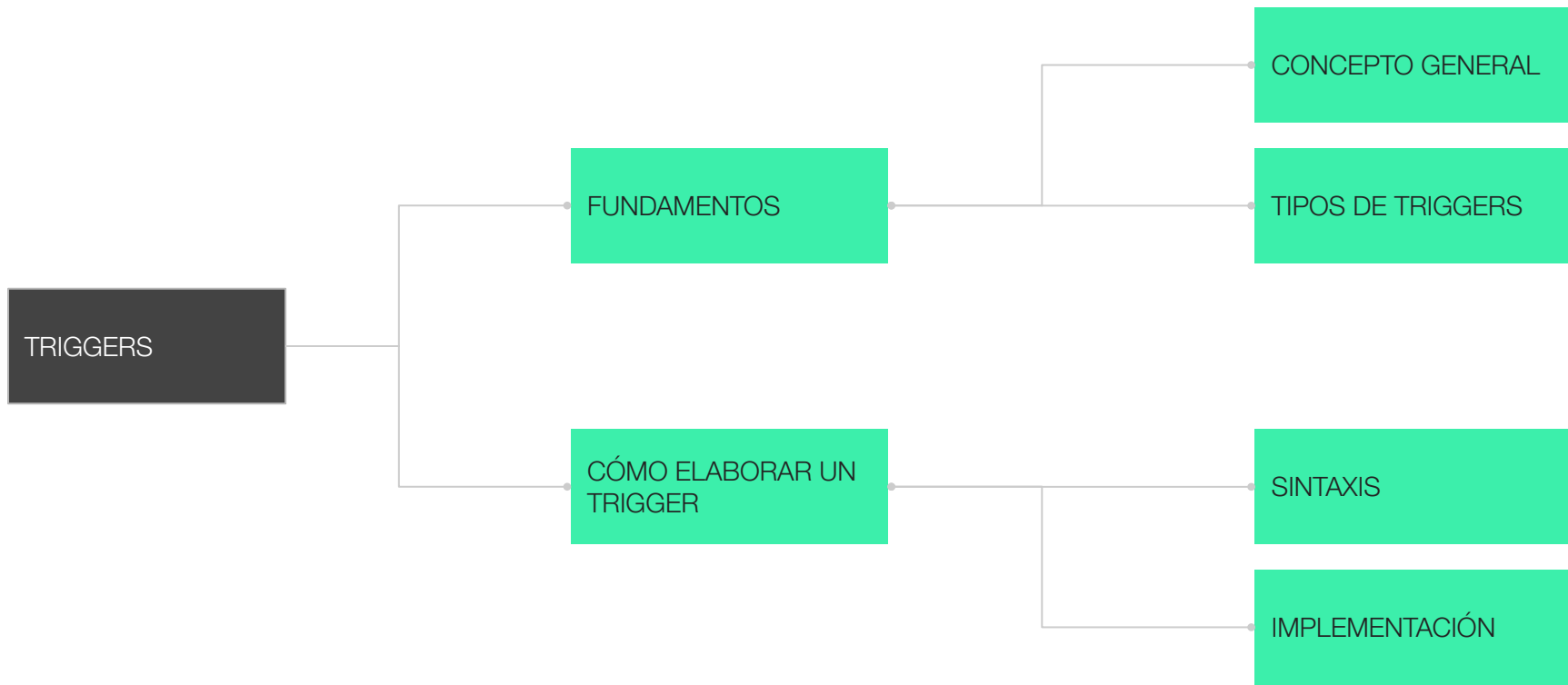
OBJETIVOS DE LA CLASE

- Concepto general de un Trigger
- Tipos de Triggers
- Sintaxis e implementación de un Trigger

MAPA DE CONCEPTOS

MAPA DE CONCEPTOS CLASE 17

¡Para
recordar!



TRIGGERS

CONCEPTO GENERAL



TRIGGER: DEFINICIÓN

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.



TRIGGER: DEFINICIÓN



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

TRIGGER: 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.

TRIGGER: USOS ESPECÍFICOS



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.

TRIGGER: TABLAS DE AUDITORÍA



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.

TRIGGER: TABLAS DE AUDITORÍA



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.

