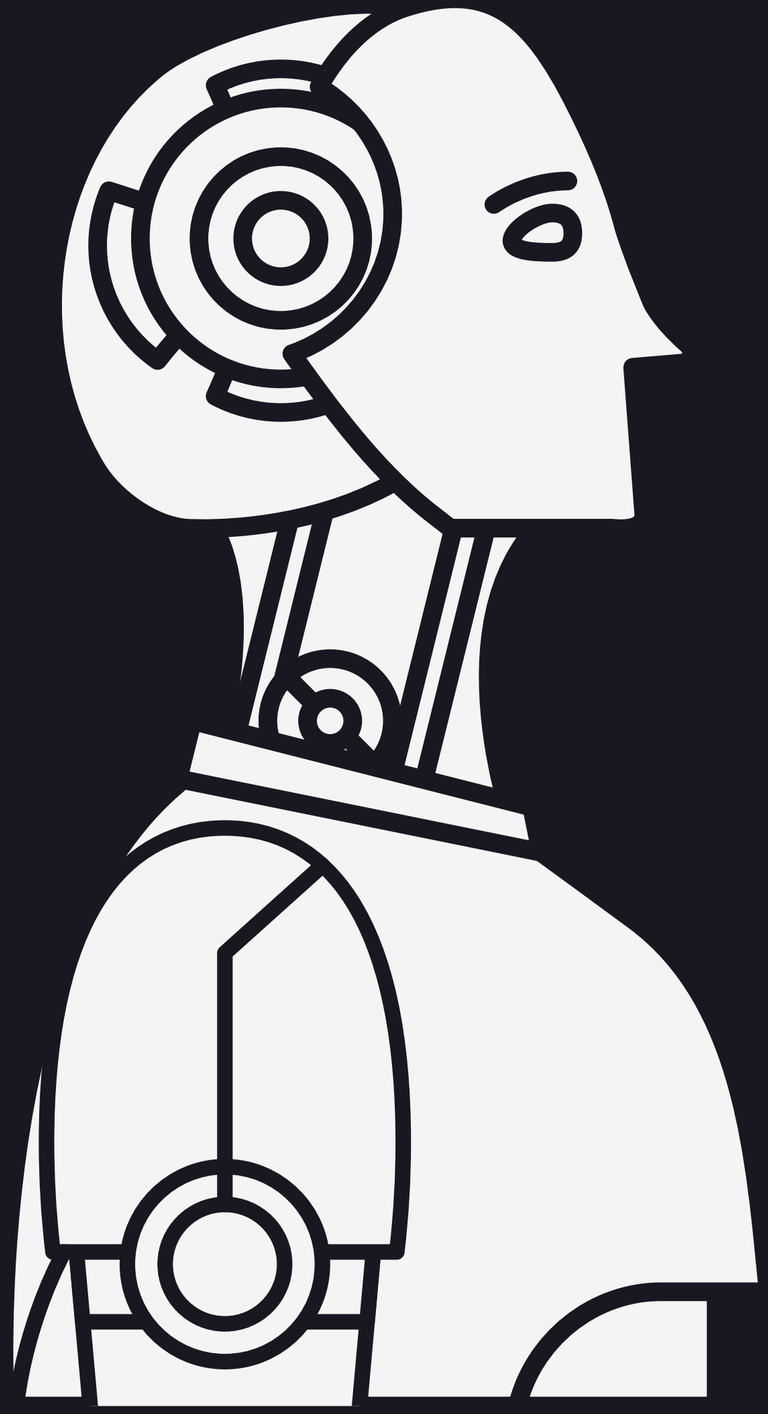

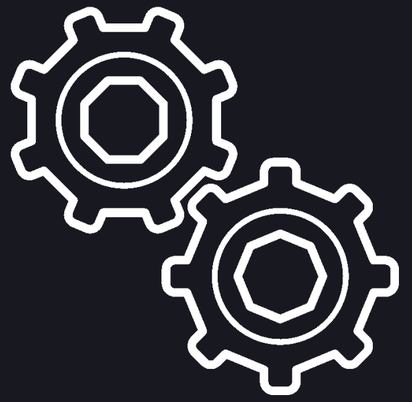

TRIGGERS

GAES 4 - DYS

PRESENTA

DANIEL ESTEBAN GONZALEZ MARROQUIN
YEISON DAVID DE LA ROSA URIBE

TRIGGERS

Un trigger, también conocido como disparador (Por su traducción al español) es un conjunto de sentencias SQL las cuales se ejecutan de forma automática cuando ocurre algún evento que modifique a una tabla. Pero no me refierón a una modificación de estructura, no, me refiero a una modificación en cuando a los datos almacenados, es decir, cuando se ejecute una sentencia INSERT, UPDATE o DELETE.

INSERT: EL TRIGGER SE ACTIVA CUANDO SE INSERTA UNA NUEVA FILA SOBRE LA TABLA ASOCIADA.

UPDATE: EL TRIGGER SE ACTIVA CUANDO SE ACTUALIZA UNA FILA SOBRE LA TABLA ASOCIADA.

DELETE: EL TRIGGER SE ACTIVA CUANDO SE ELIMINA UNA FILA SOBRE LA TABLA ASOCIADA.

```
1   CREATE TRIGGER TR_STOCK
2   ON [dbo].[Detalles de pedidos] AFTER INSERT,UPDATE
3   AS
4   DECLARE @CAN INT
5   DECLARE @STOCK INT
6   DECLARE @PRO INT
7
8   SET @CAN= (SELECT Cantidad FROM inserted);
9   SET @PRO= (SELECT IdProducto FROM inserted);
