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"

Виконав:
студент III курсу
групи КВ-21
Кузнецов Д. С.
Перевірив:
Павловский В. І.

Київ – 2024

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

Завдання роботи:

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.

Опис предметної області

Обрана предметна область – онлайн-платформа для здачі та оренди нерухомості.

Вона охоплює охоплює всі аспекти взаємодії між користувачами та управління даними для успішного функціонування онлайн-платформи.

Опис сутностей

Для побудови бази даних обраної області, були виділені такі сутності:

1. Користувач (Users)

Атрибути: ідентифікатор користувача, ім’я, електронна пошта, роль (орендодавець, орендар).

Призначення: збереження даних користувачів.

2. Оголошення оренди (Rental)

Атрибути: ідентифікатор оголошення, назва, опис, ціна, ідентифікатор користувача.

Призначення: збереження даних щодо оголошень оренди.

3. Бронювання (Reservation)

Атрибути: ідентифікатор броні, дата заселення, дата виселення, ідентифікатор користувача, ідентифікатор оголошення.

Призначення: збереження даних щодо орендованих квартир.

4. Відгуки (Reviews)

Атрибути: ідентифікатор відгуку, рейтинг, коментар, ідентифікатор користувача, ідентифікатор оголошення.

Призначення: збереження даних щодо рейтингу оголошень та відгуків.

Опис зв’язків між сутностями

Зв’язок Користувач - Оголошення оренди є зв’язком 1:N. Один Користувач може публікувати багато оголошень, але одне оголошення може бути створене лише одним користувачем.

Зв’язок Користувач - Бронювання є зв’язком N:M. Один Користувач може здійснити багато бронювань, і нерухомість може бронюватися багатьма Користувачами на різні дати.

Зв’язок Оголошення оренди - Бронювання є зв’язком 1:N. Одне оголошення може мати багато бронювань, але всі бронювання одної нерухомості здійснюються з одного оголошення.

Зв’язок Оголошення оренди - Відгуки є зв’язком 1:M. Одне оголошення може мати багато відгуків, але кожен відгук пов'язаний з одним оголошенням.

Зв’язок Користувач - Відгуки є зв’язком 1:N. Один Користувач може написати багато відгуків, але кожен відгук закріплений за одним користувачем.

Графічне подання концептуальної моделі «Сутність-зв’язок» зображено на рисунку 1.

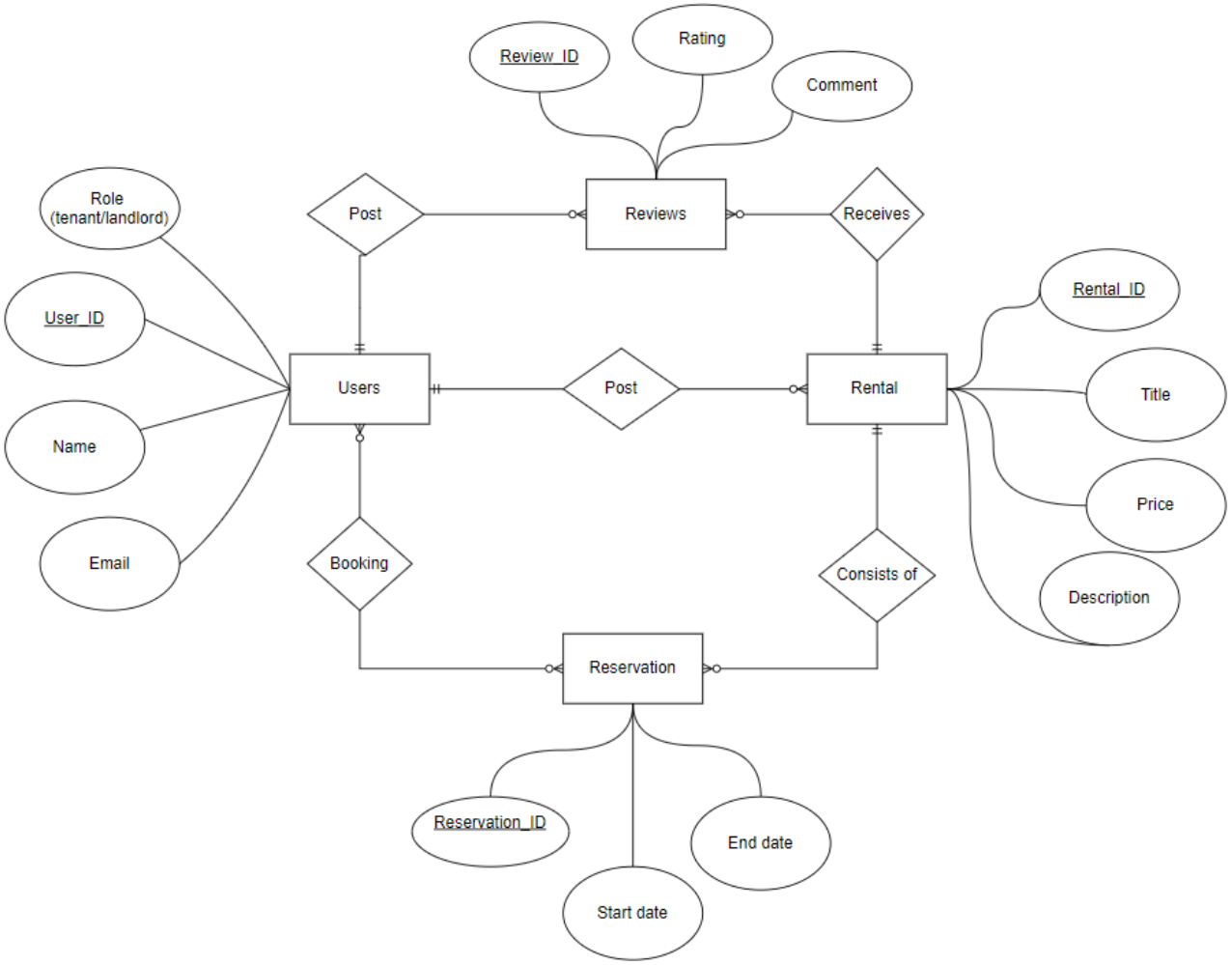


Рисунок 1 – ER-діаграма, побудована за нотацією Crow’s Foot

Перетворення концептуальної моделі у логічну модель та схему бази даних

Сутність User перетворено в таблицю Users з первинним ключем user_id та атрибутами name, email, role.