TRIGGER: TABLAS DE AUDITORÍA



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



TIPOS DE TRIGGER



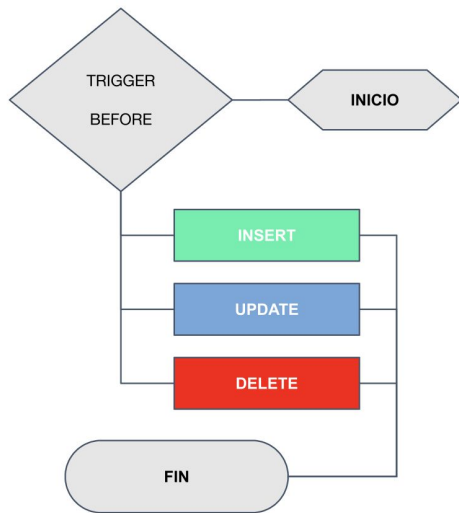
El uso de Trigger se puede establecer en dos momentos diferentes de cuando se realiza una operación del tipo UPDATE, DELETE, o INSERT.

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



TIPOS DE TRIGGER: BEFORE

BEFORE



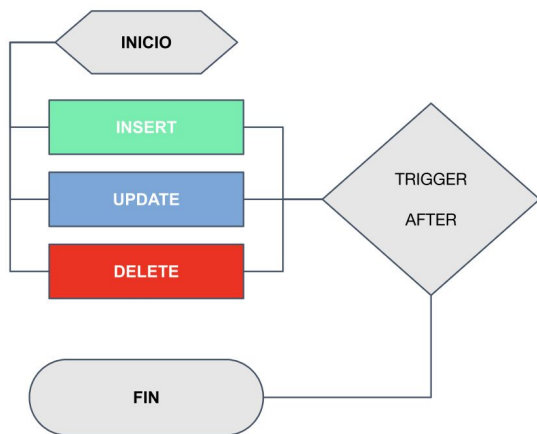
Cuando el usuario envía una operación del tipo INSERT, UPDATE o DELETE sobre una tabla, y esta tiene activo el Trigger que detecta la operación, se disparará la acción BEFORE, la cual permitirá por ejemplo registrar en una tabla de auditoría que se realizará la operación xx sobre la tabla yy.



TIPOS DE TRIGGER: AFTER

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.



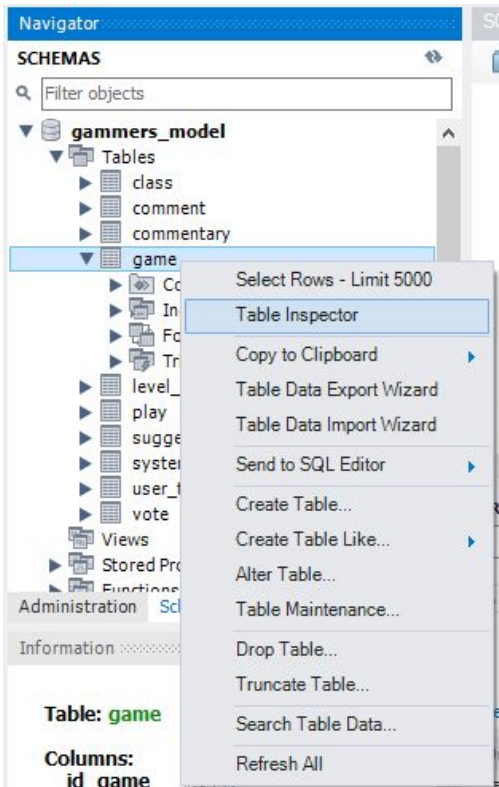
IDENTIFICAR TRIGGERS EN UNA TABLA



TIPOS DE TRIGGER: 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.





TIPOS DE TRIGGER: AFTER

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.

Info	Columns	Indexes	Triggers	Foreign keys	Partitions	Grants	DDL						
Name		Event		Timing		Created		SQL Mode		Definer		Client Character...	
 tr_add_new_game		INSERT		AFTER		2021-12-15 09:1...		STRICT_TRANS_...		root@localhost		utf8mb4	

SINTAXIS

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.



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.



SINTAXIS

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**).

```
CREATE TRIGGER `accion_y_nombre_del_trigger`  
AFTER INSERT ON `productos`  
...
```



SINTAXIS

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

CODER HOUSE



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.



