НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені Ігоря Сікорського»
ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ
**Кафедра системного програмування та спеціалізованих комп'ютерних систем**

**Лабораторна робота №1**

з дисципліни **Бази даних і засоби управління**

на тему: "Проектування бази даних та ознайомлення з базовими операціями СУБД PostgreSQL"

Виконала:
студентка групи КВ-01
Соліна Т. А.

Київ – 2022

*Метою роботи* є здобуття вмінь проектування бази даних та практичних навичок створення реляційних баз даних за допомогою PostgreSQL.

*Завдання* роботи полягає у наступному:

1. Розробити модель «сутність-зв’язок» предметної галузі, обраної студентом самостійно, відповідно до пункту «Вимоги до ER-моделі».
2. Перетворити розроблену модель у схему бази даних (таблиці) PostgreSQL.
3. Виконати нормалізацію схеми бази даних до третьої нормальної форми (3НФ).
4. Ознайомитись із інструментарієм PostgreSQL та pgAdmin 4 та внести декілька рядків даних у кожну з таблиць засобами pgAdmin 4.

*Зміст звіту*

1 Опис проблемного середовища;
2 Концептуальна модель предметної області;
3 Логічна модель (схема) даних БД;
4 Склад СУБД PostgreSQL;
5 Список обмежень цілісності в термінах СУБД PostgreSQL;
6 Фізична модель (схема) даних БД в pgAdmin III;
7 Приклад вмісту БД.

## Опис предметної області «Кінотеатр»

Предметна область “Кінотеатр” передбачає продаж та зберігання квитків. Кінотеатр має декілька залів. В одному залі відбувається декілька сеансів. На різних сеансах демонструються різні фільми. Фільми мають різні жанри, так і один жанр може бути притаманний різним фільмам. У рамках цієї системи, вводиться обмеження на те, що всі зали однакові і мають однакове розміщення крісел.

## Перелік та призначення сутностей

Було виділено наступні сутності:

1. Зал: перелік можливих залів для перегляду (атрибути: номер, технологія)

2. Жанр: перелік можливих жанрів (атрибути: назва)

3. Фільм: опис релевантної інформації стосовно фільмів (атрибути: назва, режисер, рік випуску, тривалість, жанр)

4. Сеанс: опис часу, залу і фільму, на який відвідувач збирається піти (атрибути: фільм, зал, час/дата)

5. Квитки: збір інформації необхідної для покупки білету (атрибути: номер, вартість, ряд, місце)

Через відношення сутності Жанр до Фільмів N:M було додато ще одну таблицю Жанр для (конкретного) фільму.

## Опис зв’язків між сутностями предметної області

Сутності Зали, Жанри, Фільми, Сеанси, Квитки було перетворено у відповідні таблиці.

Таблиця Зали містить колонки номер(первинний ключ) і Технологія(можна було б зробити її окремою сутністю, та щоб не ускладнювати систему, залишила її атрибутом).

Через відношення сутності Жанрів до Фільмів (N:M) було додано ще одну таблицю Жанр для (конкретного) фільму з складеним ключем (composite) для двох зовнішніх ключів - один для Фільму, один для Жанру.

Таблиця Жанри складається з колонок Id(первинний ключ) і Назви(строка).

Таблиця Фільми описується за допомогою Id(первинний ключ), щоб посилатися на конкретний фільм, Назви(строка, не НУЛЛ), прізвище продюсера(строка), рік релізу (ціле число), тривалість(ціле число хвилин).

Таблиця Сеанси також має первинний (сурогатний) ключ Id. Окрім цього, зовнішній ключ на конкретний зал, зовнішній ключ на конкретний фільм, дату і час.

Таблиця Квитки має колонки  Id(первинний ключ), зовнішній ключ на сеанс, місце і ряд (ненульові значення, з перевіркою значення більше 0) і вартість (також не НУЛЛ і більша за нуль).

**Модель «сутність-зв’язок»:**

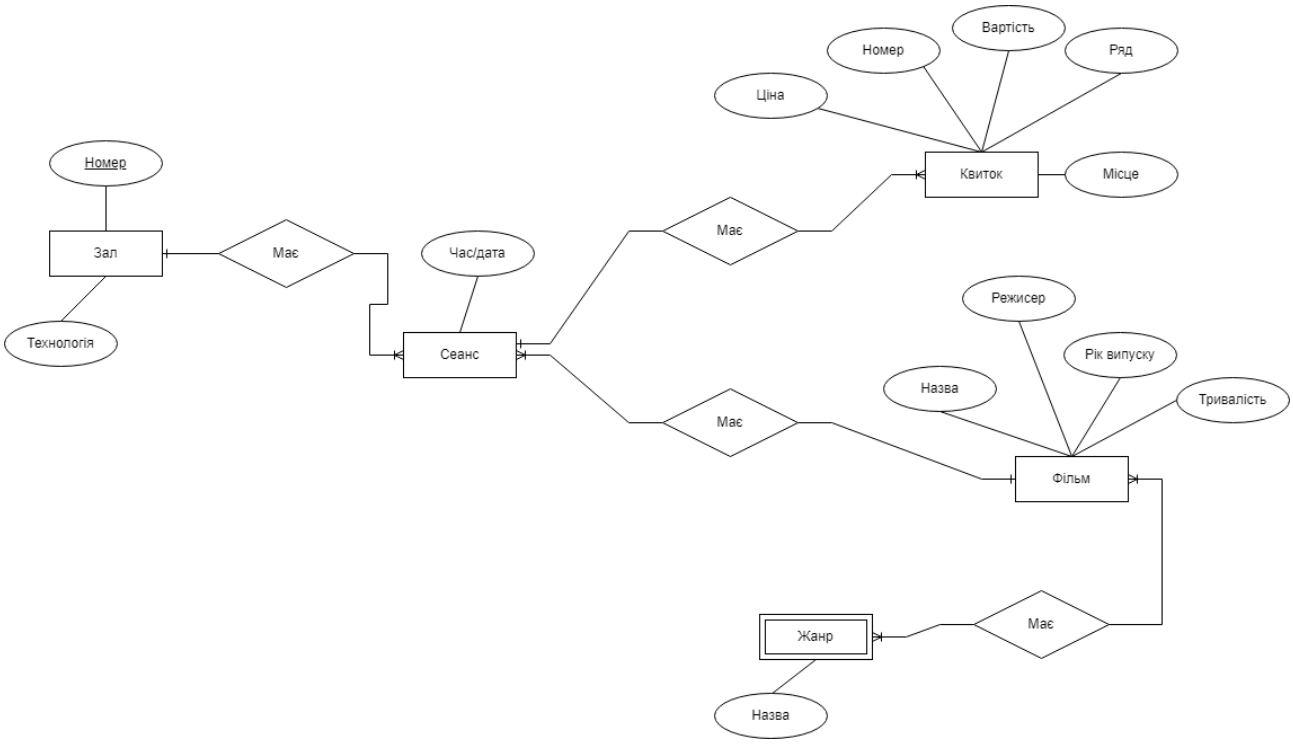


Рис 1. ER-діаграма за нотацією Чена (інструмент: draw.io)

**Схеми бази даних(логічна модель):**

Логічну модель зображено на рис. 2.