Сутність Rental перетворено в таблицю Rental з первинним ключем rental_id та атрибутами title, price, description та зовнішнім ключем user_id.

Сутність Reservation перетворено в таблицю Reservation з первинним ключем reservation_id та атрибутами start_date, end_date та зовнішніми ключами user_id, rental_id.

Сутність Reviews перетворено в таблицю Reviews з первинним ключем review_id та атрибутами rating, comment та зовнішніми ключами user_id, rental_id.

Оскільки в логічній моделі безпосередній зв'язок N:M є неможливим, а в концептуальній моделі він існує між сутностями Users і Reservation, то для його реалізації було створено таблицю Transactions, з первинним ключем transaction_id, та зовнішніми ключами user_id і reservation_id.

Таблиця 1 ілюструє детальний перехід від однієї моделі до іншої.

Таблиця 1 – Опис об'єктів бази даних

Сутність	**Атрибут**	**Тип атрибуту**
Users – містить дані про користувачів	**user_id** – унікальний ідентифікатор користувача	integer, PK, NOT NULL, UNIQUE
	name – ім'я користувача	VARCHAR(50), NOT NULL
	email – електронна пошта користувача	VARCHAR(50), NOT NULL
	role – орендодавець або орендар	VARCHAR(50), NOT NULL
Rental – містить дані про оголошення оренди	**rental_id** – унікальний ідентифікатор оголошення оренди	integer, PK, NOT NULL, UNIQUE

	title – назва оголошення	VARCHAR(50), NOT NULL
	description – опис оголошення	VARCHAR(1000), NOT NULL
	price – ціна оренди	integer, NOT NULL
	user_id – кількість студентів у групі	integer, FK, NOT NULL
Reservation – містить дані про бронювання нерухомості	**reservation_id** – унікальний ідентифікатор запису бронювання	integer, PK, NOT NULL, UNIQUE
	rental_id – ідентифікатор оголошення оренди	integer, FK, NOT NULL
	start_date – дата початку оренди	date, NOT NULL
	end_date – дата кінця оренди	date, NOT NULL
Reviews – містить дані про відгуки користувачів	**review_id** – унікальний ідентифікатор відгука	integer, PK, NOT NULL, UNIQUE
	user_id – ідентифікатор користувача, що написав відгук	integer, FK, NOT NULL

	rental_id – ідентифікатор оголошення, на яке написано відгук	integer, FK, NOT NULL
	rating – оцінка	integer, NOT NULL
	comment – коментар	VARCHAR(300)
Transactions – містить дані про транзакцію (підтвердження) бронювання для користувача	**transaction_id** – унікальний ідентифікатор транзакції	integer, PK, NOT NULL, UNIQUE
	user_id – ідентифікатор користувача	integer, FK, NOT NULL
	reservation_id – ідентифікатор запису бронювання	integer, FK, NOT NULL

Графічне подання логічної моделі «Сутність-зв’язок» зображено на рисунку 2.

