

Manual de Instalación y Configuración del Proyecto Final de Bases de Datos

Proyecto: Vibesia

Versión: 2.0

Fecha: 16/6/2025

Equipo: Ad Astra

Elaborado por:

- Oscar Alejandro Prasca Chacón
- Carlos Julio Vergel Wilches
- Karen Silvana Duque Leal
- Duvan Arley Ramírez Duran
- Juan David Jaimes Rojas

Manual de Instalación y Configuración del Proyecto Final de Bases de Datos	1
1. Introducción	2
2. Instalar PostgreSQL 17	2
2.1 Descargar el instalador:	2
2.2 Ejecutar el instalador:	2
7.3 Configurar y Finalizar la instalación:	3
3. Pre-requisitos	4
4. Estructura del Repositorio	5
5. Creación de la Base de Datos	5
Paso 5.1	5
Paso 5.2	6
Paso 5.3	7
Paso 5.4	7
Paso 5.5:	8
6. Verificación de la Instalación	9
6.1. Conexión y Estructura	9

6.2. Scripts de Triggers	9
6.3. Verificación de Triggers de Auditoría	11
7. Resolución de Problemas Comunes	13

1. Introducción

En este documento se describe el proceso completo de instalación y configuración de la base de datos "musicdb", desarrollada como proyecto final del curso de Bases de Datos. Este incluye los pasos necesarios para levantar el entorno, ejecutar scripts SQL y verificar que todo funcione correctamente, incluyendo auditoría con triggers.

2. Instalar PostgreSQL 17

Para descargar e instalar PostgreSQL realizar los siguientes pasos:

2.1 Descargar el instalador:

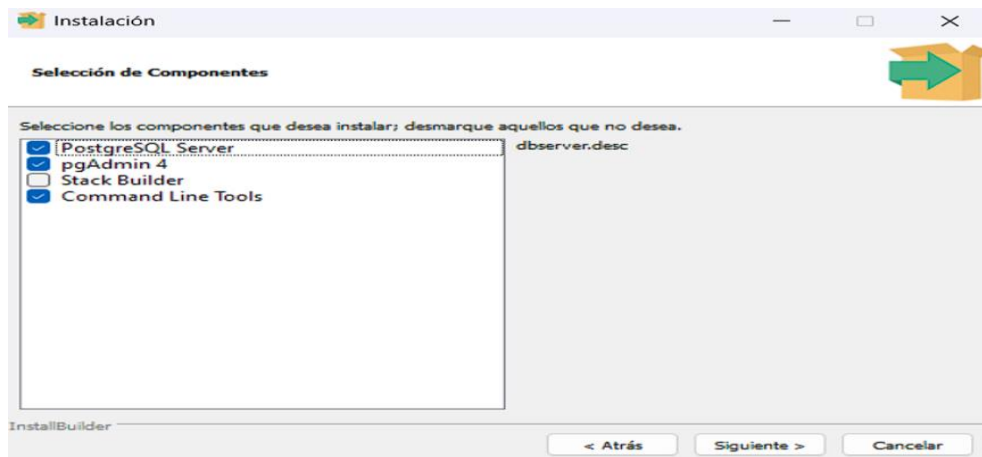
Visita la página oficial de PostgreSQL [EnterpriseDB](https://www.enterprisedb.com) y descarga el instalador para la versión 17 para Windows.

2.2 Ejecutar el instalador:

Abre el archivo .exe descargado y sigue las indicaciones del asistente de instalación.

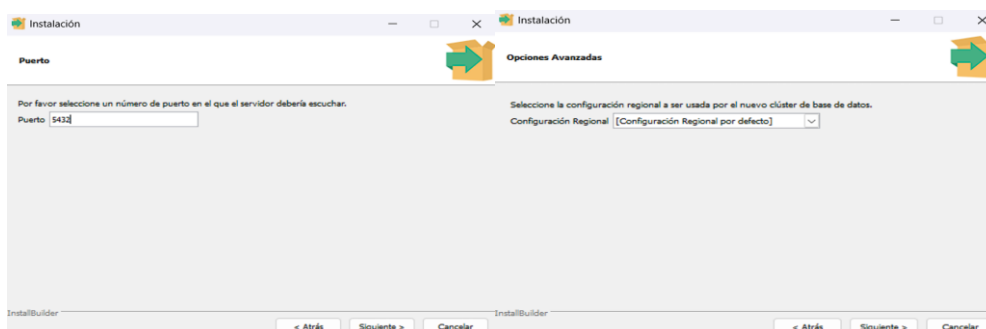
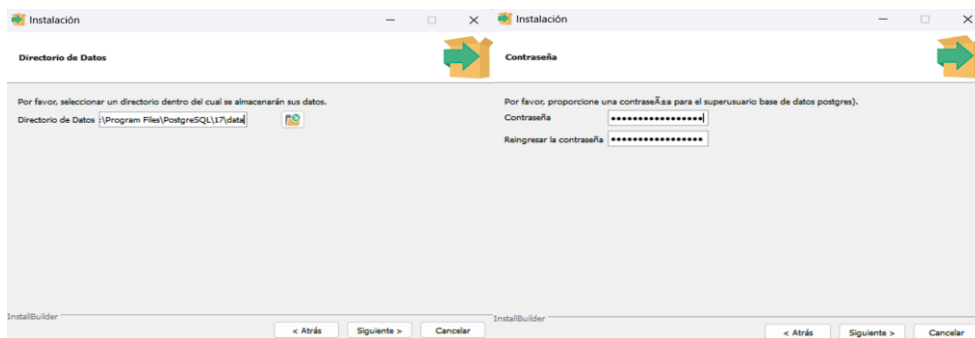