## Відповідність схеми бази даних до третьої нормальної форми

Єдине місце, де могла б порушуватися перша нормальна форма, було відношення N:M Жанри -> Фільми. Для цього нова таблиця Film Genres повністю вирішує цю проблему. У нас немає декількох значень, але могла бути у стовпці Жанр для таблиці Фільм, тому перша нормальна форма у нас гарантується.

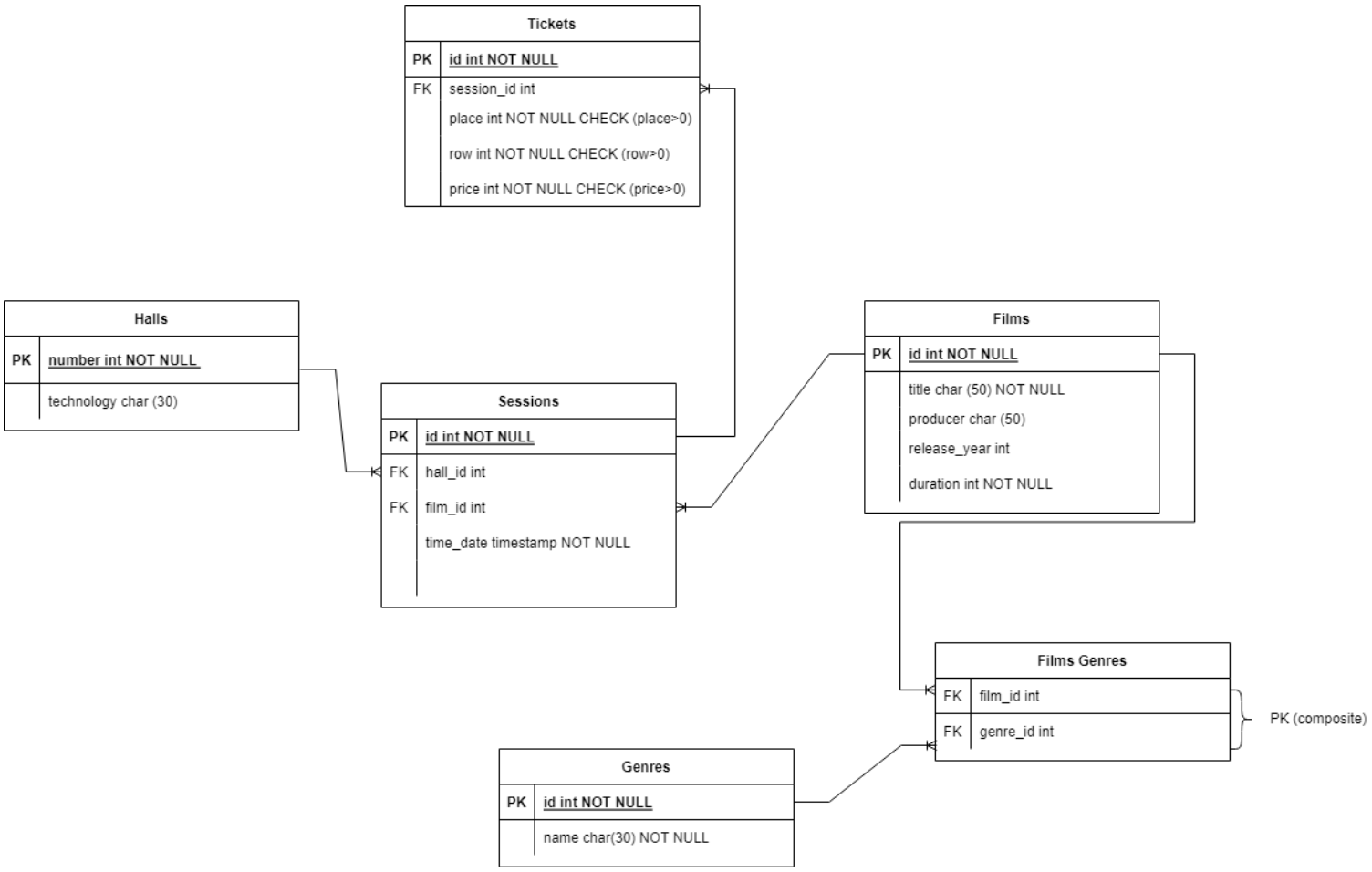


Рис 2. Схема бази даних (інструмент: draw.io)

Відношення знаходиться у другій нормальній формі, якщо кожний неключовий атрибут функціонально залежить від цілого ключа, а не від його частини.

Аналіз таблички Tickets показав, що атрибут Ціна не залежить від цілого ключа у цій табличці, та загалом з урахуванням спрощення системи (усі зали є однаковими) все ще ціна буде залежати від технології, яка використовується у залі, тобто напряму від залу.

TICKETS:

id → session_id, place, row, price

id → session_id (код сеансу залежить від коду квитка)

id → place (місце залежить від коду квитка)

id → row (ряд залежить від коду квитка)

Session_id(hall_id, hall.technology) → price (Не задовольняє 2НФ)

Ціна залежить від технології залу, а не від коду квитка. Необхідно нормалізувати схему.

Для забезпечення 2НФ і вирішення цієї проблеми, було створено окрему таблицю Технології, її змінено з атрибута Halls у окрему таблицю technologies з атрибутами: id, name і description. Тепер Halls буде замість строки технології містити зовнішній ключ technology_id.

Додаємо нову табличку Prices для того, щоб вводити залежності ціни від технології. У цій табличці будуть атрибути: id, technology_id (зовнішній ключ на конкретну технологію ) і, відповідно, ціна, скільки коштує показаний сеанс у такій технології.

На даному етапі аналіз системи показує, що для того, щоб на рівні таблички Tickets дізнатись про ціну, необхідно шукати необхідні дані через три інші таблички, а саме: нам необхідно взяти session_id, з неї витягнути конкретний сеанс, після того взяти hall_id і у табличці Hall взяти конкретний зал, після цього аналогічним чином знайти конкретну технологію за technology_id і у табличці Prices взяти ціну. Для того, щоб спростити процес пошуку ціни для конкретного білета, можна було б додати у Tickets price_id, як зовнішній ключ на конкретну ціну, проте у такому разі воно все ще порушує 2НФ, тому просто видаляємо поле ціни з таблиці Тікетів. У нас не буде дуплікації та уся необхідна інформація у всіх табличках буде актуальною. Для прозорості вирішено перейменувати Tickets на Sold Tickets.