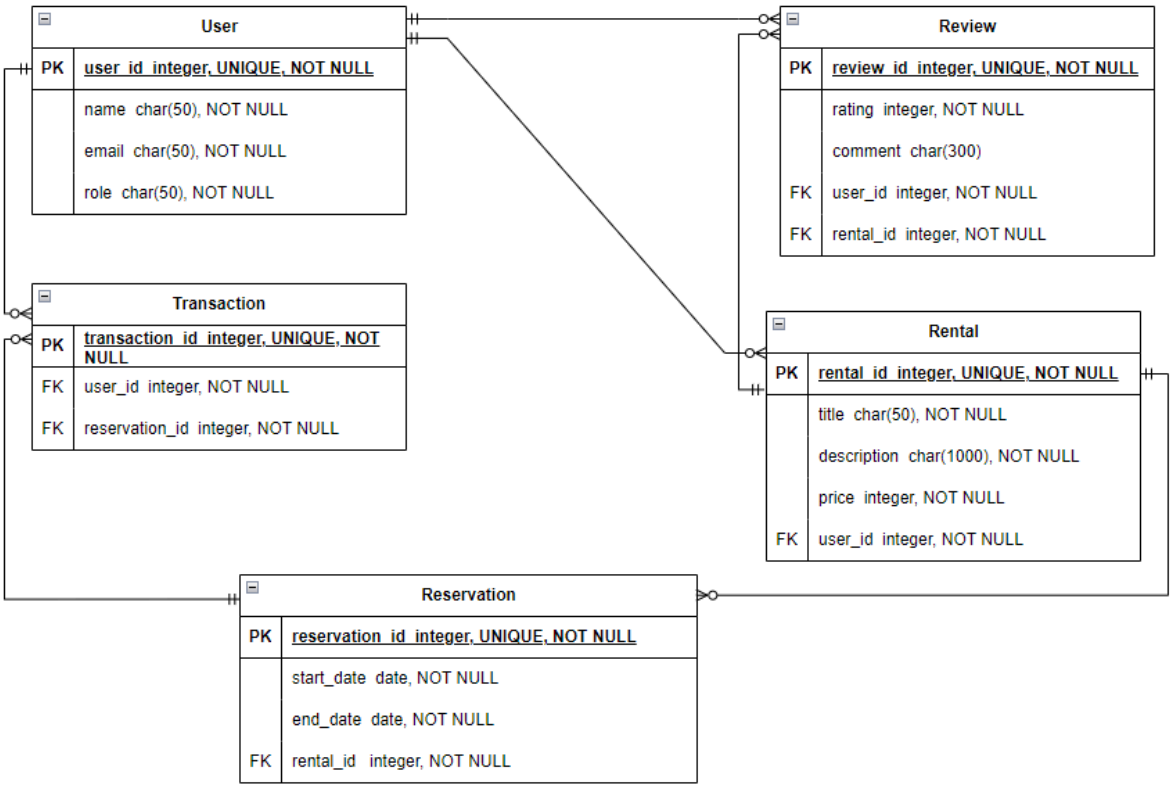


Рисунок 2 – Схема бази даних

Функціональні залежності для кожної таблиці

1. Users:
    - user_id → name, email, role

(Ідентифікатор користувача визначає всі його персональні дані та роль на платформі)

2. Rental:
    - rental_id → user_id, title, description, price

(rental_id об'єкта визначає його тип, власника і характеристики)

- user_id→ rental_id

(Один користувач може мати декілька оголошень нерухомості)

3. Reservation:
    - reservation_id → rental_id, start_date, end_date

(reservation_id об'єкта визначає його тип, дати початку та кінця оренди та всю інформацію про бронювання та оголошення до якого відноситься бронювання)

    - rental_id→ reservation_id

(Одне оголошення може містити багато бронювань на різні дати)

4. Transactions:
    - transaction_id → reservation_id , user_id

(transaction_id визначає договір на бронювання для користувача)

5. Reviews:
    - review_id → user_id, rating, comment, rental_id

(review_id визначає всю інформацію про відгук, його автора, зміст та оголошення до якого відноситься відгук)

    - user_id → review_id

(Один користувач може залишити кілька відгуків)

    - rental_id → review_id

(Одне оголошення може мати кілька відгуків)

Ці функціональні залежності вказують на те, які атрибути в кожній таблиці визначаються від інших атрибутів. Це важливо для нормалізації та управління базою даних.

Транзитивні функціональні залежності виникають, коли один атрибут функціонально визначає інший через інший атрибут. Іншими словами, якщо A визначає B, а B визначає C, то ми можемо сказати, що A транзитивно визначає C. Є декілька транзитивних відношень, але вони включають в себе ключовий атрибут.

Відповідність схеми нормальним формам

1. Щоб задовольнити умови 1НФ кожен атрибут в таблиці має бути атомарним, тобто:

    - Кожна клітинка містить єдине значення;
    - Кожен запис є унікальним.

    Дана схема відповідає 1НФ.

2. Щоб схема відповідала 2НФ повинні виконуватись умови:

    - Схема перебуває в 1НФ;
    - Кожний неключовий атрибут функціонально залежить від цілого ключа.

У даній схемі кожна таблиця має власний унікальний ідентифікатор (ключ). Кожний атрибут у кожній таблиці залежить від цього унікального ідентифікатора. Тобто, схема також в НФ2.

3. Щоб схема відповідала 3НФ повинні виконуватись умови:

    - Схема перебуває в 2НФ;
    - Кожен не первинний атрибут має бути не транзитивно залежним від кожного ключа.

Оскільки дана схема в НФ2 та неключові атрибути не транзитивно залежать від інших неключових атрибутів, схема також в НФ3.

Отже, схема бази даних відповідає нормальним формам НФ1, НФ2 та НФ3. Вона добре структурована і нормалізована, що сприяє ефективному та надійному зберіганню та обробці даних.

Схема бази даних у pgAdmin 4 зображена на рисунку 3.