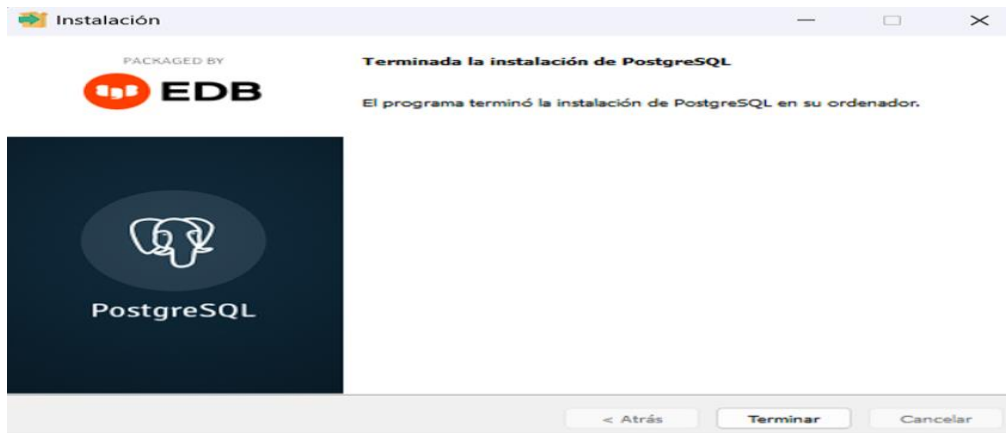
- Seleccionar los componentes (PostgreSQL Server, pgAdmin 4 y herramientas de línea de comandos).



7.3 Configurar y Finalizar la instalación:

- Elige el directorio de instalación.
- Especifica la carpeta donde se almacenarán los datos.
- Introduce una contraseña para el usuario superusuario postgres .
- Configura el puerto (por defecto es 5432).
- Elegir la configuración regional local.
- Completa el proceso y, si lo deseas, ejecuta pgAdmin 4 para conectarte al servidor y gestionar las bases de datos de manera gráfica.





3. Pre-requisitos

- PostgreSQL: Versión 17
- PgAdmin 4 instalado
- Sistema operativo: Se ha utilizado Windows 11
- Acceso a un usuario PostgreSQL con privilegios de creación de base de datos
- Carpeta del proyecto “ProyectoFinal-BD” con la siguiente estructura:

ProyectoFinal-BD/

├─ sql/

| └─ ddl/ # Scripts de definición (usuario, BD, tablas, triggers)

| | └─ 01-create-database.sql

| | └─ 02-create-tables.sql

| | └─ 03-create-triggers.sql

| | └─ (04-alter-tables.sql)

| └─ dml/ # Scripts de manipulación de datos

| └─ data/ # Datos de prueba iniciales (01-genres.sql, etc.)

| └─ Auditoria/ # Ejemplos para probar/consultar auditoría

└─ docs/ # Documentación técnica adicional (Diagramas, Diccionario, etc.)

└─ README.md # README principal del proyecto

4. Estructura del Repositorio

Carpeta	Descripción
sql/ddl/	Contiene todos los scripts SQL para la Definición de Datos (DDL). Esto incluye la creación del usuario de la aplicación, la base de datos, el esquema, todas las tablas, sus relaciones, índices y los triggers de auditoría.
sql/dml/	Contiene los scripts SQL para la Manipulación de Datos (DML), específicamente para la carga inicial de datos de prueba en las tablas.
sql/dml/Auditoria/	Incluye scripts de ejemplo para consultar y verificar el funcionamiento de la tabla de auditoría.
docs/	Alberga la documentación técnica completa del proyecto, incluyendo diagramas de modelos, diccionario de datos, proceso de normalización y manuales detallados.

5. Creación de la Base de Datos

Nota Importante: Se asume una instalación limpia.

