PostgreSQL – AutoAudit
Installation and Usage Manual for Automated Database Auditing

This manual aims to explain and illustrate the installation and use of the PostgreSQL extension called “AutoAudit”, which functions as a mechanism to record insert, update, and delete operations on all the tables in the database where it is installed.

It is important to note that the process shown here was performed on a computer running Windows.

The audit record includes:

- Unique event identifier
- Operation type (INSERT, UPDATE, DELETE)
- Name of the affected table
- Exact date and time of the event
- User/role executing the operation
- Client IP address
- Previous state of the data (before modification)
- New state of the data (after modification)

Visual example of a record:

	event_id [PK] bigint	operation_type text	table_name text	event_time timestamp with time zone	executed_by text	client_ip inet	old_data jsonb	new jso
1	1	INSERT	products	2025-09-17 18:51:24.216113-06	postgres	::1	[null]	{"n
2	2	UPDATE	products	2025-09-17 18:51:24.216113-06	postgres	::1	{"name": "TestPhone X", "brand": "TestBrand", "price": 499.99, "category": "cellphone", "product_id": 30}	{"n
3	4	INSERT	products	2025-09-17 18:51:30.655282-06	Worker	::1	[null]	{"n
4	5	UPDATE	products	2025-09-17 18:51:30.655282-06	Worker	::1	{"name": "TestPhone X", "brand": "TestBrand", "price": 499.99, "category": "cellphone", "product_id": 31}	{"n
5	6	DELETE	products	2025-09-17 18:51:30.655282-06	Worker	::1	{"name": "TestPhone X", "brand": "TestBrand", "price": 459.99, "category": "cellphone", "product_id": 31}	[nu

new_data jsonb
{"name": "TestPhone X", "brand": "TestBrand", "price": 499.99, "category": "cellphone", "product_id": 30}
{"name": "TestPhone X", "brand": "TestBrand", "price": 459.99, "category": "cellphone", "product_id": 30}
{"name": "TestPhone X", "brand": "TestBrand", "price": 499.99, "category": "cellphone", "product_id": 31}
{"name": "TestPhone X", "brand": "TestBrand", "price": 459.99, "category": "cellphone", "product_id": 31}
[null]