Після змін схема виглядає так (див. рис 3):

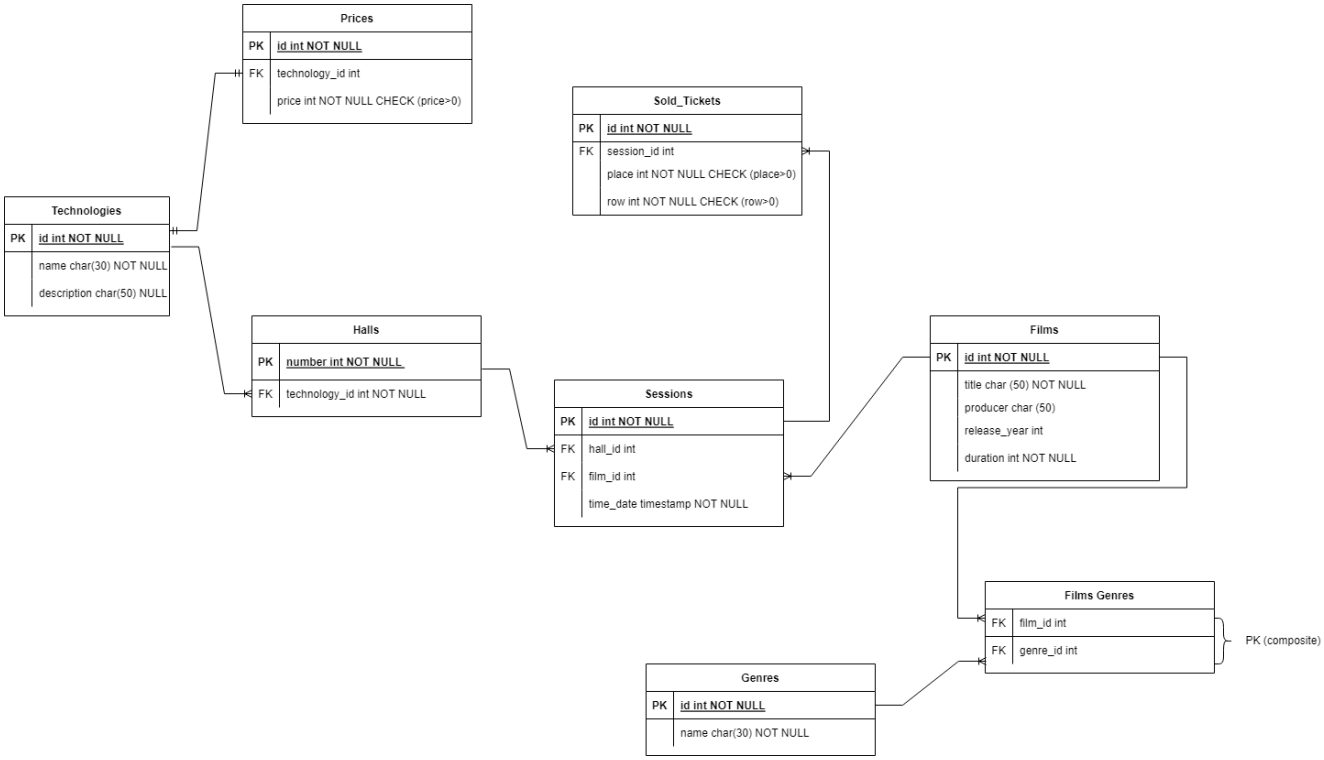


Рис 3. Схема бази даних (інструмент: draw.io)

## Опис об’єктів бази даних у вигляді таблиці

Таблиця 1 – Опис структури бази даних "Кінотеатр"

| Сутність | Атрибут | Тип атрибуту |
|---|---|---|
| **Technologies** - *містить інформацію про доступні у кінотеатрі технології* | **id** - *унікальний ідентифікатор технологій*<br>**name** – *назва технологій. Не допускає NULL*<br>**description** – *опис технологій. Не допускає NULL* | **integer** *(числовий)*<br>**character varying** *(рядок)*<br>**character varying** *(рядок)* |
| **Halls** - *містить інформацію про зал* | **number** - *унікальний ідентифікатор залу*<br>**technology_id** – *код технології.* | **integer** *(числовий)*<br>**integer** *(числовий)* |
| **Genres** – *містить інформацію про перелік усіх можливих жанрів* | **id** - *унікальний ідентифікатор жанру*<br>**name** - *назви жанру. Не допускає NULL* | **integer** *(числовий)*<br>**character varying** *(рядок)* |
| **Films** – *містить всі фільми, що є у прокаті* | **id**- *унікальний ідентифікатор фільму*<br>**title** - *назва фільму. Не допускає NULL*<br>**producer** - *режисер фільму.*<br>**release_year** - *дата виходу фільму.* | **integer** *(числовий)*<br>**character varying** *(рядок)*<br>**character varying** *(рядок)*<br>**integer** *(числовий)* |

| | | |
|---|---|---|
| | **duration** – *тривалість фільму. Не допускає NULL* | **integer** *(числовий)* |
| **Films Genres -** *містить інформацію про усі жанри притаманні фільму* | **film_id**- *унікальний ідентифікатор фільму*<br>***genre_id* - *унікальний ідентифікатор жанру*** | **integer** *(числовий)*<br>**integer** *(числовий)* |
| **Sessions** – *містить інформацію про сеанс* | **id** - *унікальний ідентифікатор сеансу*<br>**film_id** - *унікальний ідентифікатор фільму*<br>**hall_id** - *унікальний ідентифікатор залу*<br>**time_date** – *час початку та дата. Не допускає NULL* | **integer** *(числовий)*<br>**integer** *(числовий)*<br>**integer** *(числовий)*<br>**timestamp** *(час/дата)* |
| **Prices -** *містить інформацію про ціну* | **id -** *унікальний ідентифікатор ціни.*<br>**technology_id -** *унікальний ідентифікатор технологій*<br>**price –** *ціна. Не допускає NULL* | **integer** *(числовий)*<br>**integer** *(числовий)*<br>**integer** *(числовий)* |
| **Tickets** – *містить всю необхідну інформацію про продані квитка* | **id** - *унікальний ідентифікатор квитка*<br>**session_id** - *унікальний ідентифікатор сеансу*<br>**place** - *місце. Не допускає NULL*<br>**row** - *ряд. Не допускає NULL*<br>**price** - *ціна. Не допускає NULL* | **integer** *(числовий)*<br>**integer** *(числовий)*<br>**integer** *(числовий)*<br>**integer** *(числовий)*<br>**integer** *(числовий)* |

**Функціональні залежності для кожної таблиці**

TEHNOLOGIES:

id → name, description

id → name (назва конкретної технології залежить від її коду)

id → description (опис залежить від коду технології)