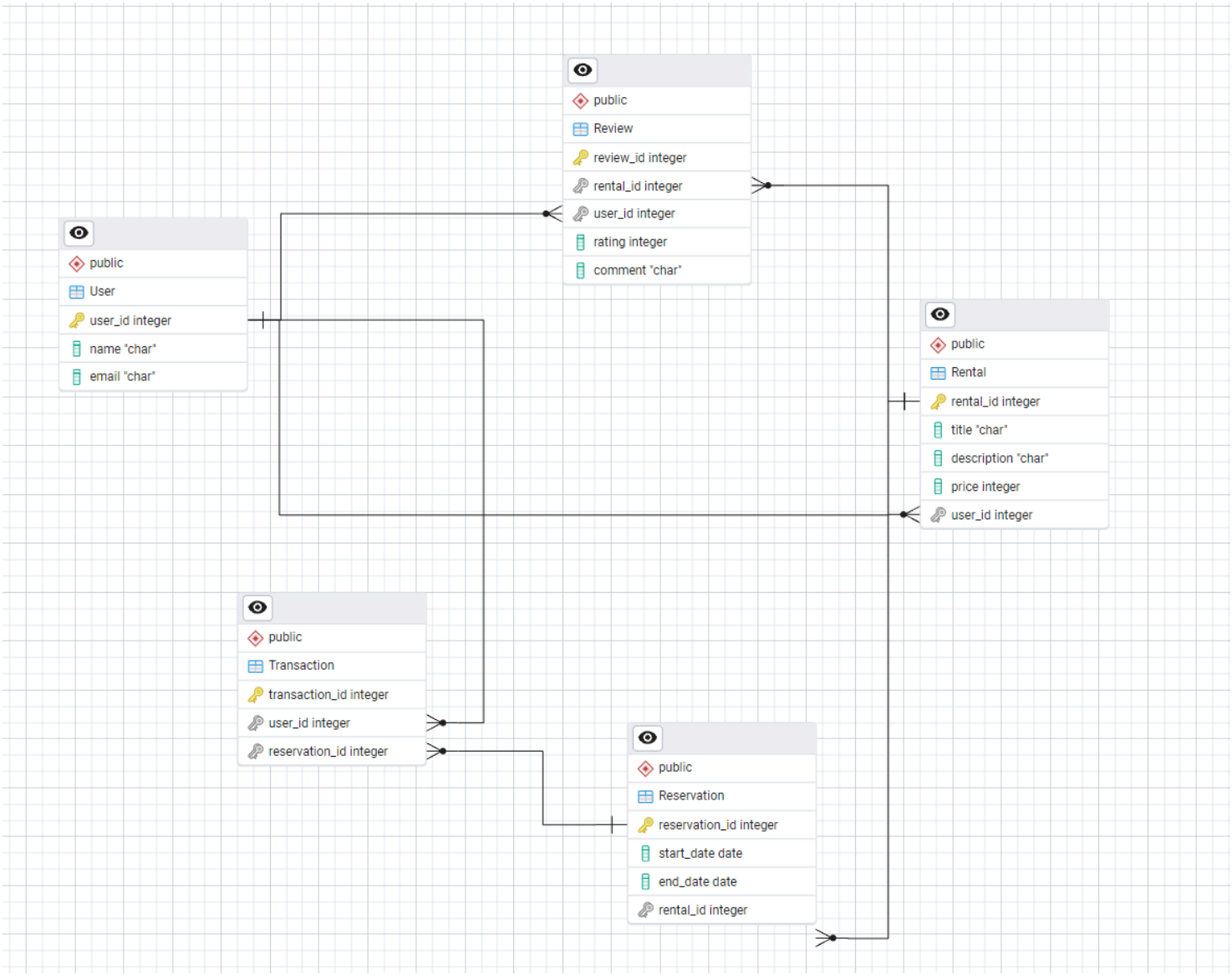
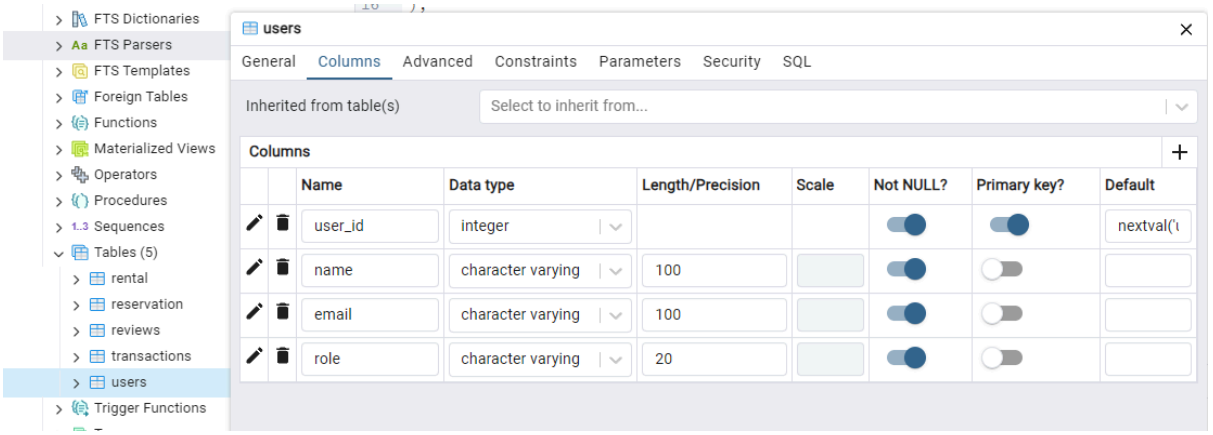
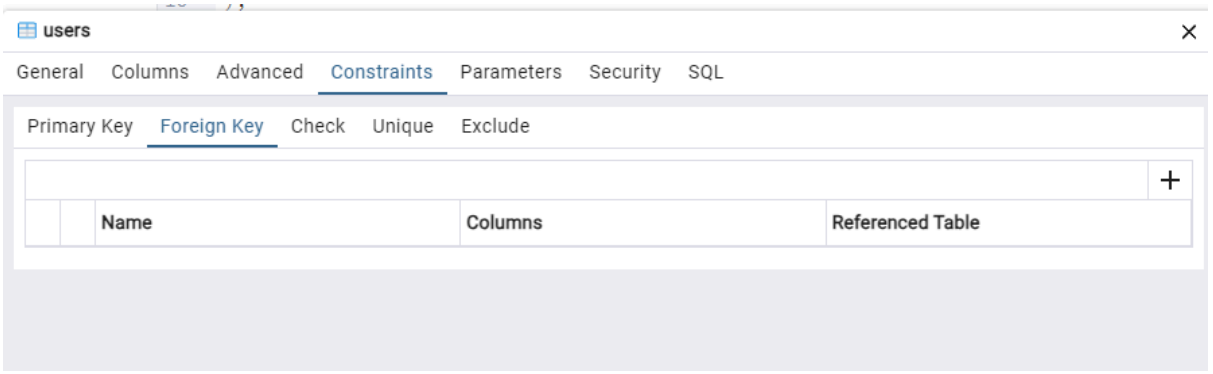