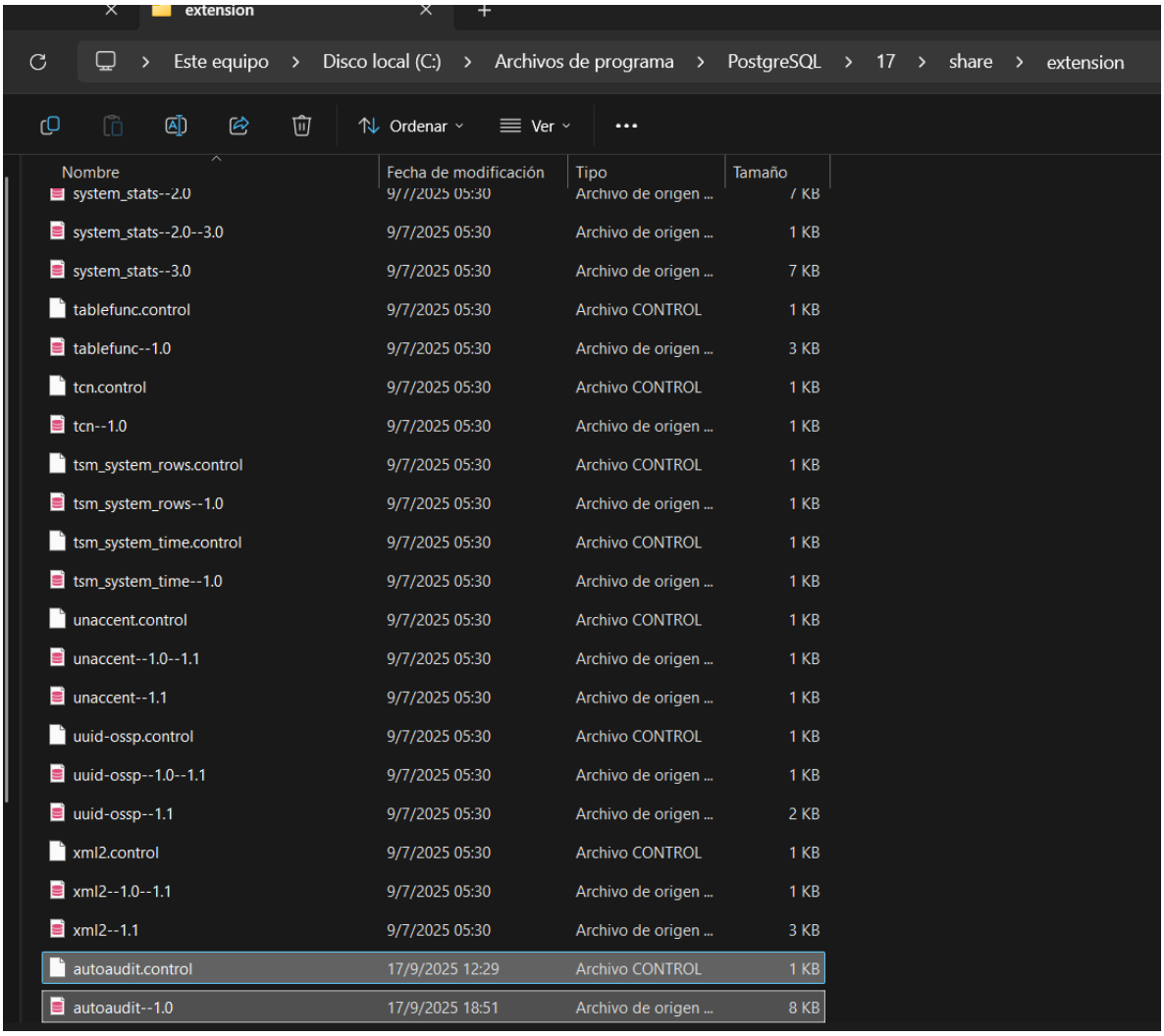
Installation

The following two files are required (found in the *autoaudit* folder):autoaudit.control

- autoaudit--1.0

Nombre	Fecha de modificación	Tipo	Tamaño
autoaudit.control	17/9/2025 12:29	Archivo CONTROL	1 KB
autoaudit--1.0	17/9/2025 18:52	Archivo de origen ...	8 KB

Once downloaded, copy them into the directory where PostgreSQL stores its extensions, which by default is: C:\Program Files\PostgreSQL\17\share\extension



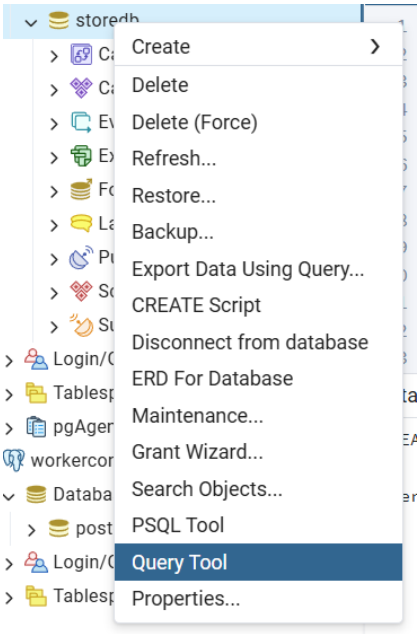
Note: this directory path may vary if it was customized when PostgreSQL was installed.

In this directory you must place the previously mentioned autoaudit files.

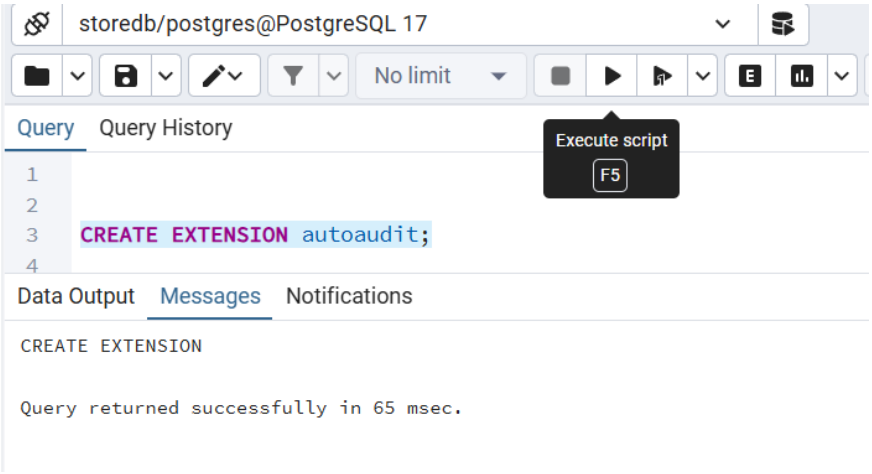
It is important to highlight that, from this point, the installation process must be repeated for each database where you want to use the extension. However, it is a very straightforward process.

Once the files are in that directory, log into PostgreSQL using the postgres superuser account (the login role with full administrative privileges; this is also the only role that Will be allowed to delete logs).

Go to the target database where you want to install the extension and open its Query Tool:



Execute the following command: **CREATE EXTENSION autoaudit;**



That's it — the extension will be active. From this moment on, every time a user inserts, updates, or deletes rows in the database tables, the operation will be added to the audit log.