HALLS:

id → technology_id (код технології залежить від коду залу)

PRICES:

id → technology_id, price

id → technology_id (код технології залежить від коду ціни)

id → price (ціна залежить від коду)

GENRES:

id → name (назва конкретного жанру залежить від його коду)

FILM GENRES:

Primary key (film_id, genre_id)

FILMS:

id → title, producer, release_year, duration

id → title (назва конкретного фільму залежить від його коду)

id → producer (прізвище продюсера залежить від коду фільму)

id → release_year (рік випуску залежить від коду фільму)

id → duration (тривалість залежить від коду фільму)

SESSIONS:

id → hall_id, film_id, session_type_id, date_time

id → hall_id (код залу залежить від коду сеансу)

id → film_id (код фільму залежить від коду сеансу)

id → date_time (час сеансу залежить від коду сеансу)

SOLD_TICKETS:

id → session_id, place, row

id → session_id (код сеансу залежить від коду квитка)

id → place (місце залежить від коду квитка)

id → row (ряд залежить від коду квитка)

# Фізична модель БД «Кінотеатр» у pgAdmin 4

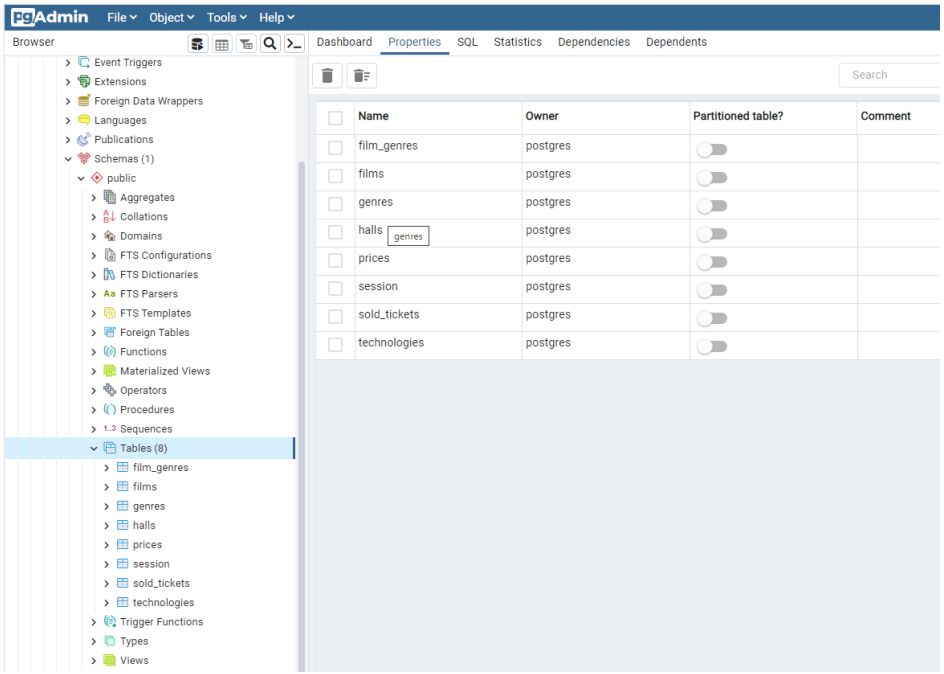


Рис 4. Скрін створених таблиць в PostgreSQL

# Скріни вмісту таблиць

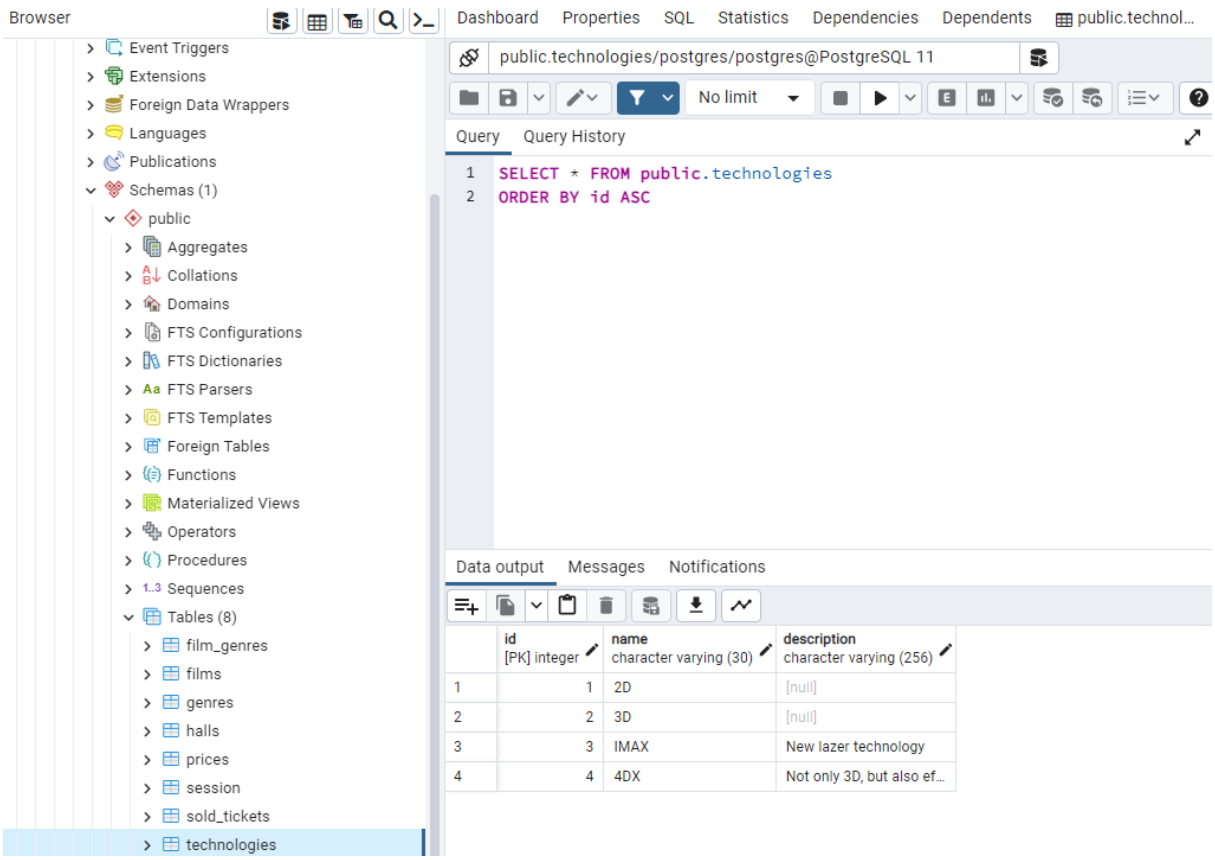


Рис 5. Скрін створених таблиці Технології в PostgreSQL

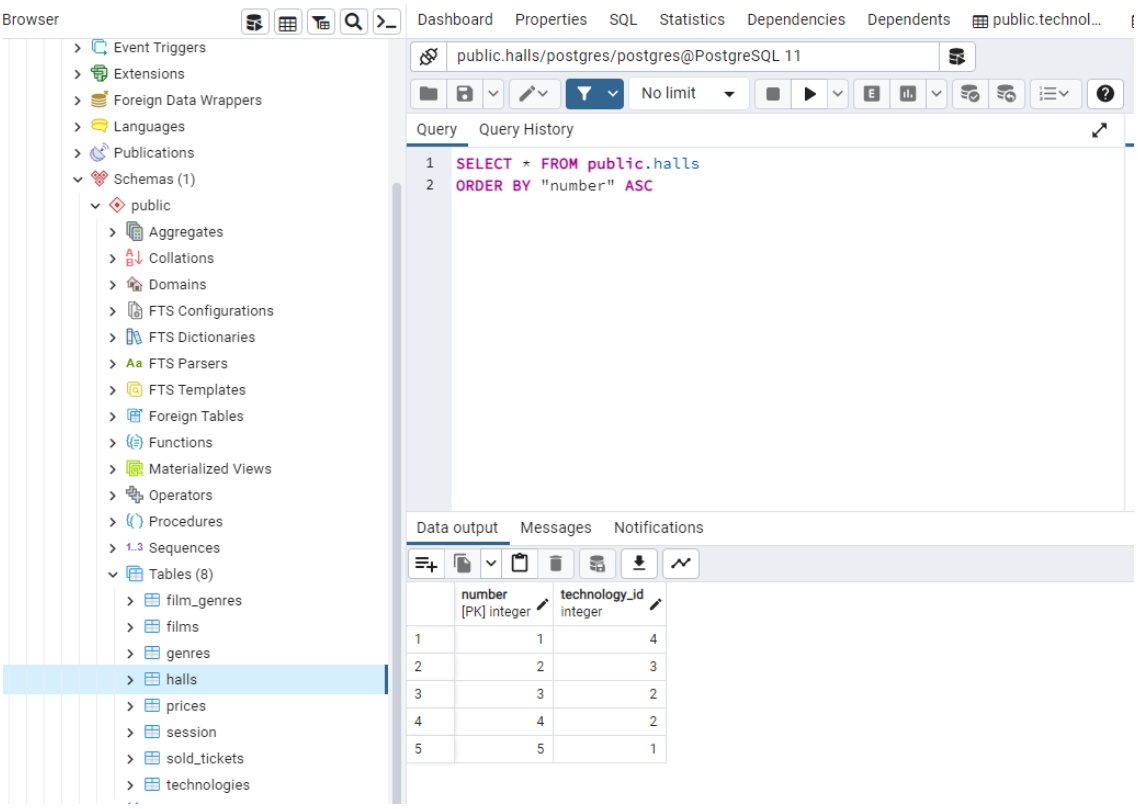


Рис 6. Скрін створених таблиці Зали в PostgreSQL

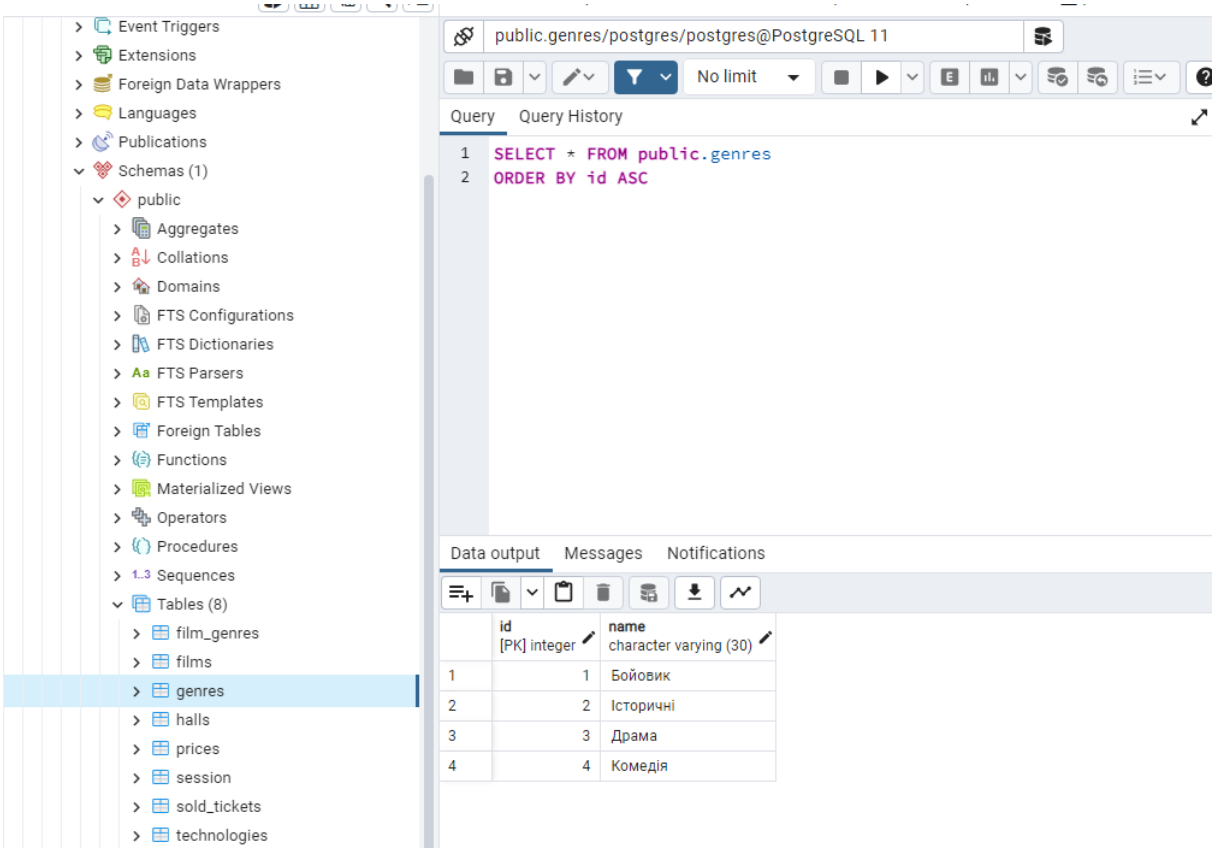


Рис 7. Скрін створених таблиці Жанри в PostgreSQL

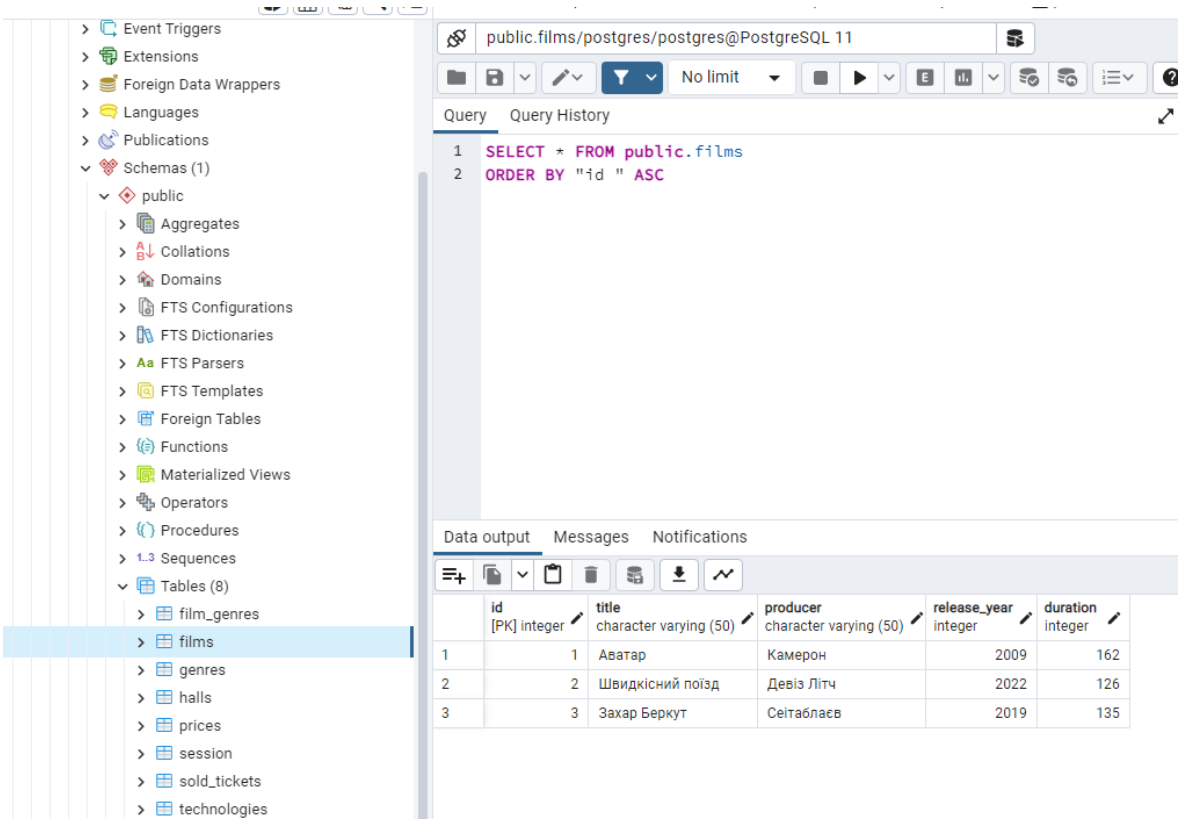


Рис 8. Скрін створених таблиці Фільми в PostgreSQL

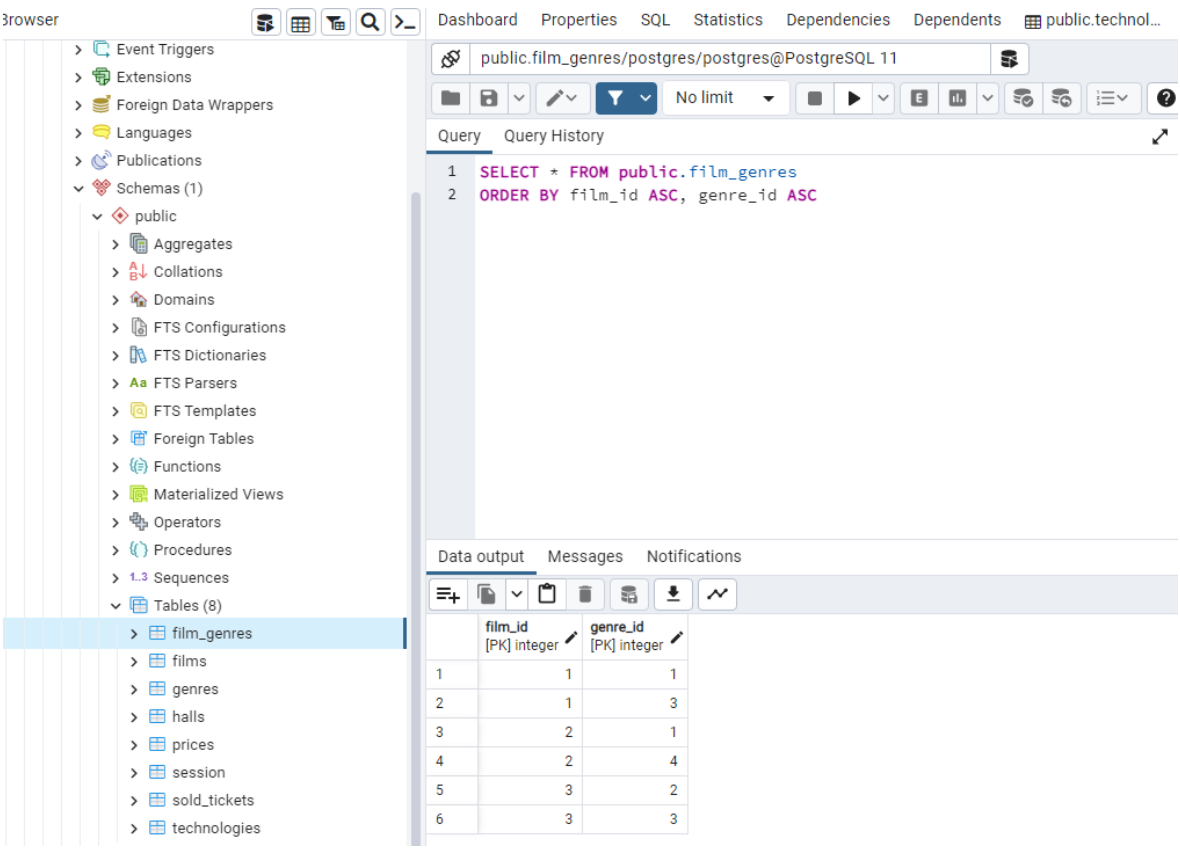


Рис 9. Скрін створених таблиці Жанри фільмів в PostgreSQL

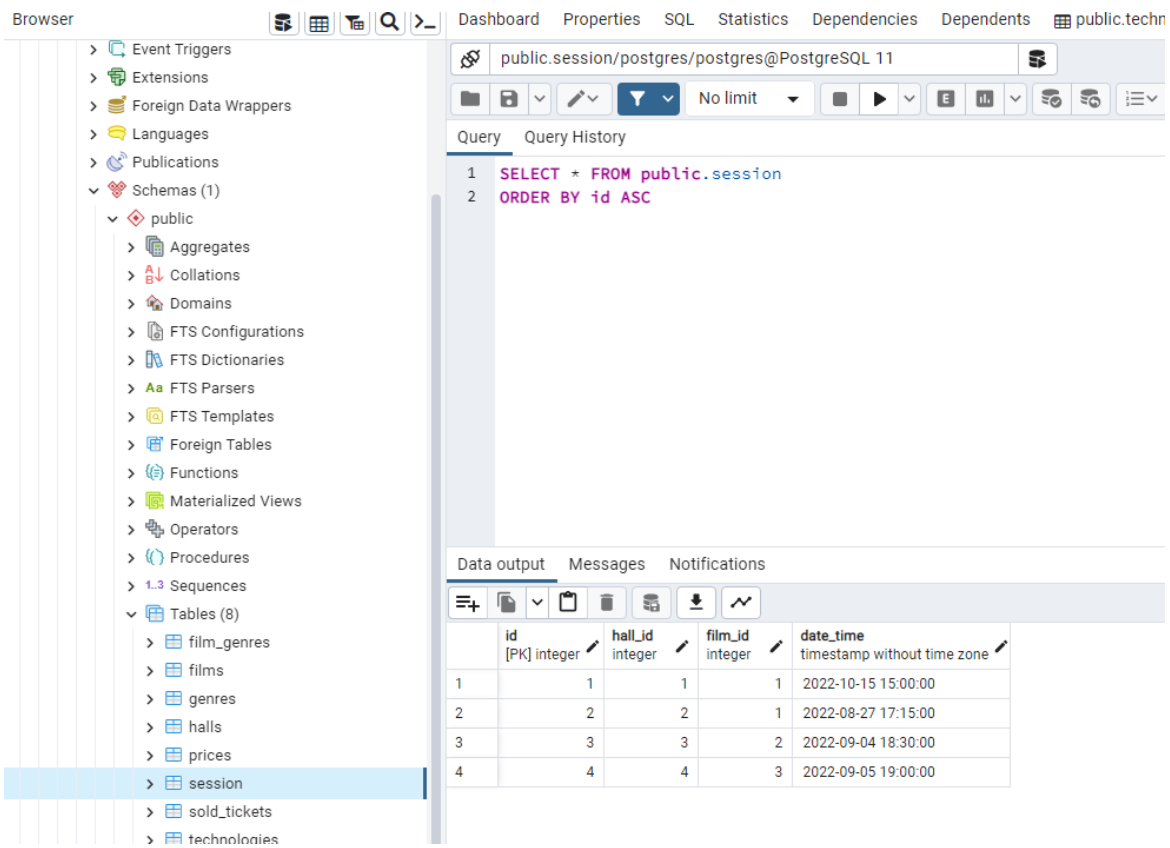


Рис 10. Скрін створених таблиці Сеанси в PostgreSQL

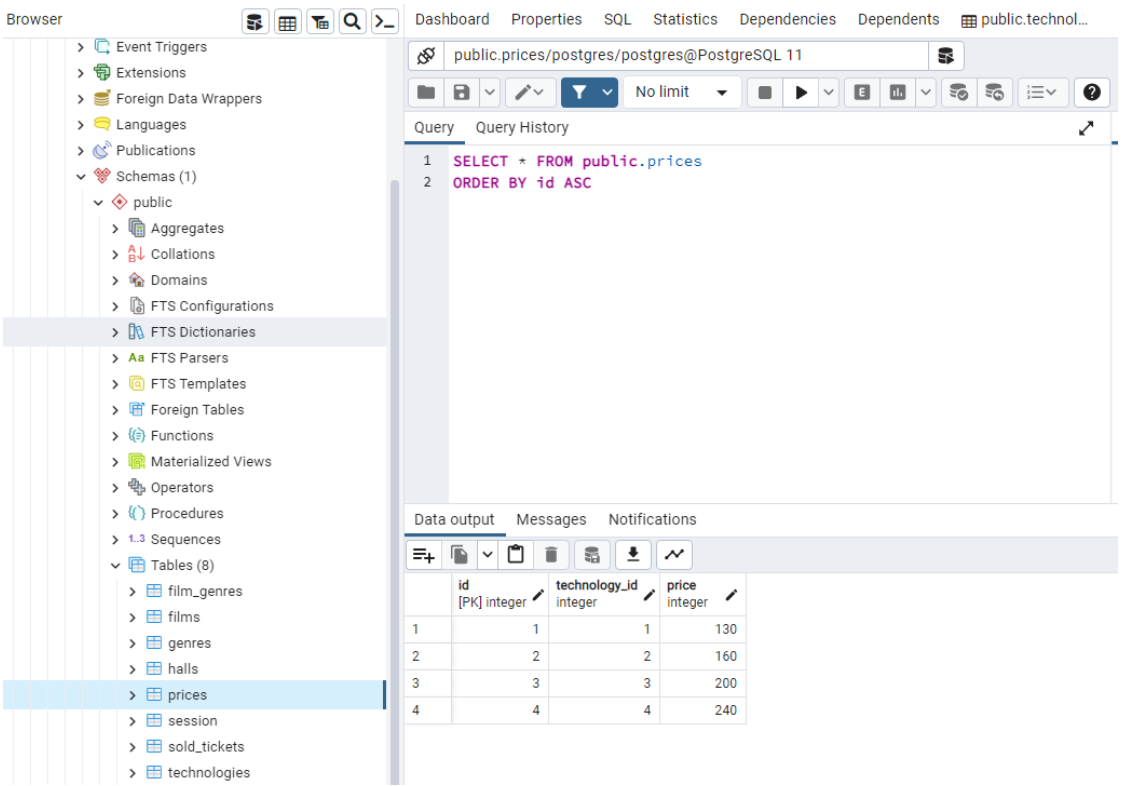


Рис 11. Скрін створених таблиці Ціни в PostgreSQL

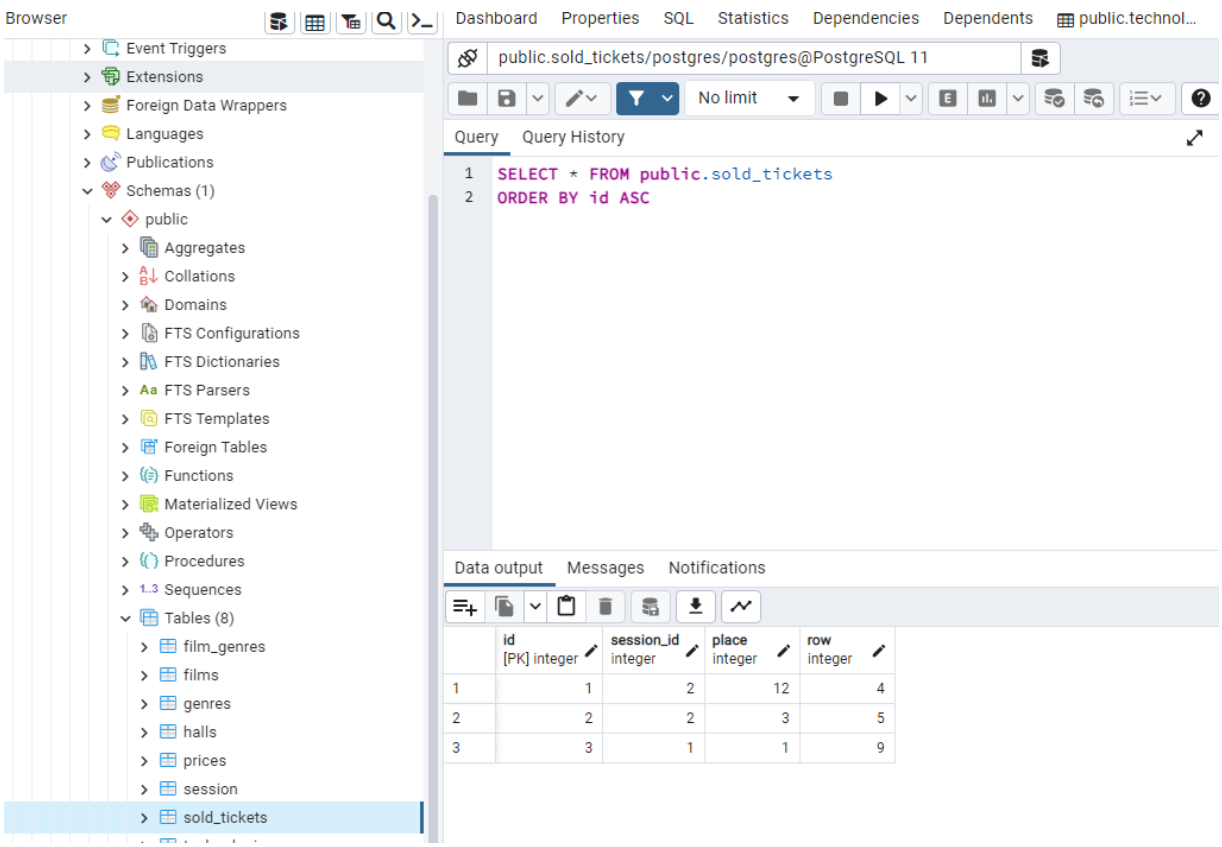


Рис 12. Скрін створених таблиці Продані квитки в PostgreSQL

## SQL-текст опису БД «Кінотеатр»

```
SET statement_timeout = 0;
SET lock_timeout = 0;
SET idle_in_transaction_session_timeout = 0;