Paso 5.1: Creación de Usuario, Base de Datos y Esquema

1. Conéctese como superusuario postgres a su servidor PostgreSQL (usando psql o pgAdmin).

```
SQL Shell (psql)
Server [localhost]:
Database [postgres]:
Port [5432]:
Username [postgres]:
Contraseña para usuario postgres:

psql (17.4)
ADVERTENCIA: El código de página de la consola (850) difiere del código
de página de Windows (1252).
Los caracteres de 8 bits pueden funcionar incorrectamente.
Vea la página de referencia de psql «Notes for Windows users»
para obtener más detalles.
Digite «help» para obtener ayuda.
```

2. Abra el archivo sql/ddl/01-create-database.sql para el usuario music_admin

```
1  -- *****
2  -- #          MUSICDB DATABASE CREATION SCRIPT          #
3  -- *****
4
5  -- 01. Create user
6  CREATE USER music_admin WITH PASSWORD '*****';
7  -- 02. Create database (with ENCODING= 'UTF8', TEMPLATE=template 0, OWNER= fc_admin)
8  CREATE DATABASE musicdb WITH ENCODING='UTF8' LC_COLLATE='es_CO.utf-8' LC_CTYPE='es_CO.utf-8' TEMPLATE=template0 OWNER = music_admin;
9  -- 03. Grant privileges
10 GRANT ALL PRIVILEGES ON DATABASE musicdb TO music_admin;
11 -- 04. Create Schema
12 CREATE SCHEMA IF NOT EXISTS vibesia_schema AUTHORIZATION music_admin;
13 -- 05. Comment on database
14 COMMENT ON DATABASE musicdb IS 'Base de datos para el sistema';
15 -- 06. Comment of schema
16 COMMENT ON SCHEMA vibesia_schema IS 'Esquema principal para el sistema';
```

3. Ejecute el contenido completo de sql/ddl/01-create-database.sql, uno por uno. Esto creará el usuario music_admin, la base de datos musicdb (owner: music_admin).

```
postgres=# CREATE USER music_admin WITH PASSWORD '*****';
CREATE ROLE
postgres=# CREATE DATABASE musicdb WITH ENCODING='UTF8' LC_COLLATE='es_CO.utf-8' LC_CTYPE='es_CO.utf-8' TEMPLATE=template0 OWNER = music_admin;
CREATE DATABASE
postgres=# GRANT ALL PRIVILEGES ON DATABASE musicdb TO music_admin;
GRANT
```

4. Conéctese a la base de datos musicdb.

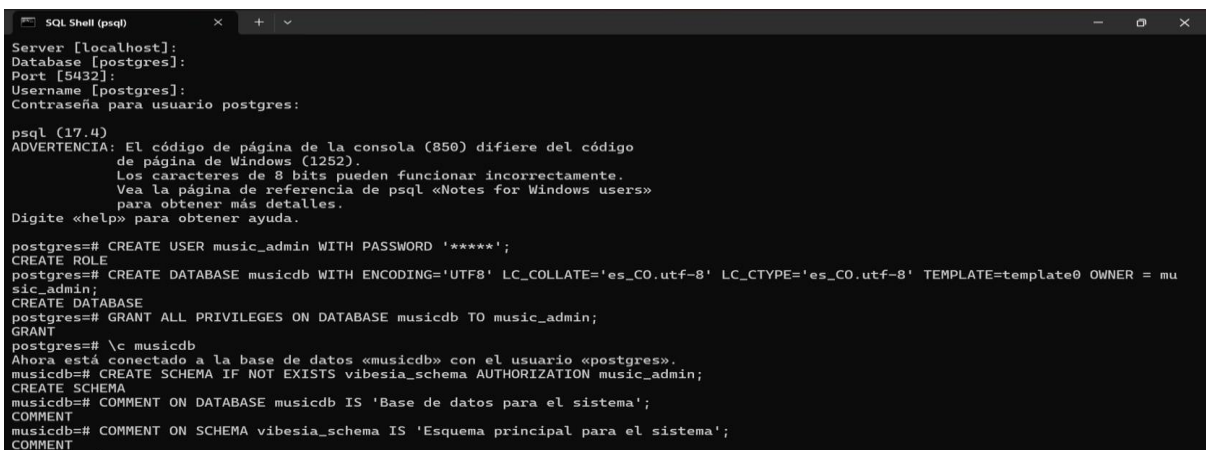
En psql: \c musicdb

5. Ejecute la creación del esquema principal:

En psql:

```
CREATE SCHEMA IF NOT EXISTS vibesia_schema AUTHORIZATION music_admin;
```

Resultados:



