Audit Trigger, for logging or Auditing your Database DML Transactions

If you experienced with SQL Server, You might know about SQL Server Audit and Change Data Capture (CDC) functionality.
As per the PostgreSQL WIKI, PostgreSQL does not have such an Inbuilt functionality for auditing your user transactions.

Most of the Application requires, Database auditing like Who Inserted, When Updated kind of DML Transactions.
Here, I am sharing one demonstration about Trigger Approach, which is not always good, but for specific requirement, we can go with this approach.

I have created separate schema and table to log about DML transactions. PostgreSQL Database Administrator can use this trigger for security purpose also.

Example:

Create a new table

```
CREATE TABLE employees(
   id INT GENERATED ALWAYS AS IDENTITY,
   first_name VARCHAR(40) NOT NULL,
   last_name VARCHAR(40) NOT NULL,
   PRIMARY KEY(id)
);
select * from employees;


create table department (id int, deptname varchar);
select * from department;
```

Create a new table for Logged Transaction:

```
CREATE TABLE tbl_LoggedTransactions
(
     SchemaName CHARACTER VARYING
     ,TableName CHARACTER VARYING
     ,UserName CHARACTER VARYING
     ,DMLAction CHARACTER VARYING
     ,OriginalData TEXT
     ,ExecutedNewData TEXT
     ,ExecutedSQL TEXT
     ,RecordDateTime TIMESTAMP WITHOUT TIME ZONE DEFAULT
NOW()
);
```

Create one trigger function for auditing DML transactions:

```
CREATE OR REPLACE FUNCTION trg_AuditDML()
RETURNS TRIGGER
AS $BODY$
DECLARE
    OldData TEXT;
    NewData TEXT;
BEGIN

    IF (TG_OP = 'UPDATE') THEN
    OldData := ROW(OLD.*);
    NewData := ROW(NEW.*);
    INSERT INTO tbl_LoggedTransactions
 (
      SchemaName
      ,TableName
      ,UserName
      ,DMLAction
      ,OriginalData
      ,ExecutedNewData
      ,ExecutedSQL
 )
    VALUES
 (
      TG_TABLE_SCHEMA::TEXT
      ,TG_TABLE_NAME::TEXT
      ,session_user::TEXT
      ,substring(TG_OP,1,1)
      ,OldData
      ,NewData
      ,current_query()
 );
    RETURN NEW;
ELSIF (TG_OP = 'DELETE') THEN
    OldData := ROW(OLD.*);
    INSERT INTO tbl_LoggedTransactions
 (
      SchemaName
      ,TableName
      ,UserName
      ,DMLAction
      ,OriginalData
      ,ExecutedSQL
 )
    VALUES
 (
      TG_TABLE_SCHEMA::TEXT
      ,TG_TABLE_NAME::TEXT
      ,session_user::TEXT
```

```
          ,substring(TG_OP,1,1)
          ,OldData
          ,current_query()
     );
        RETURN OLD;
    ELSIF (TG_OP = 'INSERT') THEN
        NewData := ROW(NEW.*);
        INSERT INTO tbl_LoggedTransactions
     (
         SchemaName
         ,TableName
         ,UserName
         ,DMLAction
         ,ExecutedNewData
         ,ExecutedSQL
     )
        VALUES
     (
         TG_TABLE_SCHEMA::TEXT
         ,TG_TABLE_NAME::TEXT
         ,session_user::TEXT
         ,substring(TG_OP,1,1)
         ,NewData
         ,current_query()
     );
        RETURN NEW;
    ELSE
        RAISE WARNING '[AuditTable.trg_AuditDML] - Other
action occurred: %, at %',TG_OP,now();
        RETURN NULL;
    END IF;
END;
$BODY$
LANGUAGE plpgsql;
```

Create dynamic trigger on all tables.

```
CREATE OR REPLACE FUNCTION public.
dynamic_master_data(source VARCHAR)
 RETURNS VARCHAR
 LANGUAGE plpgsql
AS $function$
DECLARE
    tablename    VARCHAR;
    dynamicQuery VARCHAR;
BEGIN
  dynamicQuery ='CREATE TRIGGER Audit_Log BEFORE INSERT OR
UPDATE OR DELETE ON ' || source ||
' FOR EACH ROW EXECUTE PROCEDURE trg_AuditDML()';
EXECUTE dynamicQuery;
    RETURN 'success';
END;
$function$
;
```

Execute few sample DMLs:

```
select * from dynamic_master_data ('department');
select * from dynamic_master_data ('employeesss');

select * from department;
insert into department values(2,'Database');
```

```
delete from department where id=2;


INSERT INTO employeesss (first_name, last_name)
VALUES ('bibek', 'mahatara');
```

Check AuditTable.tbl_LoggedTransactions table for logged DML transaction:

```
select * from tbl_loggedtransactions tl ;
```

To show the list of Trigger in database:

```
SELECT event_object_table AS table_name ,trigger_name
FROM information_schema.triggers GROUP BY table_name ,
trigger_name ORDER BY table_name ,trigger_name ;
```

Drop trigger in database:

```
Drop trigger tt on locations;
Drop trigger tt on employees;
Drop trigger tt on departments;
```