Auditorias de BDs

Antonio Espín Herranz

Auditoria

- Es el proceso que permite medir, asegurar, demostrar, monitorear y registrar los accesos a la información en las bases de datos incluyendo la capacidad de determinar:
  - Quién accede a los datos,
  - Cuando, se accedió a los datos,
  - Desde qué tipo de dispositivo o aplicación,
  - Desde que ubicación en la red, c
  - Cuál fue la sentencia SQL ejecutada y
  - Cuál fue el efecto del acceso a la base de datos.

Auditoria

- La auditoría de base de datos no es más que un proceso de control de la información que posee una organización, con fines estadísticos y sobre todo de seguridad,
  - Con respecto a las transacciones (INSERT, UPDATE, DELETE) que se realizan.

Sistemas para auditar BD PostgreSQL

- Los sistemas para realizar auditorías pueden ser de **dos tipos**:
  - **En los ficheros de log del servidor**.
  - **O basados en disparadores (triggers)**

Auditoria en logs del servidor - **pros y contras**

- Puede ser sencilla de implementar
- Pero no es una buena idea por:
  - El log actúa sobre todo el servidor, no sólo sobre la base de datos que queramos auditar, y tampoco filtra por objetos por lo que los ficheros de log se volverían inmensos.
  - El log es volátil por lo que tendrás que implementar un proceso que extraiga las pistas de auditoría del log para conservarlas en ficheros permanentes y además eliminar de esos ficheros el «ruido».
  - Las sentencias que se hayan ejecutado y fallen no tendrán un efecto en la base de datos sin embargo podría ser considerada por error una pista de auditoría válida.

Auditoria basada en triggers - **pros y contras**

- Realizados de una forma programática: uso de plpgsql, dblink, triggers y funciones.
  - Forma nativa en PostgreSQL podríamos utilizar disparadores para grabar en tablas, los eventos que ocurran y que sean de interés para la auditoría, sobre los objetos concretos que se precisan auditar. Incluso de campos concretos de unas determinadas tablas.
  - Con este sistema también podemos formar la pista de auditoría exacta, con la información tan detallada como sea preciso.

- Hay que programar, pero es flexible y se puede adaptar a nuestras necesidades.

Herramientas externas

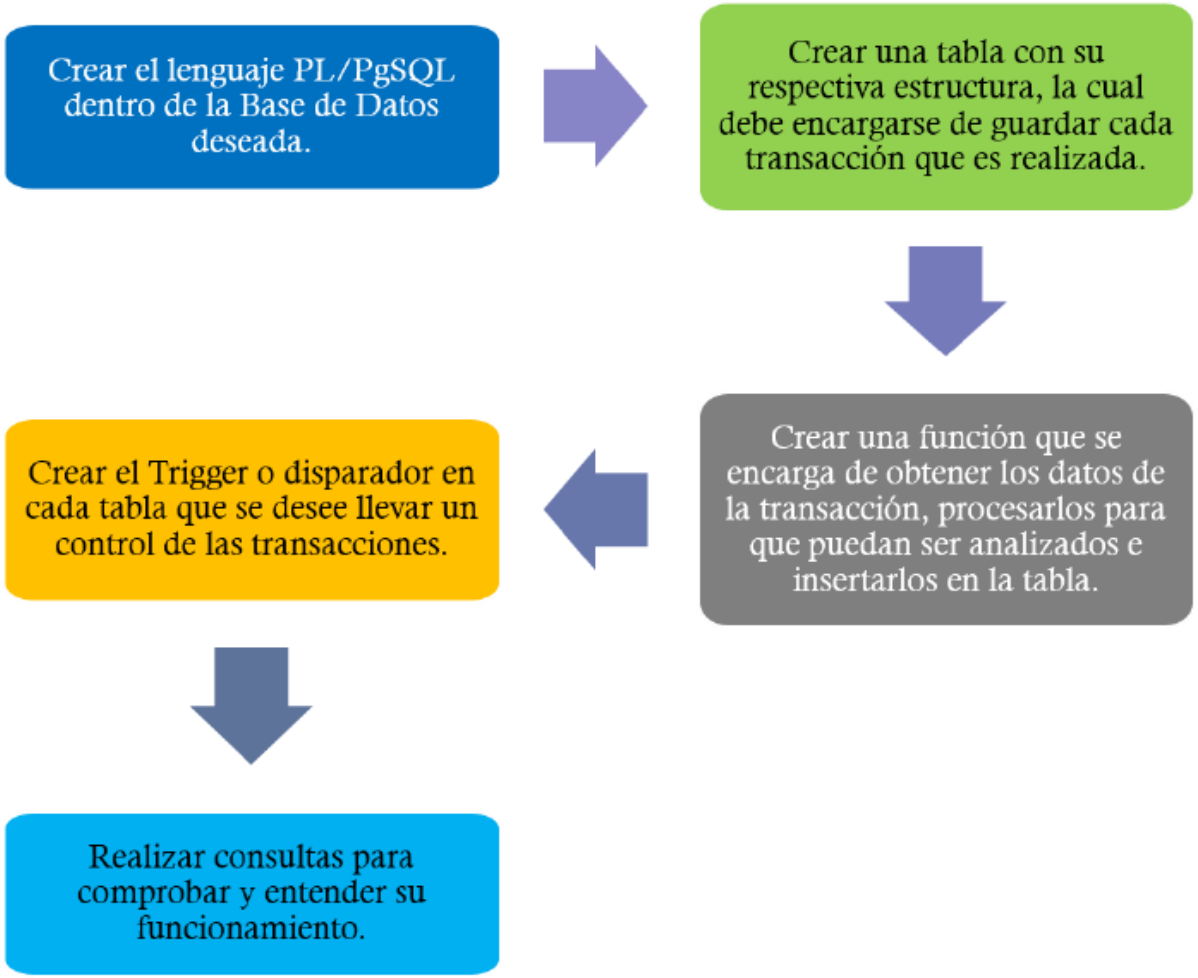
- **Basada en Triggers:**
  - **2ndQuadrant** ha implementado un sistema de auditoría basado en triggers, listo para instalar y utilizar, llamado **audit-trigger 91 plus**, que podrías considerar si buscas una herramienta libre y evitarte el trabajo de tener que desarrollarla.
  - **Enlace Git:** https://github.com/2ndQuadrant/audit-trigger