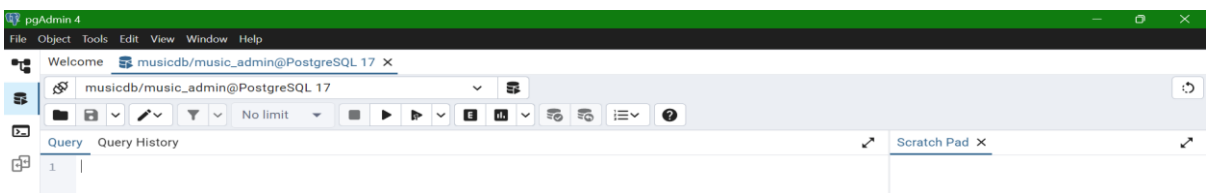
```
SQL Shell (psql)
Server [localhost]:
Database [postgres]:
Port [5432]:
Username [postgres]:
Contraseña para usuario postgres:

psql (17.4)
ADVERTENCIA: El código de página de la consola (850) difiere del código
de página de Windows (1252).
Los caracteres de 8 bits pueden funcionar incorrectamente.
Vea la página de referencia de psql «Notes for Windows users»
para obtener más detalles.
Digite «help» para obtener ayuda.

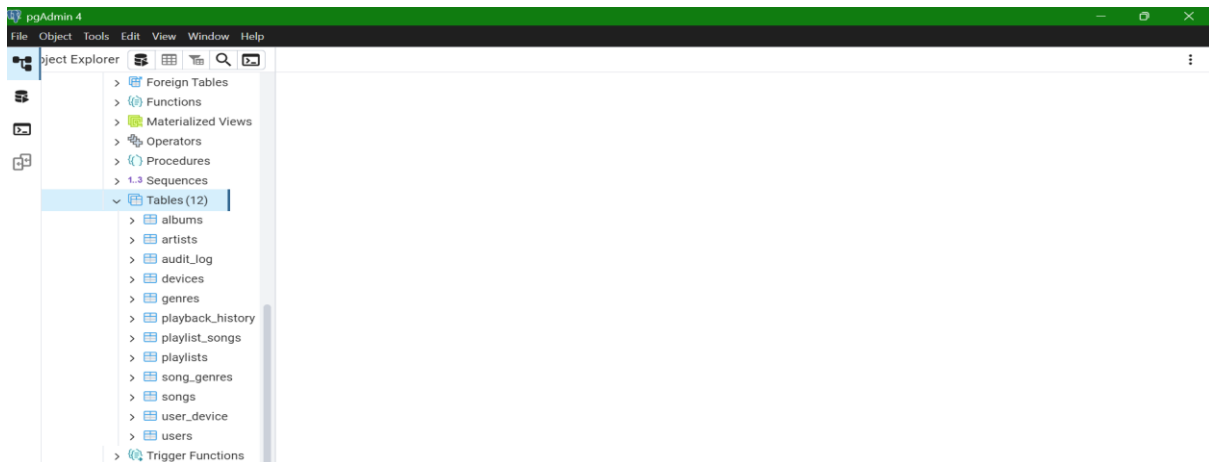
postgres=# CREATE USER music_admin WITH PASSWORD '*****';
CREATE ROLE
postgres=# CREATE DATABASE musicdb WITH ENCODING='UTF8' LC_COLLATE='es_CO.utf-8' LC_CTYPE='es_CO.utf-8' TEMPLATE=template0 OWNER = music_admin;
CREATE DATABASE
postgres=# GRANT ALL PRIVILEGES ON DATABASE musicdb TO music_admin;
GRANT
postgres=# \c musicdb
Ahora está conectado a la base de datos «musicdb» con el usuario «postgres».
musicdb=# CREATE SCHEMA IF NOT EXISTS vibesia_schema AUTHORIZATION music_admin;
CREATE SCHEMA
musicdb=# COMMENT ON DATABASE musicdb IS 'Base de datos para el sistema';
COMMENT
musicdb=# COMMENT ON SCHEMA vibesia_schema IS 'Esquema principal para el sistema';
COMMENT
```

Paso 5.2: Creación de Tablas y Estructura

1. Conéctese a “musicdb” como el usuario “music_admin”



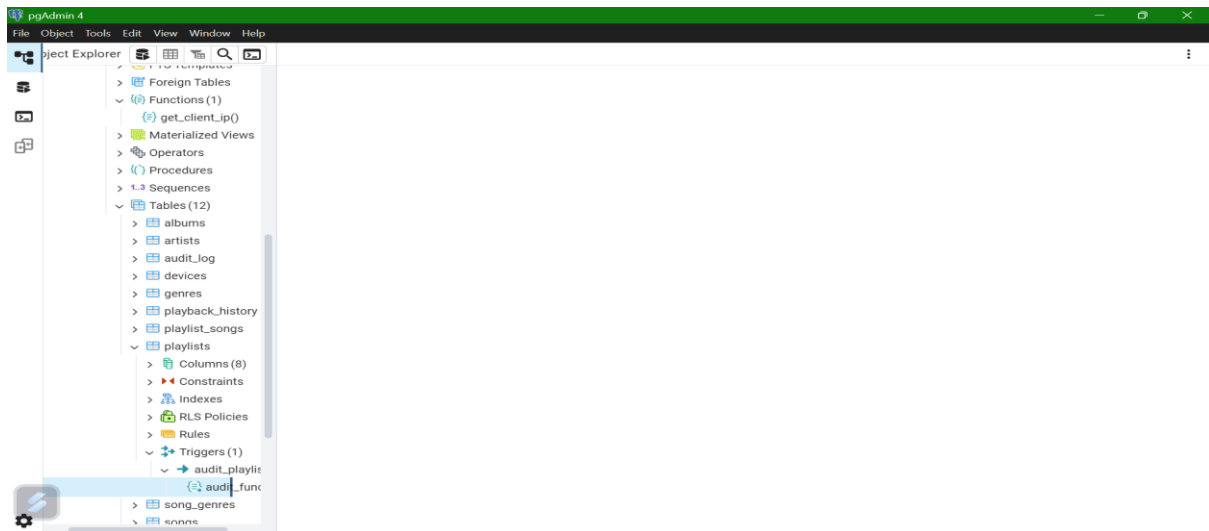
2. Ejecute el contenido completo del archivo sql/ddl/02-create-tables.sql. Esto creará todas las tablas, relaciones, índices y aplicará alteraciones.