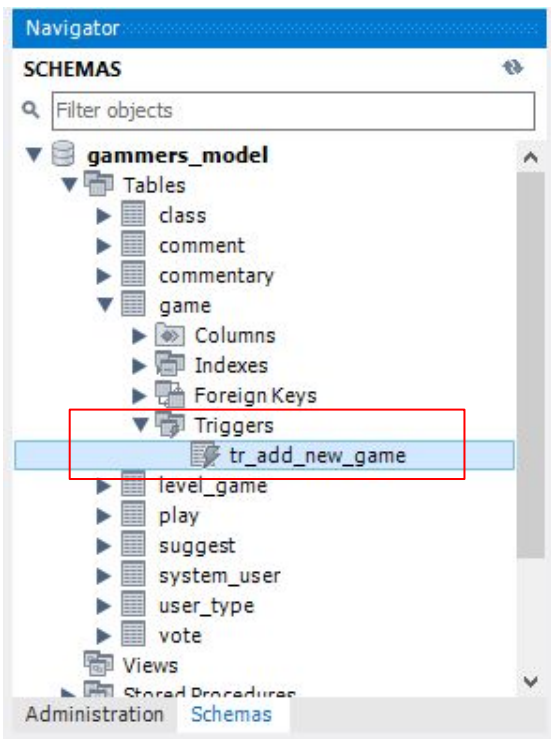
NEW

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

```
CREATE TRIGGER `accion_y_nombre_del_trigger`  
AFTER INSERT ON `productos`  
INSERT INTO `tabla_auxiliar` (campo1, campo2...)   
VALUES (NEW.campo1, NEW.campo2...)
```



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**.



BREAK

¡5/10 MINUTOS Y VOLVEMOS!

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**.

FOR EACH ROW



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

```
CREATE TRIGGER `accion_y_nombre_del_trigger`  
AFTER INSERT ON `productos`  
FOR EACH ROW  
INSERT INTO `tabla_auxiliar` (campo1, campo2...)  
VALUES (NEW.campo1, NEW.campo2...)
```

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 debemos ejecutar.

INTEGRACIÓN DE FUNCIONES

FUNCIONES DEL SISTEMA



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.



FUNCIONES DEL SISTEMA

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()

```
1 ● SELECT NOW();  
2 ● SELECT CURRENT_DATE();  
3 ● SELECT CURDATE();  
4 ● SELECT CURRENT_TIME();  
5 ● SELECT CURTIME();  
6 ● SELECT CURRENT_TIMESTAMP();
```



FUNCIONES DE USUARIO

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

- SESSION_USER()
- SYSTEM_USER()
- USER()

```
1 ● SELECT SESSION_USER();  
2 ● SELECT SYSTEM_USER();  
3 ● SELECT USER();
```



FUNCIONES DE PLATAFORMA

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

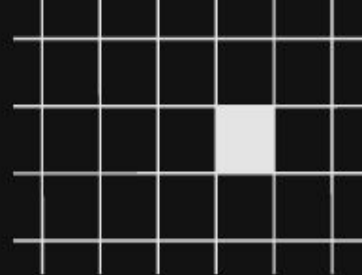
- DATABASE()
- VERSION()

```
1 ● SELECT DATABASE();  
2 ● SELECT 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.

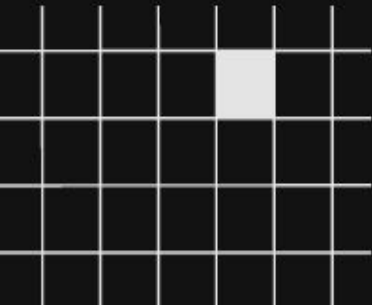
Y las funciones de Usuario son sumamente útiles cuando trabajamos con bases de datos correctamente estructuradas a nivel de seguridad, estableciendo diferentes usuarios y permisos para cada objeto que la integra.



EJEMPLO EN VIVO

Veremos cómo implementar Triggers algo más complejos.

Pueden seguir a la par, en sus computadoras, el ejemplo en vivo.





IMPLEMENTAR TRIGGER

Crearemos un 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**

```
1 • SELECT * from game;
```

	id_game	name	description	id_level	id_class
▶	1	Forza Horizon 5	odio donec	2	143
	2	Call of Duty: Vanguard	morbi non	6	153
	3	Shin Megami Tensei 5	turpis integer aliquet massa id	3	243
	4	Marvels Guardianes de la Galaxia	lobortis sapien sapien non mi	4	245
	5	Age of Empires IV		2	50
	6	Football Manager 22	nulla suspendisse potenti	8	236
	7	Football Manager 22	mauris lacinia sapien quis libero	11	173
	8	Blue Reflection: Second Light	libero rutrum ac lobortis	2	18
	9	Darkest Dungeon II	erat nulla	11	90
	10	Voice of Cards	parturient montes nascetur	2	275
	11	Elden Ring	et ultrices posuere cubilia curae	15	208
	12	FIFA 22: Ultimate Team	suspendisse	14	205



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**).

```
1 • ○ CREATE TABLE new_games (  
2     id_game INT PRIMARY KEY,  
3     name varchar(100),  
4     description varchar(300)  
5     );
```



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.




