Изображение выглядит как текст

Автоматически созданное описание

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 3 курсу

групи КВ-92

Невінчаний А. В.

Перевірив:

Павловський В.І.

Київ – 2021

**Мета роботи**

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.

**Завдання 1**

**Опис структурі БД «Кінотеатр»**

Модель «сутність-зв’язок» галузі продажу квитків у кіно.

Модель має чотири сутності: movie, showtime, cinema, ticket.

Сутність movie – описує фільм, на який був куплений квиток. Кожен фільм має власний id, назву та короткий опис.

Сутність showtime – описує час сеансу. Атрибутами даної сутності є персональний номер сеансу, час початку сеансу та id фільму.

Сутність cinema – описує кінотеатр. Кожен кінотеатр має власний id, назву, id фільму.

Сутність ticket – описує квиток, куплений глядачем. Кожен квиток має власний id, ім’я глядача, дату перегляду, вартість, номер ряду та місця у залі, id сеансу.

**Опис зв’язків**

Оскільки у одному кінотеатрі може показуватись багато фільмів(не водночас), то між сутностями cinema та movie існує зв'язок 1:N.

На один сеанс може бути придбано багато квитків, тому між сутностями showtime та ticket існує зв’язок 1:M.

Оскільки у одного фільму є декілька сеансів, тому між сутностями movie та showtime існує зв’язок 1: M.

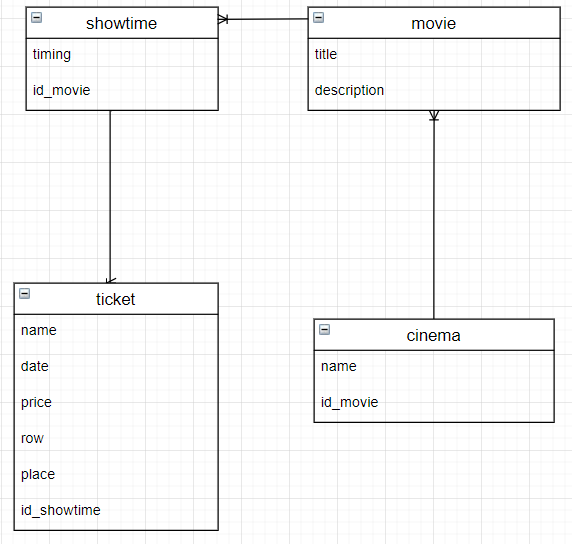


Рисунок 1 - Концептуальна модель

**Завдання 2**

**Опис таблиць**

Сутність cinema була перетворена у таблицю cinema, а зв’язок 1:N цієї сутності з сутністю movie зумовив появу у ній зовнішнього ключа id\_movie.

Сутність movie була перетворена у таблицю movie.

Сутність showtime була перетворена у таблицю showtime, а зв’язок 1:N цієї сутності з сутністю movie зумовив появу у ній зовнішнього ключа id\_movie.

Сутність ticket була перетворена у таблицю ticket, а зв’язок 1:N цієї сутності з сутністю showtime зумовив появу у ній зовнішнього ключа id\_showtime.