"Tras ejecutar 02-create-tables.sql, todas las tablas, relaciones e índices son creados en vibesia_schema."

Paso 5.3: Creación de Triggers de Auditoría

1. Aún conectado como "music_admin a musicdb".
2. Ejecute el contenido completo del archivo sql/ddl/03-create-triggers.sql.
(Nota: Puede ocurrir un error si se intenta crear un trigger para una tabla 'ratings' no existente; esto puede ser ignorado si no es parte del diseño).



"El script 03-create-triggers.sql define las funciones de auditoría y los aplica a las tablas designadas."

Paso 5.4: Carga de Datos Iniciales (DML)

1. Aún conectado como "music_admin" a "musicdb".
2. Ejecute los scripts de la carpeta sql/dml/data/ en orden numérico (ej. 01-genres.sql, 02-artists.sql, etc.).

01-genres	31/05/2025 4:51 p. m.	Archivo de origen SQL	3 KB
02-artists	31/05/2025 4:51 p. m.	Archivo de origen SQL	3 KB
03-albums	31/05/2025 4:51 p. m.	Archivo de origen SQL	3 KB
04-music	31/05/2025 4:51 p. m.	Archivo de origen SQL	7 KB
05-song-genres	31/05/2025 4:51 p. m.	Archivo de origen SQL	3 KB
06-users	31/05/2025 4:51 p. m.	Archivo de origen SQL	2 KB
07-devices	31/05/2025 4:51 p. m.	Archivo de origen SQL	1 KB
08-user-device	31/05/2025 4:51 p. m.	Archivo de origen SQL	2 KB
09-playlists	31/05/2025 4:51 p. m.	Archivo de origen SQL	3 KB
10-playlist-song	31/05/2025 4:51 p. m.	Archivo de origen SQL	4 KB
11-reproductions	31/05/2025 4:51 p. m.	Archivo de origen SQL	4 KB

Nota: Se recomienda copiar y pegar el contenido de cada archivo en la herramienta de consulta.

Paso 5.5: Inserción de los índices

1. Aún conectado como “music_admin” a “musicdb”.
2. Ejecute el contenido completo del archivo sql/ddl/05-create-indexes.sql.

```

1  -- #####
2  -- # SCRIPT 05: SPECIALIZED INDEXES FOR PERFORMANCE OPTIMIZATION #
3  -- # Project: Vibesia Music Platform #
4  -- # Purpose: Defines functional, composite, and partial indexes #
5  -- #           to enhance query performance for both common application #
6  -- #           operations and complex analytical workloads. #
7  -- # Note: This script is intended to be run AFTER the main DDL #
8  -- #       scripts (01-03) have been executed and an initial #
9  -- #       database schema alignment with backend models (e.g., #
10 -- #       via Alembic) has been performed. #
11 -- #####
12
13 -- #####
14 -- # FUNCTIONAL INDEXES (Case-Insensitive) #
15 -- #####
16 -- For efficient case-insensitive searches.
17 -- Backend queries should use LOWER(column_name) = LOWER('search_term') to leverage these.
18
19 CREATE INDEX IF NOT EXISTS idx_users_username_lower ON vibesia_schema.users (LOWER(username));
20 CREATE INDEX IF NOT EXISTS idx_users_email_lower ON vibesia_schema.users (LOWER(email));
21 CREATE INDEX IF NOT EXISTS idx_artists_name_search ON vibesia_schema.artists (LOWER(name));
22 CREATE INDEX IF NOT EXISTS idx_albums_title_search ON vibesia_schema.albums (LOWER(title));
23 CREATE INDEX IF NOT EXISTS idx_songs_title_search ON vibesia_schema.songs (LOWER(title));
24 CREATE INDEX IF NOT EXISTS idx_genres_name_lower ON vibesia_schema.genres (LOWER(name));
25 CREATE INDEX IF NOT EXISTS idx_playlists_name_lower ON vibesia_schema.playlists (LOWER(name));
26
27 -- #####
28 -- # INDEXES FOR ANALYTICAL QUERIES #
29 -- #####

```

Nota: "El script 05-create-indexes.sql introduce un conjunto de índices especializados, incluyendo índices funcionales para búsquedas insensibles a mayúsculas/minúsculas, e índices compuestos y parciales diseñados para acelerar tanto operaciones comunes de la aplicación como consultas analíticas complejas. Estos índices son complementarios a los creados durante la definición inicial de las tablas y tienen como

objetivo mejorar la eficiencia general y los tiempos de respuesta de la base de datos Vibesia.”