SET client_encoding = 'UTF8';
SET standard_conforming_strings = on;
SELECT pg_catalog.set_config('search_path', '', false);
SET check_function_bodies = false;
SET xmloption = content;
SET client_min_messages = warning;
SET row_security = off;

CREATE SCHEMA public;
SET default_with_oids = false;
```

```
CREATE TABLE public.film_genres (
    film_id integer NOT NULL,
    genre_id integer NOT NULL
);

CREATE TABLE public.films (
    "id " integer NOT NULL,
    title character varying(50) NOT NULL,
    producer character varying(50),
    release_year integer,
    duration integer NOT NULL
);

CREATE TABLE public.genres (
    id integer NOT NULL,
    name character varying(30) NOT NULL
);

CREATE TABLE public.halls (
    number integer NOT NULL,
    technology_id integer NOT NULL
);

CREATE TABLE public.prices (
    id integer NOT NULL,
    technology_id integer NOT NULL,
    price integer NOT NULL,
    CONSTRAINT positive_price CHECK ((price > 0))
);

CREATE TABLE public.session (
    id integer NOT NULL,
    hall_id integer NOT NULL,
    film_id integer NOT NULL,
```

```
    date_time timestamp without time zone NOT NULL
);
CREATE TABLE public.sold_tickets (
    id integer NOT NULL,
    session_id integer NOT NULL,
    place integer NOT NULL,
    "row" integer NOT NULL,
    CONSTRAINT tickets_place_check CHECK ((place > 0)),
    CONSTRAINT tickets_row_check CHECK (("row" > 0))
);

CREATE TABLE public.technologies (
    id integer NOT NULL,
    name character varying(30) NOT NULL,
    description character varying(256)
);

ALTER TABLE ONLY public.film_genres
    ADD CONSTRAINT film_genres_pkey PRIMARY KEY (film_id, genre_id);

ALTER TABLE ONLY public.films
    ADD CONSTRAINT films_pkey PRIMARY KEY ("id ");

ALTER TABLE ONLY public.genres
    ADD CONSTRAINT genres_pkey PRIMARY KEY (id);

ALTER TABLE ONLY public.halls
    ADD CONSTRAINT halls_pkey PRIMARY KEY (number);

ALTER TABLE ONLY public.prices
    ADD CONSTRAINT prices_pkey PRIMARY KEY (id);

ALTER TABLE ONLY public.session
    ADD CONSTRAINT session_pkey PRIMARY KEY (id);