Рисунок 3 - Схема бази даних у pgAdmin 4

Таблиці бази даних у pgAdmin4

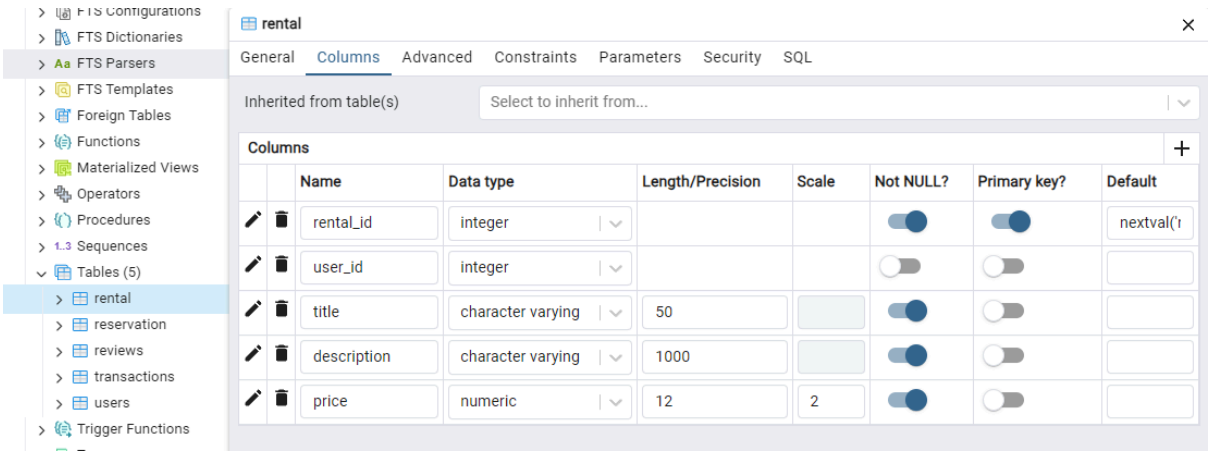
Users

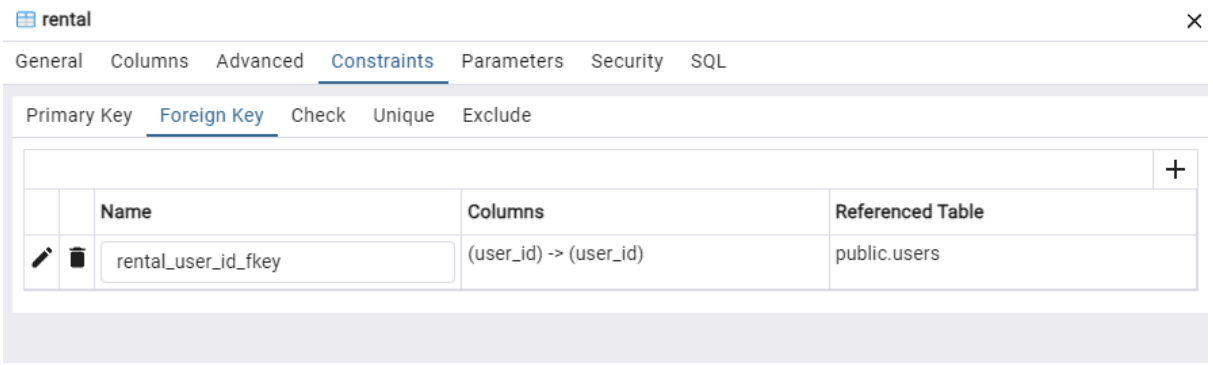




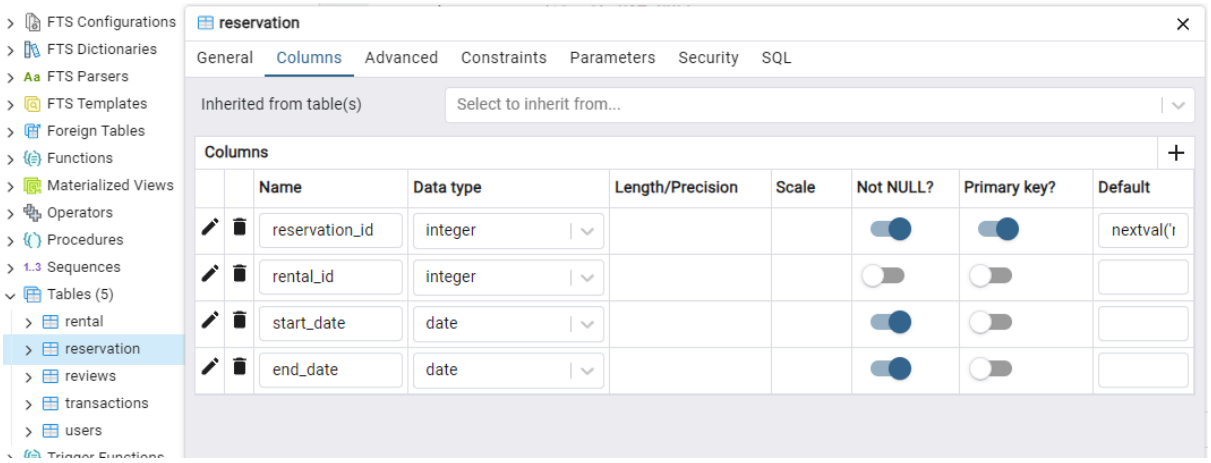
У цієї таблиці немає зовнішніх ключів (FK)