```
1 • CREATE TRIGGER `tr_add_new_game`  
2   AFTER INSERT ON `game`  
3   FOR EACH ROW  
4   INSERT INTO `new_games` (id_game, name, description) VALUES (NEW.id_game, NEW.name, NEW.description);  
-
```



SCRIPT DEL TRIGGER

Verifiquemos si el Trigger fue creado exitosamente.

Info	Columns	Indexes	Triggers	Foreign keys	Partitions	Grants	DDL	
Name	Event	Timing	Created	SQL Mode	Definer	Client Character...		
 tr_add_new_game	INSERT	AFTER	2021-12-15 09:1...	STRICT_TRANS_...	root@localhost	utf8mb4		

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

```
INSERT INTO game (id_game, name, description, id_level, id_class)
VALUES (150, 'Mortal Kombat', 'play station', 2, 143);
```



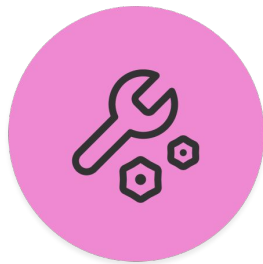
Finalmente verás que el Trigger almacenó la operación realizada, copiando los campos indicados en esta tabla de **new_games**.

The screenshot shows a database management tool interface. At the top, there is a toolbar with various icons. Below the toolbar, a SQL query is entered in a text area:

```
1 • SELECT * FROM new_games;  
2  
3  
4  
5  
6
```

Below the query editor, there is a section labeled "Result Grid". It includes a "Filter Rows:" input field. The result grid displays the following data:

	id_game	name	description
▶	150	Mortal Kombat	play station
★	NULL	NULL	NULL



CREACIÓN DE TRIGGERS

Practicaremos la implementación de Triggers.

Tiempo estimado: 15 minutos

CREACIÓN DE TRIGGERS

Desafío
generico



Con la tabla **comment** del modelo gamers deberás crear dos triggers:

1. Debe detectar la modificación de un registro de la tabla en cuestión, justo antes de que ocurra, almacenando en otra tabla **first_date** y **last_date**
2. Debe detectar la eliminación de un registro, posterior a dicha operación, registrando también fecha y hora, más el usuario que lo eliminó

La tabla bitácora puede ser un espejo de la tabla elegida o una tabla del tipo LOG.



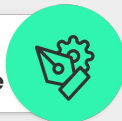
SCRIPT DE CREACIÓN DE TRIGGERS

Presentar en formato .sql el script de creación de 2 triggers con base en los datos de la base de datos del proyecto final.

SCRIPT DE CREACIÓN DE TRIGGERS

Formato: El archivo a presentar debe ser del tipo .sql nombrado como “Triggers+Apellido”.

Desafío
entregable



>> Consigna: En la base de datos de tu proyecto final, debes incluir al menos dos tablas del tipo LOG, Bitácora o Movimientos. Elegir dos de las tablas más importantes donde se operan con registros de forma frecuente, y crearás dos Triggers en cada una de ellas.

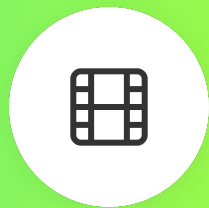
→ Los Triggers a crear deberán controlar la acción previo a la operación elegida (**BEFORE**), y una acción posterior a otra operación elegida (**AFTER**).

>>Aspectos a incluir en el entregable:

Agrega una explicación por cada trigger a crear, explicando qué controlará el mismo.

Recuerda agregar el script de creación de las tablas LOG.

Debe registrar **el usuario** que realizó la operación, **la fecha**, y **la hora** (*en campos separados*).



***¿QUIERES SABER MÁS? TE DEJAMOS
MATERIAL AMPLIADO DE LA CLASE***



- [Mysql Workbench - Triggers](#) | **SAKLAR**
- [Mysql Functions](#) | **W3Schools**

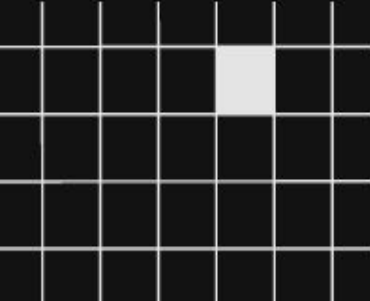
¿PREGUNTAS?





¡MUCHAS GRACIAS!

Resumen de lo visto en clase hoy:

- Fundamentos de Triggers.
 - Sintaxis de creación de Triggers.
 - Formas de implementación de un Trigger.
- 



OPINA Y VALORA ESTA CLASE

#DEMOCRATIZANDO LA EDUCACIÓN

CODER HOUSE