10  SET @STOCK=(SELECT UnidadesEnExistencia FROM Productos WHERE IdProducto=@PRO)
11  IF @CAN <=@STOCK
12  BEGIN
13      UPDATE Productos SET UnidadesEnExistencia =-UnidadesEnExistencia - @CAN
14      WHERE IdProducto=@PRO
15  END
16  ELSE
17
18  BEGIN
19      RAISERROR('LA CANTIDAD EXCEDE AL STOCK',16,10)
20  END
```

```
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
```

PROGRAMAR LOS TRIGGERS DE TAL MANERA QUE SE EJECUTEN ANTES O DESPUÉS, DE DICHAS SENTENCIAS; DANDO COMO RESULTADO SEIS COMBINACIONES DE EVENTOS.

- BEFORE INSERT: ACCIONES A REALIZAR ANTES DE INSERTAR UNO MÁS O REGISTROS EN UNA TABLA.
- AFTER INSERT: ACCIONES A REALIZAR DESPUÉS DE INSERTAR UNO MÁS O REGISTROS EN UNA TABLA.
- BEFORE UPDATE: ACCIONES A REALIZAR ANTES DE ACTUALIZAR UNO MÁS O REGISTROS EN UNA TABLA.
- AFTER UPDATE: ACCIONES A REALIZAR DESPUÉS DE ACTUALIZAR UNO MÁS O REGISTROS EN UNA TABLA.
- BEFORE DELETE: ACCIONES A REALIZAR ANTES DE ELIMINAR UNO MÁS O REGISTROS EN UNA TABLA.
- AFTER DELETE: ACCIONES A REALIZAR DESPUÉS DE ELIMINAR UNO MÁS O REGISTROS EN UNA TABLA.

¿Cuándo ejecuta su acción un Trigger?
Los triggers pueden ejecutar su acción en diferentes momentos.