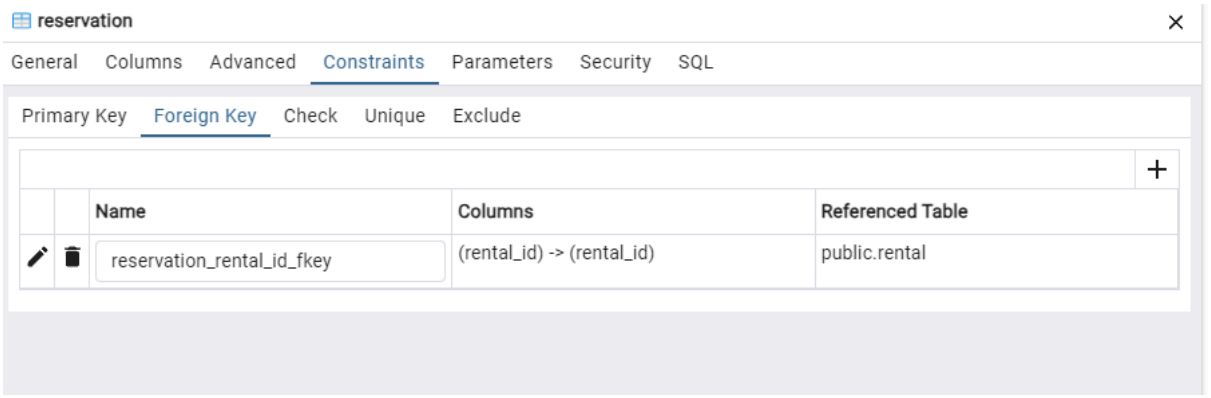
Rental



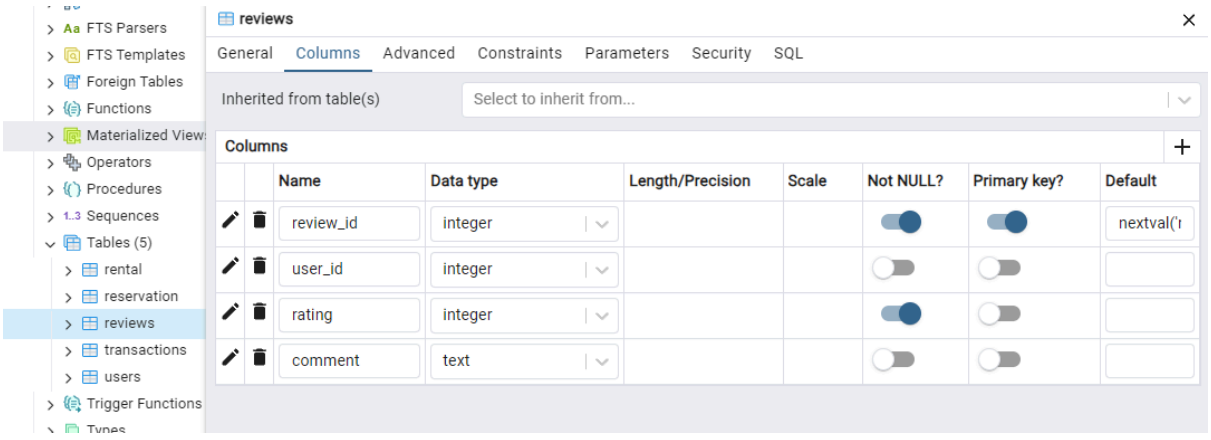


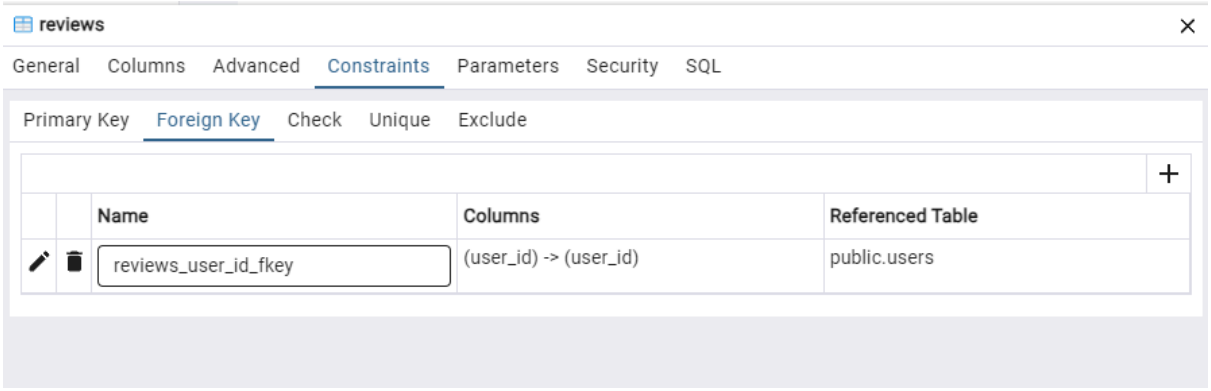
Reservation


FTS Configurations
FTS Dictionaries
FTS Parsers
FTS Templates
Foreign Tables
Functions
Materialized Views
Operators
Procedures
Sequences
Tables (5)
rental
reservation
reviews
transactions
users
reservation
General
Columns
Advanced
Constraints
Parameters
Security
SQL
Inherited from table(s)
Select to inherit from...
Columns
Name
Data type
Length/Precision
Scale
Not NULL?
Primary key?
Default
reservation_id
integer
nextval('r
rental_id
integer
start_date
date
end_date
date

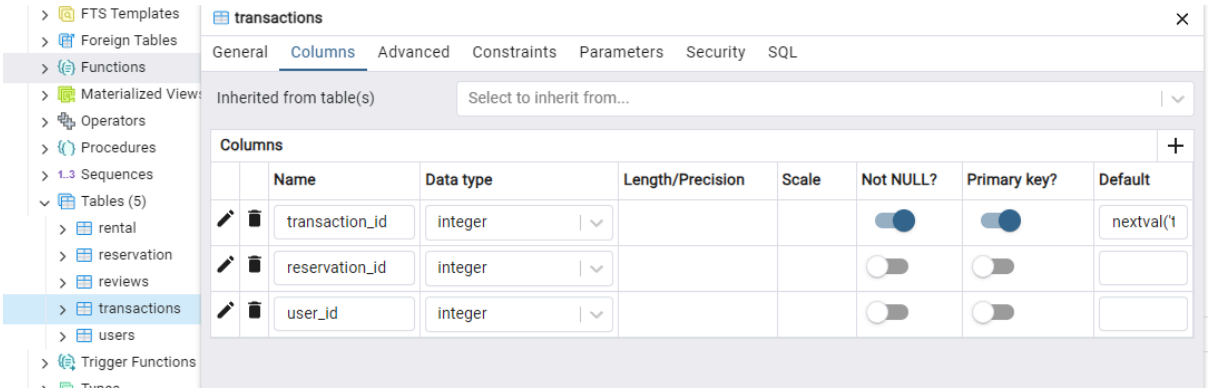

reservation
General
Columns
Advanced
Constraints
Parameters
Security
SQL
Primary Key
Foreign Key
Check
Unique
Exclude
Name
Columns
Referenced Table
reservation_rental_id_fkey
(rental_id) -> (rental_id)
public.rental