Usage

To query the log, simply execute the following command (always from the postgres superuser role) in the database where the extension was installed:

select * from autoaudit.audit_log

Showing rows: 1 to 3 Page No: 1 of 1

	event_id [PK] bigint	operation_type text	table_name text	event_time timestamp with time zone	executed_by text	client_ip inet	old_data jsonb
1	1	INSERT	products	2025-09-17 17:46:22.761144-06	postgres	::1	[null]
2	2	UPDATE	products	2025-09-17 17:46:22.761144-06	postgres	::1	{"name": "TestPhone X", "brand": "TestBrand", "price": 499.99, "category": "cellphone", "product_id": 11}
3	3	DELETE	products	2025-09-17 17:46:22.761144-06	postgres	::1	{"name": "TestPhone X", "brand": "TestBrand", "price": 459.99, "category": "cellphone", "product_id": 11}

…

client_ip inet	old_data jsonb	new_data jsonb
::1	[null]	{"name": "TestPhone X", "brand": "TestBrand", "price": 499.99, "category": "cellphone", "product_id": 11}
::1	{"name": "TestPhone X", "brand": "TestBrand", "price": 499.99, "category": "cellphone", "product_id": 11}	{"name": "TestPhone X", "brand": "TestBrand", "price": 459.99, "category": "cellphone", "product_id": 11}
::1	{"name": "TestPhone X", "brand": "TestBrand", "price": 459.99, "category": "cellphone", "product_id": 11}	[null]

This table is located within the autoaudit schema of the database, and its name is audit_log.
Only the postgres superuser, who created it, can modify or delete this schema and table.

Naturally, regular users who will be working with the database will have their own permissions and restrictions for normal database use (those permissions are outside the scope of this manual).

What is important here is to demonstrate that a user without modification permissions on autoaudit cannot make manual changes to the audit table.

(For these examples, we created a user Worker with permissions only to modify tables within the public schema.)

Query Query History

```
--Showing this user can't modify the autoaudit log
-- Insert log into audit table
   INSERT INTO autoaudit.audit_log (
       operation_type,
       table_name,
       executed_by,
       client_ip,
       old_data,
       new_data
   )
   VALUES (
      'INSERT',     -- INSERT, UPDATE, DELETE
      'products',       -- table affected
       current_user,        -- user executing
       '127.0.0.1'::inet, -- client IP
       NULL,
       NULL
   );
```

Data Output Messages Notifications

```
ERROR:  permiso denegado al esquema autoaudit
LINE 4:     INSERT INTO autoaudit.audit_log (
                        ^

SQL state: 42501
Character: 100
```

Just as users without permissions cannot manually insert records into the audit table, they also cannot delete or modify rows in it.

On the other hand, it is demonstrated that when such roles/users make changes in the base tables of the database, those operations are correctly recorded in the audit log.

storedb/Worker@workerconnection

No limit

Query | Query History

```sql
-- Insert a new product for testing
INSERT INTO products (name, category, brand, price)
VALUES ('TestPhone X', 'cellphone', 'TestBrand', 499.99);
-- Update that product
UPDATE products
SET price = 459.99
WHERE name = 'TestPhone X';
-- Delete that product
DELETE FROM products
WHERE name = 'TestPhone X';
```

Data Output | Messages | Notifications

```
DELETE 1

Query returned successfully in 67 msec.
```

Showing

	event_id [PK] bigint	operation_type text	table_name text	event_time timestamp with time zone	executed_by text	client_ip inet	old_data jsonb
1	1	INSERT	products	2025-09-17 18:51:24.216113-06	postgres	::1	[null]
2	2	UPDATE	products	2025-09-17 18:51:24.216113-06	postgres	::1	{"name": "TestPhone X", "
3	3	DELETE	products	2025-09-17 18:51:24.216113-06	postgres	::1	{"name": "TestPhone X", "
4	4	INSERT	products	2025-09-17 18:51:30.655282-06	Worker	::1	[null]
5	5	UPDATE	products	2025-09-17 18:51:30.655282-06	Worker	::1	{"name": "TestPhone X", "
6	6	DELETE	products	2025-09-17 18:51:30.655282-06	Worker	::1	{"name": "TestPhone X", "

Finally, we can see that the postgres superuser (who has full privileges) can delete or modify records in the audit log — while other restricted users cannot.

```
--this user can delete things from the audit, other users can't
delete from autoaudit.audit_log where event_id = 3
select * from autoaudit.audit_log
```

Data Output | Messages | Notifications

```
DELETE 1

Query returned successfully in 125 msec.
```

	event_id [PK] bigint	operation_type text	table_name text	event_time timestamp with time zone	executed_by text	client_ip inet
1	1	INSERT	products	2025-09-17 18:51:24.216113-06	postgres	::1
2	2	UPDATE	products	2025-09-17 18:51:24.216113-06	postgres	::1
3	4	INSERT	products	2025-09-17 18:51:30.655282-06	Worker	::1

For example, in the demonstration, audit record #3 has been deleted by the superuser.

This ensures that other users cannot tamper with this critical audit history.

You now have everything necessary to install this PostgreSQL extension and view the records of this automated auditing system!