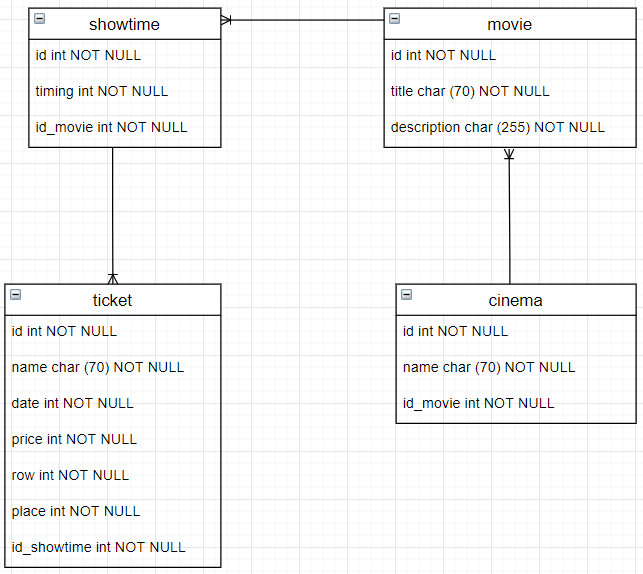


Рисунок 2 - Логічна модель

|  |  |  |
| --- | --- | --- |
| **Відношення** | **Атрибут** | **Тип** |
| Відношення «movie»  Містить інформація про фільм | id – персональний номер фільму  title – назва фільму  description – короткий опис фільму | Числовий  Текстовий (70)  Текстовий (500) |
| Відношення «showtime»  Містить інформацію про розклад сеансів | id – персональний номер сеансу  timing – тривалість сеансу  id\_movie – номер фільму на даний сеанс | Числовий  Числовий  Числовий |
| Відношення «cinema»  Містить інформацію про кінотеатр | id – персональний номер кінотеатру  name – назва кінотеатру  id\_movie – персональний номер сеансу | Числовий  Текстовий (70)  Числовий |
| Відношення «ticket»  Містить інформацію про придбаний квиток | id – персональний номер квитка  name – ім’я глядача  date – дата перегляду фільму  price – вартість квитка  id\_showtime – персональний номер сеансу  row – номер ряду у залі  place – номер місця у залі | Числовий  Текстовий (70)  Числовий  Числовий  Числовий  Числовий  Числовий |

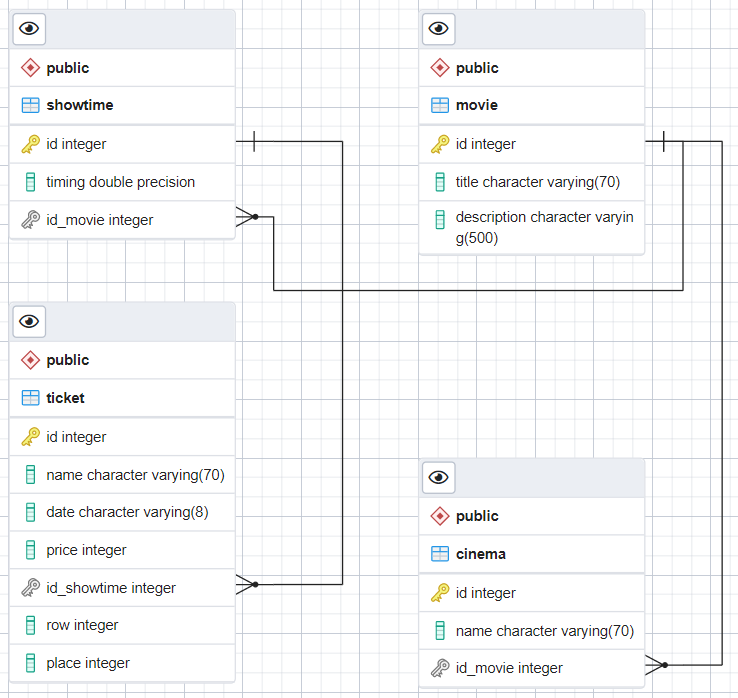


Рисунок 3 - Схема бази даних у pgAdmin 4

**Завдання 3**

**Нормалізація схеми базі даних до третьої нормальної форми**

**Функціональні залежності для кожної таблиці**

movie (id, title, description)

id → title

id → description

id → title, description

showtime (id, timing, id\_movie)

id → timing

id → id\_movie

id → timing, id\_movie

cinema (id, name, id\_movie)

id → name

id → id\_movie

id → name, id\_movie

ticket (id, name, date, price, id\_showtime, row, place)

id → name

id → date

id → price

id → id\_showtime

id → row

id → place

id → name, date, price, id\_showtime, row, place

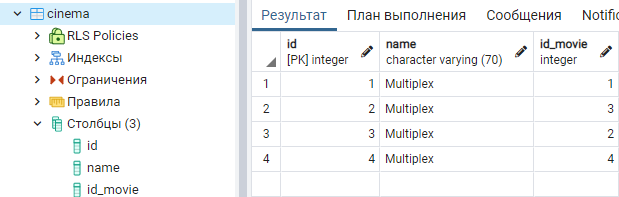
Схема бази даних відповідає нормальній формі НФ1, тому що всі атрибути таблиці є атомарними і в них немає рядків даних, що повторюються.