- **Basada en Logs:**
  - **pgAudit** tiene dos modos de funcionamiento:
  - registro de auditoría a nivel de sesión y
    - por si estamos interesados en una amplia colección de eventos de la sesión, independientemente de los objetos de los que se trate
  - registro de auditoría a nivel de objetos,
    - o si estamos interesados únicamente en registrar las acciones sobre algunas relaciones o sobre determinados campos de dichas relaciones.

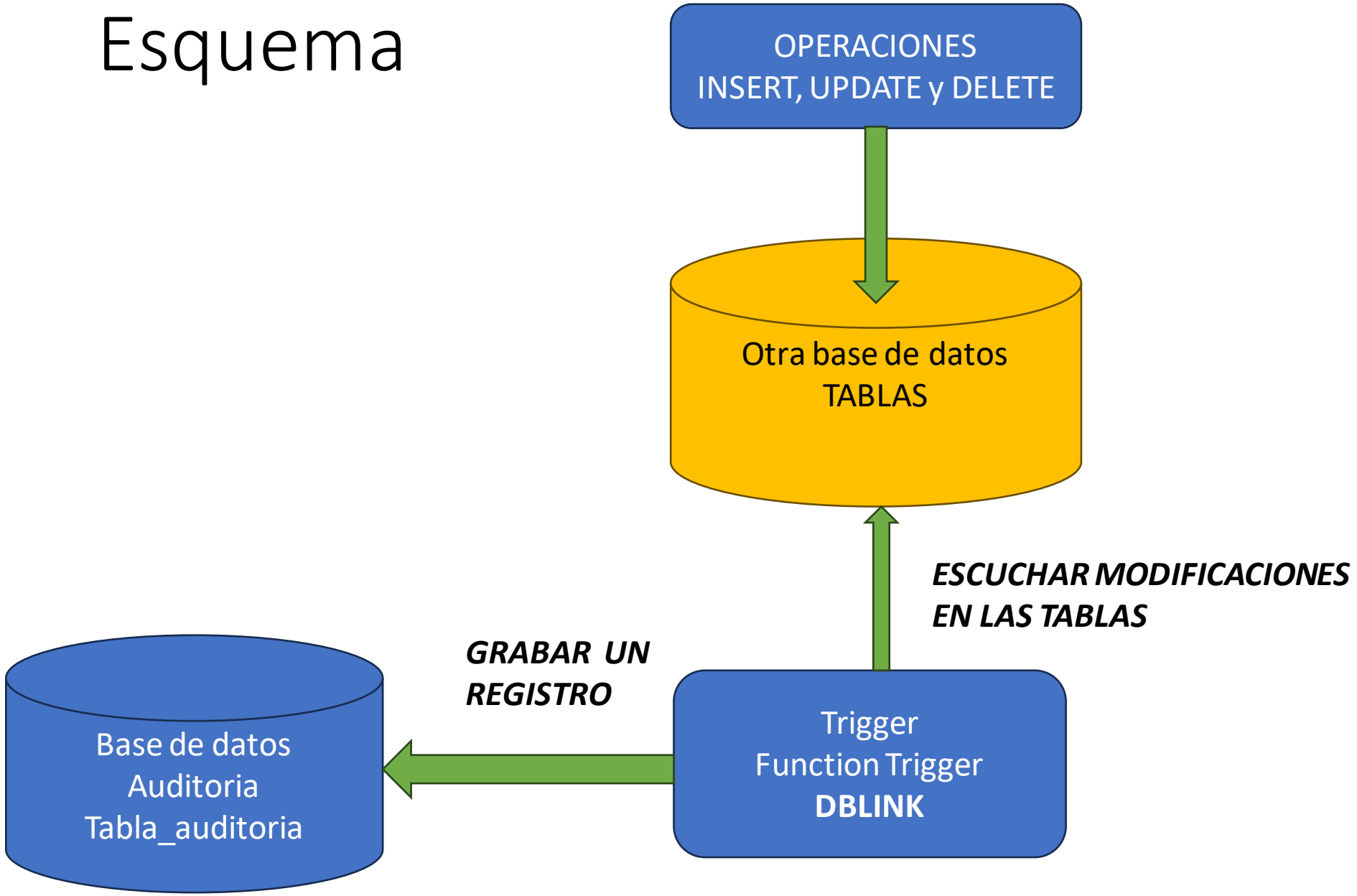
  - En ambos casos, la pista de auditoría se graba en el log con la etiqueta AUDIT y en formato CSV, lo cual facilita la tarea de reconocer las pistas de auditoría del resto de eventos del sistema y permite su exportación sencilla a unos ficheros Excel o a una tabla de base de datos.

