

Investigación sobre Disparadores en Bases de Datos

◇ ¿Qué son los Disparadores?

Los **disparadores**, también conocidos como **triggers**, son **objetos de base de datos que se asocian a una tabla o vista**, y que se ejecutan automáticamente cuando ocurre un determinado evento de manipulación de datos. Estos eventos pueden ser:

- **INSERT** (cuando se inserta una nueva fila)
- **UPDATE** (cuando se actualiza una fila existente)
- **DELETE** (cuando se elimina una fila)

En lugar de ser llamados directamente por el usuario o la aplicación, los disparadores **se activan en segundo plano** como respuesta a estos eventos.

📌 **Importante:** No todos los sistemas de gestión de bases de datos (SGBD) soportan disparadores de la misma forma. Algunos como **PostgreSQL**, **MySQL**, **Oracle** y **SQL Server** ofrecen soporte completo, pero con ligeras diferencias de sintaxis.

◇ ¿Para qué sirven los Disparadores?

Los disparadores tienen diversos usos en la administración y lógica de las bases de datos. Entre los más comunes están:

🔗 1. Control de integridad de datos

Permiten imponer reglas que van más allá de las restricciones tradicionales como NOT NULL, CHECK o FOREIGN KEY. Por ejemplo, evitar que se hagan cambios si una condición no se cumple.

🕵️ 2. Auditoría de datos

Se pueden usar para registrar automáticamente quién hizo una modificación, qué datos se cambiaron y cuándo. Esto es útil para mantener un historial de cambios o cumplir con normativas legales.

3. Replicación o sincronización

Se pueden actualizar datos automáticamente en otras tablas relacionadas cuando ocurre una modificación en la tabla principal.

4. Automatización de procesos

Por ejemplo, enviar notificaciones, calcular valores automáticamente, llenar otras tablas, etc., sin intervención manual ni lógica desde la aplicación.

◇ Tipos de Disparadores

Según el momento de ejecución:

- **BEFORE:** Se ejecuta **antes** de que se realice la operación (permite modificar los datos antes de que sean insertados o actualizados).
- **AFTER:** Se ejecuta **después** de la operación (útil para auditorías o procesos dependientes de los datos finales).
- **INSTEAD OF:** En algunos SGBD (como SQL Server), reemplaza la operación (útil para vistas que no pueden modificarse directamente).

Según el tipo de operación:





- **INSERT**
- **UPDATE**
- **DELETE**

Según el nivel de ejecución:





- **FOR EACH ROW:** Se ejecuta una vez por cada fila afectada.
- **FOR EACH STATEMENT:** Se ejecuta una sola vez por cada sentencia, sin importar cuántas filas sean afectadas (menos común y depende del SGBD).

◇ Ventajas y Desventajas

Ventajas

Ventaja	Descripción
 Seguridad	Se garantiza que ciertas reglas se cumplan sin depender del código externo.
 Automatización	Se pueden ejecutar tareas complejas automáticamente en la base de datos.
 Auditoría	Permiten registrar cambios sin alterar la lógica del programa.
 Integridad referencial avanzada	Más allá de las restricciones estándar.

Desventajas

Desventaja	Descripción
 Rendimiento	Pueden hacer más lentas las operaciones si el trigger realiza tareas pesadas.
 Complejidad	Dificultan la depuración y el mantenimiento del sistema.
 Ejecución oculta	El programador puede olvidar que hay lógica en el trigger, generando resultados inesperados.
 Orden no garantizado	Si hay varios triggers, su orden de ejecución puede no estar bien definido.

Sintaxis y Uso en PostgreSQL

Paso 1: Crear la función que se ejecutará

```
CREATE OR REPLACE FUNCTION log_actualizaciones()
RETURNS TRIGGER AS $$
BEGIN
    INSERT INTO auditoria (tabla, operacion, fecha)
    VALUES (TG_TABLE_NAME, TG_OP, NOW());
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

- TG_OP: devuelve el tipo de operación (INSERT, UPDATE, DELETE).
- TG_TABLE_NAME: devuelve el nombre de la tabla que activó el trigger.

Paso 2: Crear el trigger asociado

```
CREATE TRIGGER trigger_auditoria  
AFTER INSERT OR UPDATE OR DELETE ON empleados  
FOR EACH ROW  
EXECUTE FUNCTION log_actualizaciones();
```