6. Verificación de la Instalación

6.1. Conexión y Estructura

1. Conéctese a “musicdb” como “music_admin” y verifique la existencia de las tablas en el esquema “vibesia_schema” (ej. usando \dt vibesia_schema.* en psql)

6.2. Scripts de Triggers

Ejecute la siguiente consulta para un resumen rápido del conteo de registros:

```
```sql
SELECT 'users' AS tabla, COUNT(*) AS total FROM
vibesia_schema.users

UNION ALL

SELECT 'songs', COUNT(*) FROM
vibesia_schema.songs

UNION ALL

SELECT 'artists', COUNT(*) FROM
vibesia_schema.artists

UNION ALL

SELECT 'albums', COUNT(*) FROM
vibesia_schema.albums

UNION ALL

SELECT 'genres', COUNT(*) FROM
vibesia_schema.genres
```

```
UNION ALL

SELECT 'playlists', COUNT(*) FROM

vibesia_schema.playlists

UNION ALL

SELECT 'devices', COUNT(*) FROM

vibesia_schema.devices

UNION ALL

SELECT 'playback_history', COUNT(*) FROM

vibesia_schema.playback_history

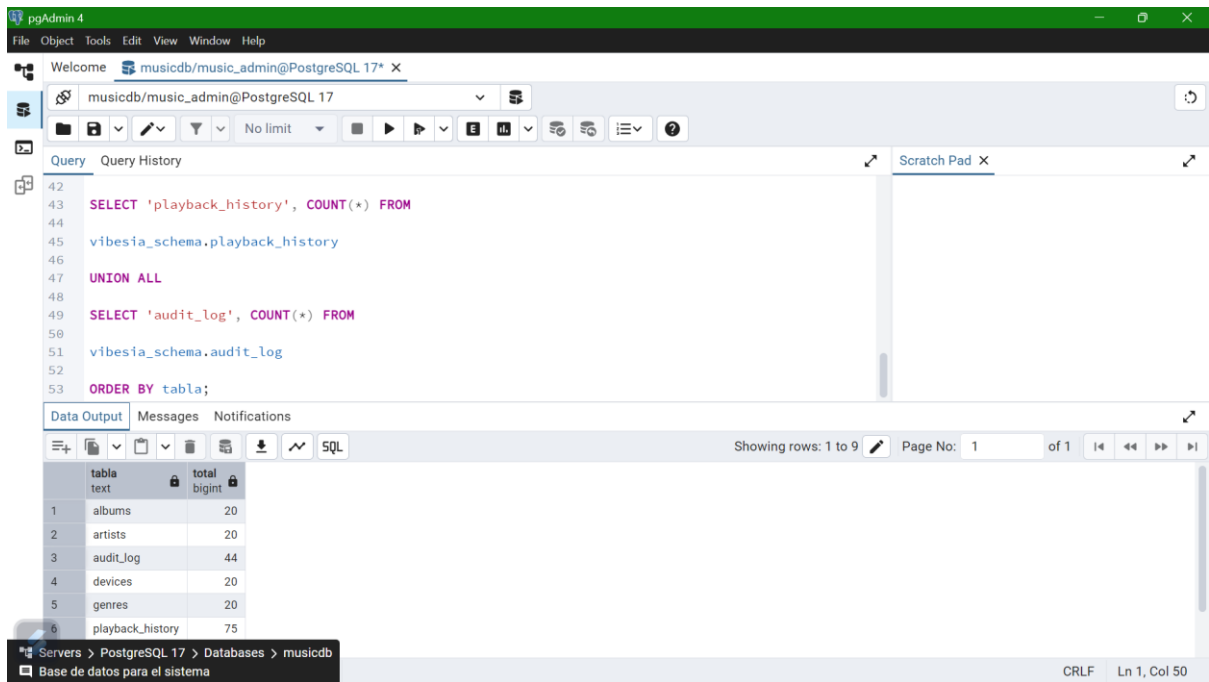
UNION ALL

SELECT 'audit_log', COUNT(*) FROM

vibesia_schema.audit_log

ORDER BY tabla;
```

Resultado:



### 6.3. Verificación de Triggers de Auditoría

Realice las siguientes operaciones de prueba y verifique la tabla audit\_log:

1. Insertar un usuario de prueba:

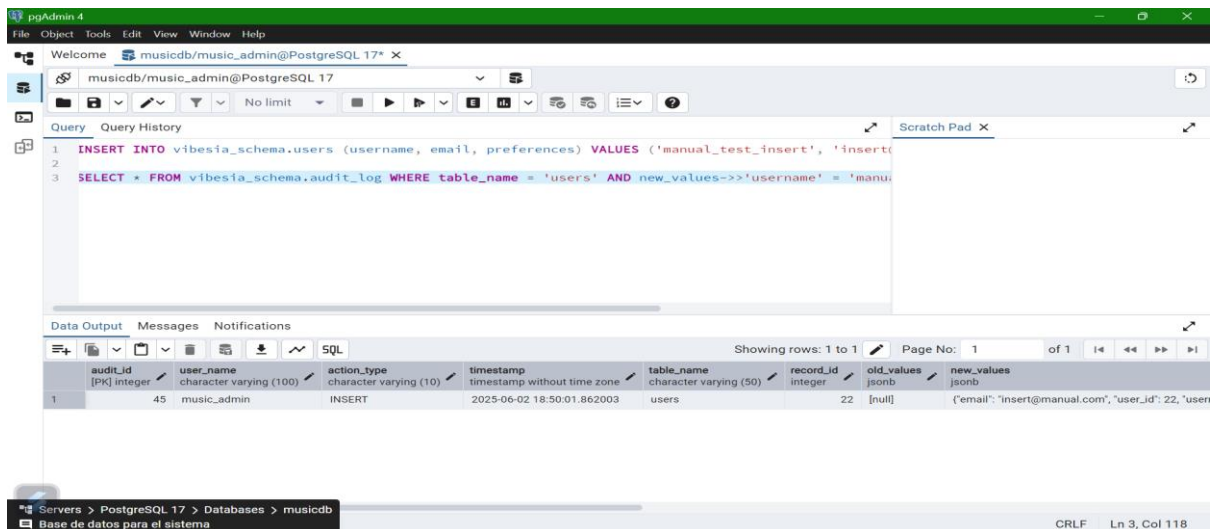
```