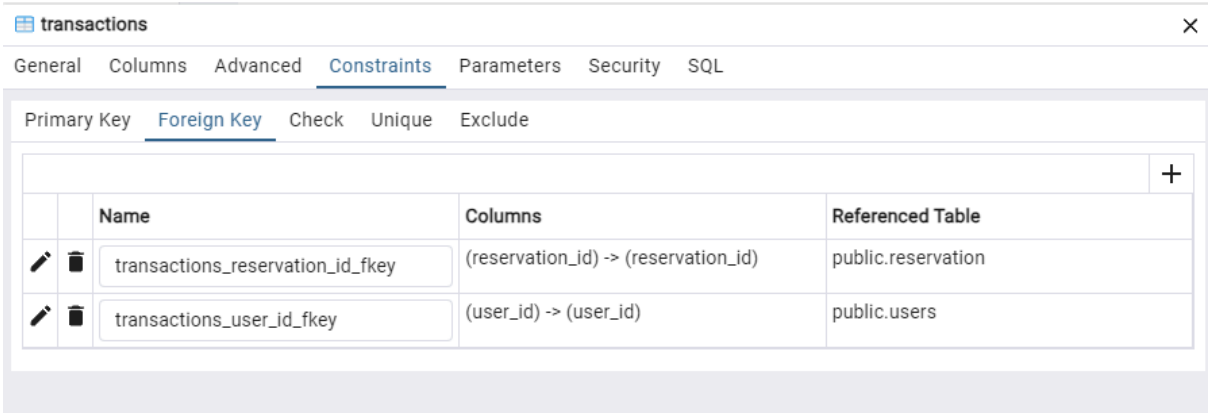
Reviews


FTS Parsers
FTS Templates
Foreign Tables
Functions
Materialized Views
Operators
Procedures
Sequences
Tables (5)
rental
reservation
reviews
transactions
users
Trigger Functions
reviews
General
Columns
Advanced
Constraints
Parameters
Security
SQL
Inherited from table(s)
Select to inherit from...
Columns
Name
Data type
Length/Precision
Scale
Not NULL?
Primary key?
Default
review_id
integer
nextval('r
user_id
integer
rating
integer
comment
text


reviews
General
Columns
Advanced
Constraints
Parameters
Security
SQL
Primary Key
Foreign Key
Check
Unique
Exclude
Name
Columns
Referenced Table
reviews_user_id_fkey
(user_id) -> (user_id)
public.users

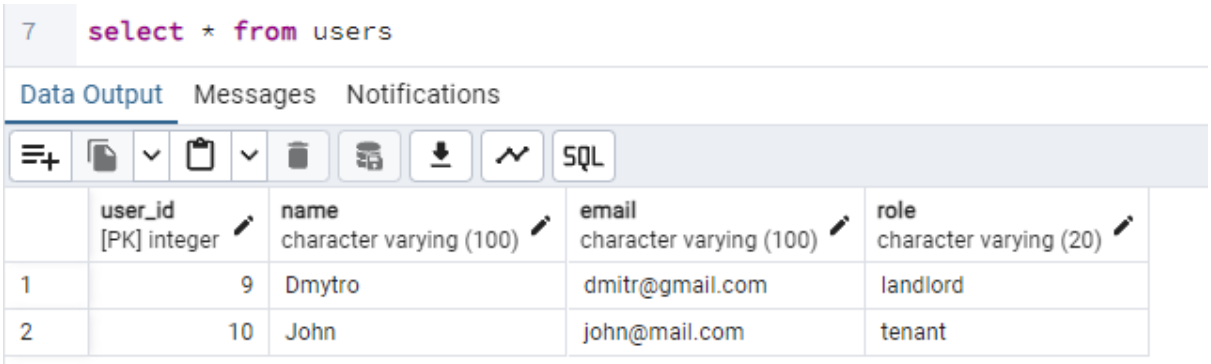
Transactions





Таблиці заповнені даними (уривки знімків екрану pgAdmin4)

Users



Rental

```
14  select * from rental
15
```

Data Output | Messages | Notifications

	rental_id [PK] integer	user_id integer	title character varying (50)	description character varying (1000)	price numeric (12,2)
1	11	9	квартира	Вул. Київська, 12, Київ	50000.00
2	12	9	будинок	Вул. Гагарина, 5, Житомир	100000.00

Reservation

```
21
22  select * from reservation
23
```

Data Output | Messages | Notifications

	reservation_id [PK] integer	rental_id integer	start_date date	end_date date
1	5	11	2024-10-01	2025-09-30
2	6	12	2024-10-15	2025-10-14

Transactions

```
30  select * from transactions
31
```

Data Output | Messages | Notifications

	transaction_id [PK] integer	reservation_id integer	user_id integer
1	5	5	10
2	6	6	10

Reviews

```
37
38  select * from reviews
39
```

Data Output | Messages | Notifications

	review_id [PK] integer	user_id integer	rental_id integer	rating integer	comment text
1	1	10	11	5	Хороші умови та ціна
2	2	10	12	4	