  - https://www.pgaudit.org/
  - **Enlace Git:** https://github.com/pgaudit/pgaudit

Auditoria basada en triggers



Esquema


OPERACIONES
INSERT, UPDATE y DELETE
Otra base de datos
TABLAS
ESCUCHAR MODIFICACIONES
EN LAS TABLAS
GRABAR UN
REGISTRO
Base de datos
Auditoria
Tabla_auditoria
Trigger
Function Trigger
DBLINK

BD para auditoria

```
-- tabla para controlar los cambios sobre la tabla contratos
CREATE TABLE tabla_auditoria (
 id_auditoria serial NOT NULL,
 nombre_tabla character(45) NOT NULL,
 operacion char(1) NOT NULL,
 valor_viejo text,
 valor_nuevo text,
 fecha_actual timestamp without time zone NOT NULL,
 usuario character(45) NOT NULL,
 CONSTRAINT id_auditoria_pk PRIMARY KEY (id_auditoria));
```

Activar la extensión **dblink**

Function

```
CREATE OR REPLACE FUNCTION funcion_auditoria() RETURNS trigger AS
$$
BEGIN
 IF (TG_OP = 'DELETE') THEN
   PERFORM dblink_connect('port=5432 dbname=auditoria user=postgres password=abril');
   PERFORM dblink_exec ('INSERT INTO tabla_auditoria (nombre_tabla, operacion,
   valor_viejo,    valor_nuevo, fecha_actual, usuario) VALUES ('''||TG_TABLE_NAME||''',
   '''||TG_OP||''', '''||OLD||''', NULL, now(), USER);');
   PERFORM dblink_disconnect();
   RETURN OLD;
 ELSIF (TG_OP = 'UPDATE') THEN
   PERFORM dblink_connect('port=5432 dbname=auditoria user=postgres password=abril');
   PERFORM dblink_exec ('INSERT INTO tabla_auditoria (nombre_tabla, operacion,
   valor_viejo,    valor_nuevo, fecha_actual, usuario) VALUES ('''||TG_TABLE_NAME||''',
   '''||TG_OP||''', '''||OLD||''', '''||NEW||''', now(), USER);');
   PERFORM dblink_disconnect();
   RETURN NEW;
 ELSIF (TG_OP = 'INSERT') THEN
   PERFORM dblink_connect('port=5432 dbname=auditoria user=postgres password=abril');
   PERFORM dblink_exec ('INSERT INTO tabla_auditoria (nombre_tabla, operacion,
   valor_viejo, valor_nuevo, fecha_actual, usuario) VALUES ('''||TG_TABLE_NAME||''',
   '''||TG_OP||''', NULL, '''||NEW||''', now(), USER);');
   PERFORM dblink_disconnect();
   RETURN NEW;
 END IF;
 RETURN NULL;
END;
$$
LANGUAGE 'plpgsql' VOLATILE;
```

PERFORM nos sirve para llamar a funciones que no devuelven nada y tampoco tenemos que almacenar nada.

Trigger

```
create trigger tbl_atributos_tg_audit
after insert or update or delete
on "Contrato"
for each row execute procedure funcion_auditar();
```

Fuentes

- PDF: Administración-bases-de-datos.pdf
- https://dbasinapuros.com/sistemas-para-auditar-bases-de-datos-postgresql/