- Antes de ejecutar la sentencia (before statement)
- Después de ejecutar la sentencia y de comprobar las restricciones y condiciones aplicables (after statement)
- Antes de modificar la fila de la tabla afectada por la sentencia del trigger, y de comprobar las restricciones y condiciones de ejecución (before row)
- Después de modificar la fila de la tabla afectada por la sentencia del trigger, y de comprobar las restricciones y condiciones de ejecución (after row)

TIPOS DE TRIGGER
EXISTEN 2 TIPOS DE DISPARADORES, EN FUNCIÓN DE LAS EJECUCIONES QUE REALIZAN.

- DISPARADORES DE FILA: TAMBIÉN LLAMADOS ROW TRIGGERS, SON AQUELLOS CUYA EJECUCIÓN SE REALIZA A TRAVÉS DE LLAMADAS DESDE UNA TABLA ASOCIADA AL TRIGGER.

- DISPARADORES DE SECUENCIA. TAMBIÉN LLAMADOS STATEMENT TRIGGERS, SON AQUELLOS QUE SE EJECUTAN SOLO UNA VEZ, INDEPENDIENTEMENTE DE LA CANTIDAD DE VECES QUE SE CUMPLAN LAS CONDICIONES PARA SU EJECUCIÓN.

EJEMPLO: TRIGGER DE OPERACIÓN

```
CREATE TABLE PRODUCTO(CODIGO NUMBER(6) PRIMARY KEY,
                      PRECIO NUMBER(6));

CREATE OR REPLACE TRIGGER CONTROL_INSERCION_PRODUCTOS
BEFORE INSERT ON PRODUCTO
BEGIN

 IF TO_CHAR(SYSDATE,'D') = '7'THEN
  RAISE_APPLICATION_ERROR(-20506,'¡NO LOS DOMINGOS!');
 END IF;
END;
/
```

DOMINGO (PUEDE SER 1 O 7 SEGÚN LA CONFIG. DEL NLS)

SQLCODE Y SQLERRM ASIGNADOS POR EL USUARIO

LA INSTRUCCIÓN RAISE_APPLICATION_ERROR ABORTARÁ LA OPERACIÓN DE INSERCIÓN SI ESTA SE INTENTA HACER UN DOMINGO.

EJEMPLO: TRIGGER DE FILA

```
CREATE TABLE CUENTA(
NRO_CTA NUMBER(10) PRIMARY KEY,
SALDO NUMBER(8) NOT NULL CHECK (SALDO >= 0)
);

INSERT INTO CUENTA VALUES(1, 100);
INSERT INTO CUENTA VALUES(2, 200);

CREATE TABLE RETIRO(
NRO_RETIRO NUMBER(10) PRIMARY KEY,
CTA NUMBER(10) REFERENCES CUENTA NOT NULL,
VALOR NUMBER(8) NOT NULL CHECK (VALOR > 0)
);
```

CREACIÓN Y USO DE VARIABLES EN UN TRIGGER

EN FUNCIONES Y PROCEDIMIENTOS ALMACENADOS, EN UN TRIGGER PODEMOS CREAR VARIABLES Y MANIPULARLAS. LA CREACIÓN DE UNA VARIABLE SE REALIZA MEDIANTE EL COMANDO DECLARE SEGUIDO DEL NOMBRE DE LA VARIABLE Y SU TIPO. POR EJEMPLO:

```
1. DECLARE $var1 SMALLINT;
```

Y LA ASIGNACIÓN O MODIFICACIÓN DE SU VALOR MEDIANTE EL COMANDO SET:

```
1. SET $var1 = 38;
```

Ejemplo

```
1.  DELIMITER $$
2.  CREATE TRIGGER trigger_usuario_identificador
3.  AFTER UPDATE ON usuario
4.  FOR EACH ROW
5.  BEGIN
6.    DECLARE $identificador TEXT;
7.    SET $identificador = CONCAT(NEW.id, '_', NEW.nombre, '_', NEW.dni);
8.    UPDATE usuario
9.    SET identificador = $identificador
10.   WHERE id = OLD.id;
11. END; $$
```

LA ESTRUCTURA DE UN TRIGGER EN ORDEN ES LA SIGUIENTE:

1. DEFINIR UN SÍMBOLO O CONJUNTO DE ELLOS QUE SERÁN EL DELIMITADOR PARA NUESTRA PROGRAMACIÓN. PARA ESTO USAREMOS LA SENTENCIA DELIMITER.
2. UTILIZAR EL COMANDO CREATE DE SQL PARA CREAR EL DISPARADOR Y ASOCIARLE UN NOMBRE: CREATE TRIGGER NOMBRE_DISPARADOR
3. INDICAR UN EVENTO SOBRE UNA TABLA QUE PRODUCIRÁ LA EJECUCIÓN DEL DISPARADOR: AFTER/BEFORE DELETE/UPDATE/INSERT ON TABLA
4. ESCRIBIR FOR EACH ROW PARA QUE SE EJECUTE PARA CADA REGISTRO INSERTADO/ACTUALIZADO/BORRADO.
5. DELIMITAR LA ZONA DE PROGRAMACIÓN MEDIANTE BEGIN Y END.
6. POR ÚLTIMO CERRAR LA DECLARACIÓN DEL TRIGGER MEDIANTE EL USO DEL SÍMBOLO O CONJUNTO DE SÍMBOLOS QUE INDICAMOS AL PRINCIPIO.

EJEMPLOS

- CREAR UNA COPIA AUTOMATIZADA QUE REGISTRE CADA CLIENTE QUE ENTRA EN UN ECOMMERCE.
- CONTABILIZAR ESTADÍSTICAS SOBRE ACCESOS A UNA BASE DE DATOS Y LAS ACCIONES QUE SE HAN LLEVADO A CABO EN ELLAS.
- DUPLICAR TABLAS CON INFORMACIÓN SINCRONIZADA.
- PROGRAMAR LA BASE DE DATOS PARA QUE SE REALICE UN PEDIDO DE FORMA AUTOMÁTICA CUANDO EL STOCK HA LEGADO A UN MÍNIMO DETERMINADO DE UNIDADES.
- BLOQUEAR LA INSERCIÓN DE DATOS INCORRECTOS O DE TRANSACCIONES INVÁLIDAS.
- CREAR ALERTAS QUE AVISEN DE CUANDO SE HA MODIFICADO ALGÚN DATO EN UNA TABLA.

TRIGGER QUE NOS PERMITA MANTENER UNA COPIA DE TODOS LOS CLIENTES QUE SE INSERTEN EN UNA BASE DE DATOS DE UNA TIENDA ONLINE.

```
1. DELIMITER $$
2. CREATE TRIGGER trigger_cliente_historico
3. AFTER INSERT ON cliente
4. FOR EACH ROW
5. BEGIN
6.   INSERT INTO cliente_historico(nombre, dni, direccion)
7.   VALUES (NEW.nombre, NEW.dni, NEW.direccion, CURDATE());
8. END; $$
```

TRIGGER INSERTARÁ UN VALOR NUEVO EN UN CAMPO ÚNICO IDENTIFICADOR TRAS UNA ACTUALIZACIÓN DE DATOS EN LA TABLA USUARIO_WEB

```
1. DELIMITER $$
2. CREATE TRIGGER trigger_usuario_identificador
3. AFTER UPDATE ON usuario
4. FOR EACH ROW
5. BEGIN
6.   UPDATE usuario SET identificador = CONCAT(NEW.id, '_', NEW.nombre, '_', NEW.dni)
7.   WHERE id = OLD.id;
8. END; $$
```

WEBGRAFIA

https://ayudaleyprotecciondatos.es/bases-de-datos/trigger/
https://codigofacilito.com/articulos/triggers_mysql
https://josejuansanchez.org/bd/unidad-12-teoria/index.html#triggers
https://www.srcodigofuente.es/aprender-sql/triggers-sql
https://www.tutorialesprogramacionya.com/oracleya/temarios/descripcion.php?cod=261&punto=1&inicio=