Таблиці в коді SQL

```
-- DROP TABLE public."users" CASCADE;

CREATE TABLE IF NOT EXISTS public."users" (

    user_id SERIAL PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    role VARCHAR(20) CHECK (role IN ('tenant', 'landlord')) NOT NULL
);

TABLESPACE pg_default;
ALTER TABLE public."users"
 OWNER to postgres;

-------------------------------
-- DROP TABLE public."rental" CASCADE;

CREATE TABLE IF NOT EXISTS public."rental" (
    rental_id SERIAL PRIMARY KEY,
    user_id INT REFERENCES Users(user_id)  NOT NULL,
    title VARCHAR(50) NOT NULL,
    description VARCHAR(1000) NOT NULL,
    price DECIMAL(12, 2) NOT NULL
);

TABLESPACE pg_default;
ALTER TABLE public."rental"
 OWNER to postgres;



-------------------------------
-- DROP TABLE public."reservation" CASCADE;

CREATE TABLE IF NOT EXISTS public."reservation" (
    reservation_id SERIAL PRIMARY KEY,
    rental_id INT REFERENCES Rental(rental_id)  NOT NULL,
```

```
    start_date DATE NOT NULL,
    end_date DATE NOT NULL
);

TABLESPACE pg_default;
ALTER TABLE public."reservation"
 OWNER to postgres;



---------------------------------
-- DROP TABLE public."transactions" CASCADE;

CREATE TABLE IF NOT EXISTS public."transactions" (
    transaction_id SERIAL PRIMARY KEY,
    reservation_id INT REFERENCES Reservation(reservation_id) NOT NULL,
    user_id INT REFERENCES Users(user_id) NOT NULL
);

TABLESPACE pg_default;
ALTER TABLE public."transactions"
 OWNER to postgres;



---------------------------------
-- DROP TABLE public."reviews" CASCADE;

CREATE TABLE IF NOT EXISTS public."reviews" (
    review_id SERIAL PRIMARY KEY,
    user_id INT REFERENCES Users(user_id)  NOT NULL,
          rental_id INT REFERENCES Rental(rental_id)  NOT NULL,
    rating INT CHECK (rating BETWEEN 1 AND 5) NOT NULL,
    comment TEXT
);

TABLESPACE pg_default;
ALTER TABLE public."reviews"
 OWNER to postgres;
```

Заповнення таблиць в коді SQL

```
INSERT INTO users (name, email, role)
VALUES
('Dmytro', 'dmitr@gmail.com', 'landlord'),
('John', 'john@mail.com','tenant');

select * from users


INSERT INTO rental (user_id, title, description, price)
VALUES
(9, 'квартира', 'Вул. Київська, 12, Київ', 50000),
(9, 'будинок', 'Вул. Гагарина, 5, Житомир', 100000);

select * from rental


INSERT INTO reservation (rental_id, start_date, end_date)
VALUES
(11, '2024-10-01', '2025-09-30'),
(12, '2024-10-15', '2025-10-14');

select * from reservation

INSERT INTO transactions (reservation_id, user_id)
VALUES
(5, 10),
(6, 10);

select * from transactions

INSERT INTO reviews (user_id, rental_id, rating, comment)
VALUES
(10, 11, 5, 'Хороші умови та ціна'),
(10, 12, 4, '');

select * from reviews
```

Контрольні запитання

1. Призначення діаграм типу «сутність-зв’язок» (ER-діаграм)

ER-діаграми — це графічний інструмент для моделювання даних, що використовується для опису структури бази даних.

Основні призначення ER-діаграм:

- Візуалізація структури даних - вони дозволяють зрозуміти, як різні сутності в базі даних взаємодіють одна з одною через зв’язки.
- Проектування бази даних - визначення атрибутів сутностей та визначення відношень між ними.
- Дозволяють зрозуміти вимоги до структури даних перед реалізацією системи.
- Документація структури бази даних.

2. Основні об'єкти схеми PostgreSQL

- Таблиці (Tables) - основні об'єкти для зберігання даних, що містять рядки (записи) і стовпці (атрибути).
- Індекси (Indexes) - прискорюють пошук і доступ до даних у таблицях.
- Послідовності (Sequences) - генератори чисел, що зазвичай використовуються для створення унікальних ідентифікаторів.
- Види (Views) - віртуальні таблиці, які представляють собою результат виконання запиту.
- Функції (Functions) - програмні модулі, які можуть виконувати певні дії або повертати результат на основі переданих параметрів.
- Тригери (Triggers) - спеціальні процедури, що виконуються автоматично у відповідь на певні події (наприклад, вставку, оновлення або видалення записів).
- Обмеження (Constraints) - правила, що забезпечують цілісність даних (наприклад, первинні та зовнішні ключі, унікальність, перевірки).
- Типи даних (Data Types)
- Простори імен (Schemas): Логічне групування об'єктів бази даних, що дозволяє організовувати об'єкти в окремі блоки.
- Користувачі та ролі (Users and Roles).

- Агрегатні функції (Aggregates) - виконують обчислення над кількома значеннями і повертають одне значення (SUM(), AVG(), MIN(), MAX())
- Збережені процедури (Stored Procedures) - дозволяють виконувати блоки коду в базі даних, які можуть містити логіку, цикли, умовні оператори і виклик інших функцій.

Висновки

У ході виконання лабораторної роботи була розроблена база даних для онлайн-платформа для здачі та оренди нерухомості.

База даних включає такі сутності: "Users", "Rental", "Reservation", "Transaction" та "Review". Кожна з цих сутностей має відповідні атрибути, які дозволяють зберігати та керувати інформацією про користувачів, оголошення оренди, їх рейтинг та бронювання.

ER-діаграма бази даних була підготовлена для візуального відображення взаємозв'язків між сутностями. Нотація Crow’s Foot використана для позначення зв'язків та атрибутів.

Схема бази даних відповідає нормальним формам НФ1, НФ2 та НФ3.

Крім того, схеми бази даних має оновлену версію, яка включає в себе таблицю "Transactions" для відображення зв'язків між користувачами та їхніми бронюваннями.

Звіт містить копії екранів з pgAdmin4, що демонструють властивості стовпців та обмеження, а також вміст таблиць бази даних у PostgreSQL.

Отже, розроблена база даних відповідає поставленим завданням та вимогам.