INSERT INTO vibesia_schema.users (username, email) VALUES
('test_audit_user', 'testaudit@example.com');

SELECT * FROM vibesia_schema.audit_log WHERE table_name = 'users' AND
new_values->>'username' = 'manual_test_insert';

```

Resultado:

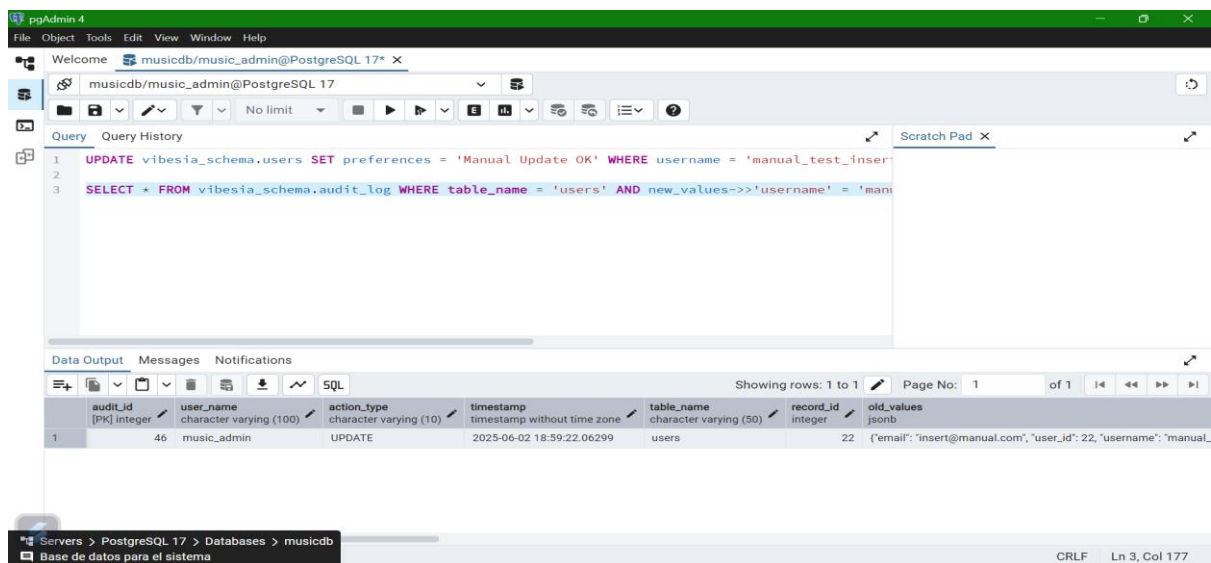


## 2. Actualizar el usuario de prueba:

```
UPDATE vibesia_schema.users SET preferences = 'Manual Update OK' WHERE
username = 'manual_test_insert';
```

```
SELECT * FROM vibesia_schema.audit_log WHERE table_name = 'users' AND
new_values->>'username' = 'manual_test_insert' AND action_type = 'UPDATE'
ORDER BY audit_id DESC LIMIT 1;
```

Resultado:



The screenshot shows the pgAdmin 4 interface. The top pane displays two SQL queries: an UPDATE statement and a SELECT statement. The bottom pane shows the results of the SELECT query in a table format.

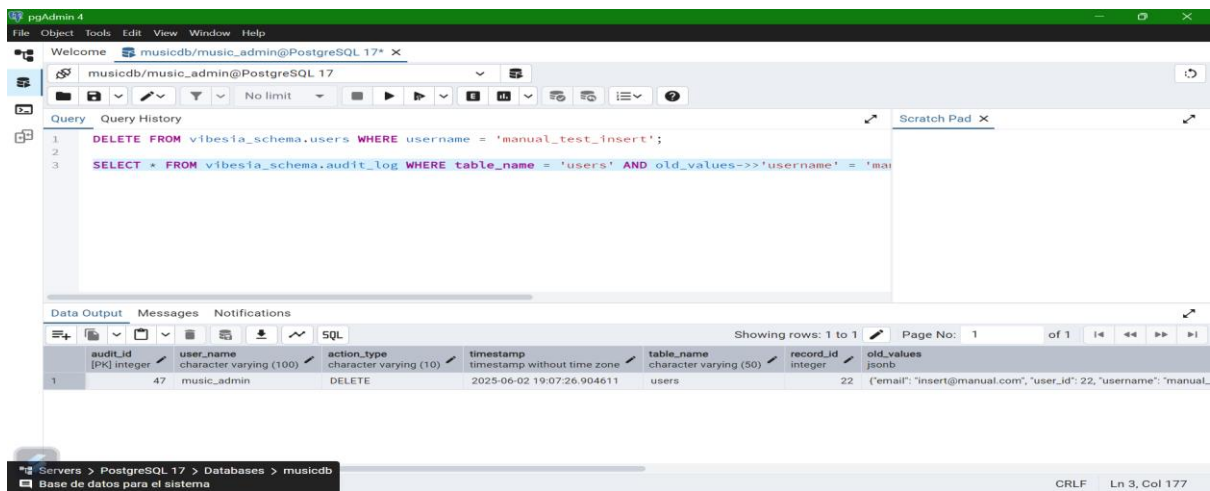
audit_id	user_name	action_type	timestamp	table_name	record_id	old_values
46	music_admin	UPDATE	2025-06-02 18:59:22.06299	users	22	{"email": "insert@manual.com", "user_id": 22, "username": "manual_test_insert"}

## 3. Eliminar el usuario de prueba:

```
DELETE FROM vibesia_schema.users WHERE username = 'manual_test_insert';
```

```
SELECT * FROM vibesia_schema.audit_log WHERE table_name = 'users' AND
old_values->>'username' = 'manual_test_insert' AND action_type = 'DELETE'
ORDER BY audit_id DESC LIMIT 1;
```

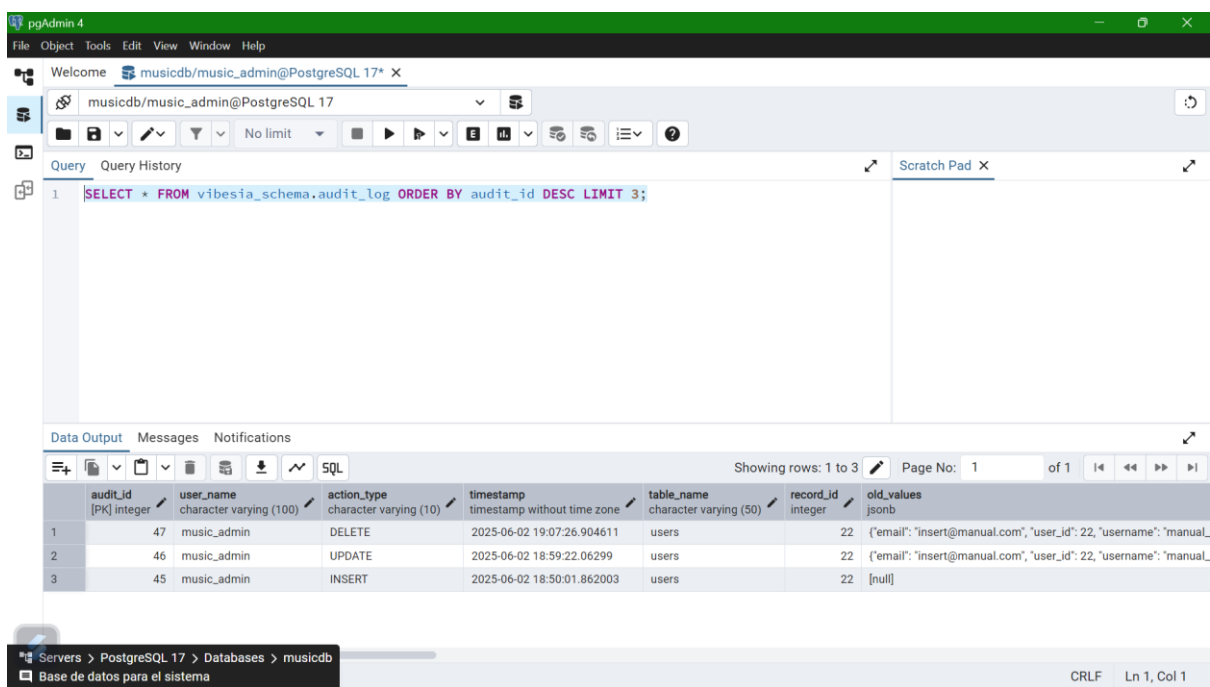
Resultado:



#### 4. Verificar el log de auditoría:

`SELECT * FROM vivesia_schema.audit_log ORDER BY audit_id DESC LIMIT 3;`

Se espera ver registros de INSERT, UPDATE y DELETE para ' manual\_test\_insert ', con music\_admin como user\_name en la tabla audit\_log.



## 7. Resolución de Problemas Comunes

Problema	Solución
----------	----------

"relation does not exist"	Hay que asegurar que los scripts DDL (especialmente 02-create-tables.sql) se ejecutaron correctamente y en orden.
"permission denied"	Verificar que se está conectado con el usuario correcto (postgres para crear BD/usuario, music_admin para crear tablas/triggers y manipular datos).
Triggers no funcionan	Hay que confirmar que 03-create-triggers.sql se ejecutó sin errores para las tablas deseadas.

**Fin del manual**