```

```
ALTER TABLE ONLY public.technologies
    ADD CONSTRAINT technologies_pkey PRIMARY KEY (id);

ALTER TABLE ONLY public.sold_tickets
    ADD CONSTRAINT tickets_pkey PRIMARY KEY (id);

ALTER TABLE ONLY public.film_genres
    ADD CONSTRAINT film_id FOREIGN KEY (film_id) REFERENCES
public.films("id ");

ALTER TABLE ONLY public.session
    ADD CONSTRAINT film_id FOREIGN KEY (film_id) REFERENCES
public.films("id ");

ALTER TABLE ONLY public.film_genres
    ADD CONSTRAINT genre_id FOREIGN KEY (genre_id) REFERENCES
public.genres(id);

ALTER TABLE ONLY public.session
    ADD CONSTRAINT hall_id FOREIGN KEY (hall_id) REFERENCES
public.halls(number);
ALTER TABLE ONLY public.sold_tickets
    ADD CONSTRAINT session_id FOREIGN KEY (session_id) REFERENCES
public.session(id);

ALTER TABLE ONLY public.halls
    ADD CONSTRAINT technology_id FOREIGN KEY (technology_id) REFERENCES
public.technologies(id);

ALTER TABLE ONLY public.prices
    ADD CONSTRAINT technology_id FOREIGN KEY (technology_id) REFERENCES
```