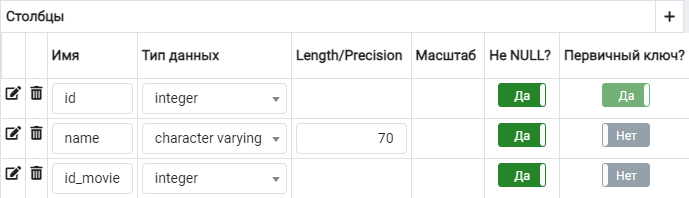
Схема бази даних відповідає нормальній формі НФ2, тому що вона відповідає нормальній формі НФ1 і кожен неключовий атрибут функціонально залежить від цілого ключа, а не від його частини.

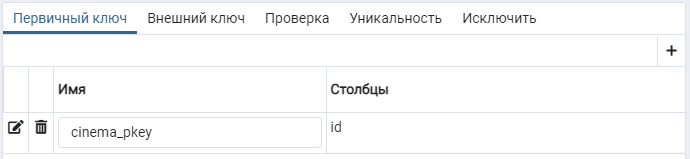
Схема бази даних відповідає нормальній формі НФ3, тому що вона відповідає нормальній формі НФ2 і відсутні транзитивні залежності між даними.

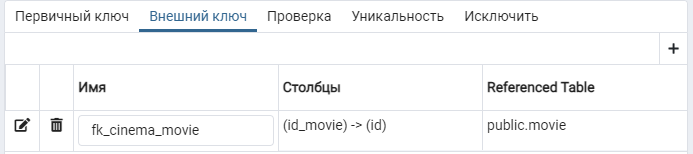
**Завдання 4**

Cinema:

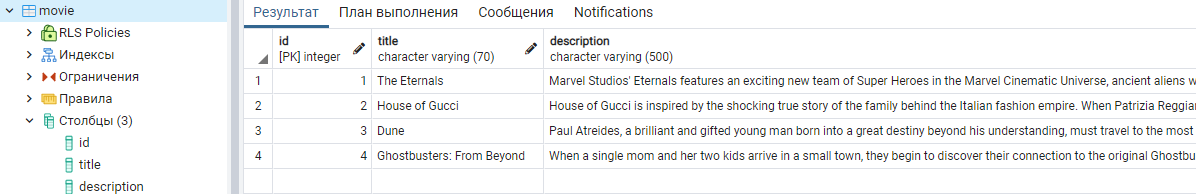


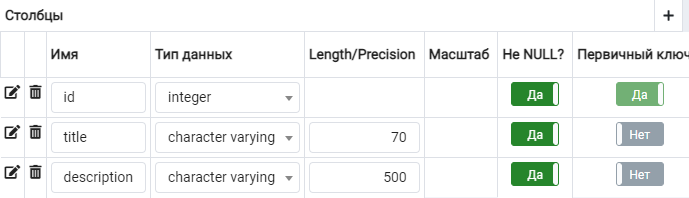


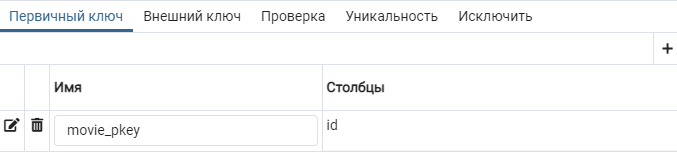




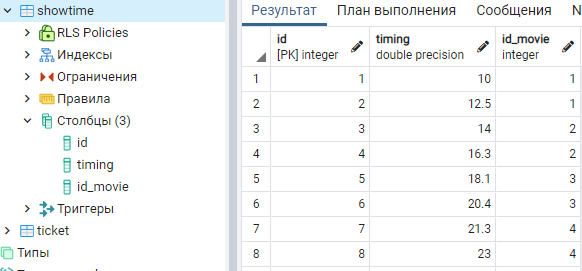
Movie:



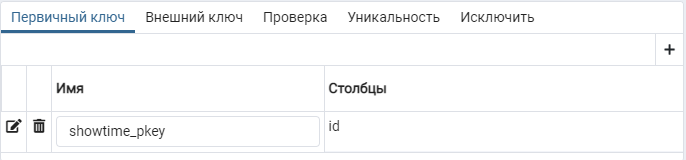


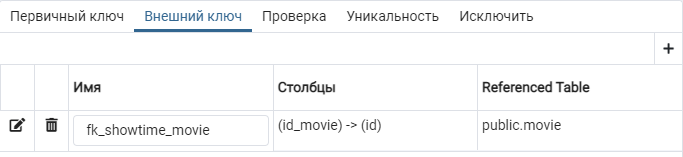


Showtime:

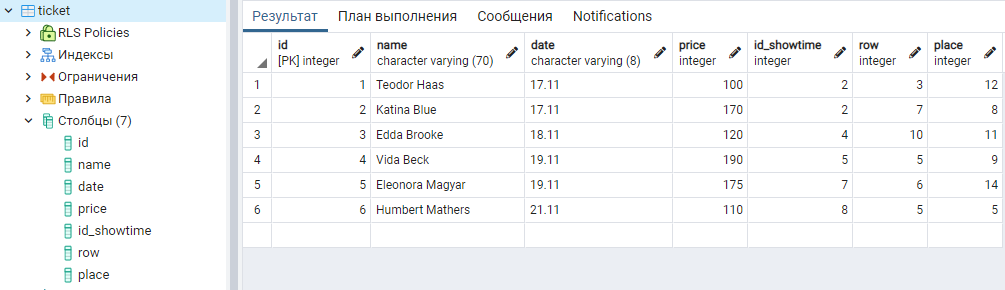


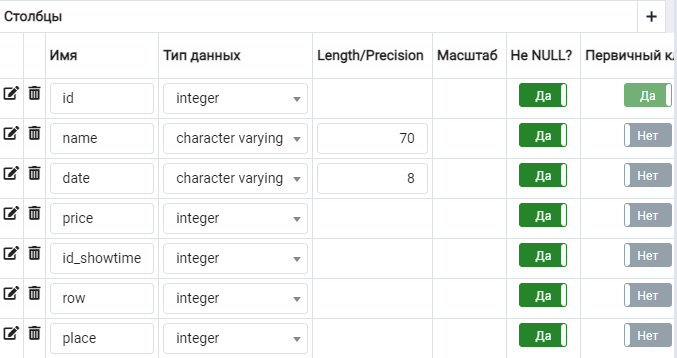


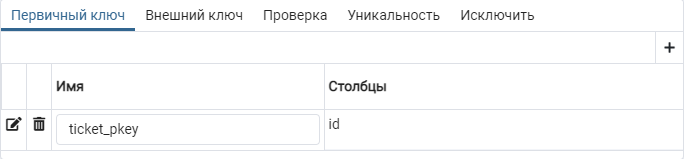


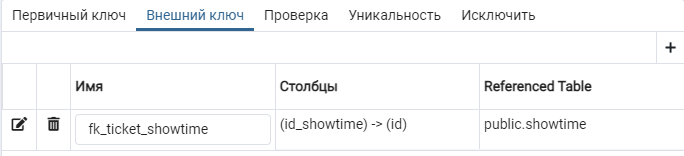


Ticket:









Згенерований SQL-текст:

BEGIN;

CREATE TABLE IF NOT EXISTS public.cinema

(

id integer NOT NULL,

name character varying(70) NOT NULL,

id\_movie integer NOT NULL,

PRIMARY KEY (id)

);

CREATE TABLE IF NOT EXISTS public.movie

(

id integer NOT NULL,

title character varying(70) NOT NULL,

description character varying(500) NOT NULL,

PRIMARY KEY (id)

);

CREATE TABLE IF NOT EXISTS public.showtime

(

id integer NOT NULL,

timing double precision NOT NULL,

id\_movie integer NOT NULL,

PRIMARY KEY (id)

);

CREATE TABLE IF NOT EXISTS public.ticket

(

id integer NOT NULL,

name character varying(70) NOT NULL,

date character varying(8) NOT NULL,

price integer NOT NULL,

id\_showtime integer NOT NULL,

"row" integer NOT NULL,

place integer NOT NULL,

PRIMARY KEY (id)

);

ALTER TABLE public.cinema

ADD FOREIGN KEY (id\_movie)

REFERENCES public.movie (id)

NOT VALID;

ALTER TABLE public.showtime

ADD FOREIGN KEY (id\_movie)

REFERENCES public.movie (id)

NOT VALID;

ALTER TABLE public.ticket

ADD FOREIGN KEY (id\_showtime)

REFERENCES public.showtime (id)

NOT VALID;

END;