

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 ІII курсу

групи КВ-02

Кириленко А. В.

Перевірив:

Київ – 2022

*Метою роботи є* здобуття 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 4;
7. Приклад вмісту БД.

**Опис предметної області «Магазин»**

Обрана предметна область передбачає купівлю та оформлення замовлень на товари в магазині. Магазин має декілька відділів. В кожному відділі знаходяться певні категорії товарів. З товарів складаються різноманітні замовлення.

**Опис сутностей предметної області**

Для побудови бази даних для обраної області було виділено сутності, зображені на рисунку 1:

* + - 1. Товари (Goods), з атрибутами: код товару, назва, ціна, код категорії, код відділу, код замовлення. Призначена для збереження інформації про товари, що продаються в магазині.
      2. Категорії товарів (Categories), з атрибутами: код категорії, код відділу, назва. Призначена для збереження інформації про товари та їх категорії з послідуючим додаванням в замовлення.
      3. Відділи (Departments), з атрибутами: код відділу, назва, код товару, кількість товару в наявності. Призначена для збереження інформації про наявність товарів у відділах.
      4. Замовлення (Orders), з атрибутами: код замовлення, код товару, дата замовлення, код клієнта. Призначена для фіксування продажу товарів різних категорій з різних відділів магазину.
      5. Клієнт (Customer), з атрибутами: код клієнта, ім’я, код замовлення. Призначена для збереження інформації про замовника та його замовлення.

**Опис зв’язків між сутностями предметної області**

Сутність «Товари» має зв’язок 1:N по відношенню до сутності «Категорії товарів», тому що в одній категорії товарів можуть бути лише визначені нею товари, але ці ж товари можуть знаходитись і в інших категоріях. Також ця сутність має зв’язок 1:N по відношенню до сутності «Відділ», тому що в одному відділ багато товарів, але один і той самий вид товару може бути лише в одному відділі. А також ця сутність має зв’язок 1:N по відношенню до сутності «Замовлення», тому що в один товар може бути тільки в одному замовленні і одне замовлення може складатися з різних товарів.

Сутність «Категорії товарів» має зв’язок 1:N по відношенню до сутності «Відділи», тому що одна категорія товарів можу знаходитись лише в одному відділі, але один і той самий відділ може включати в себе декілька різних категорій товарів. А також ця сутність має зв’язок 1:N з сутністю «Товари», що описано вище.

Сутність «Відділи» має зв’язок 1:N з сутністю «Товари» та зв’язок 1:N з сутністю «Категорії товарів», що описано вище.

Сутність «Клієнт» має зв’язок 1:N по відношенню до сутності «Замовлення», тому що одне замовлення може бути лише в одного клієнта, але один клієнт може мати декілька замовлень.

**Концептуальна модель предметної області “Кінотеатр”**

Концептуальна модель наведена на рисунку 1.

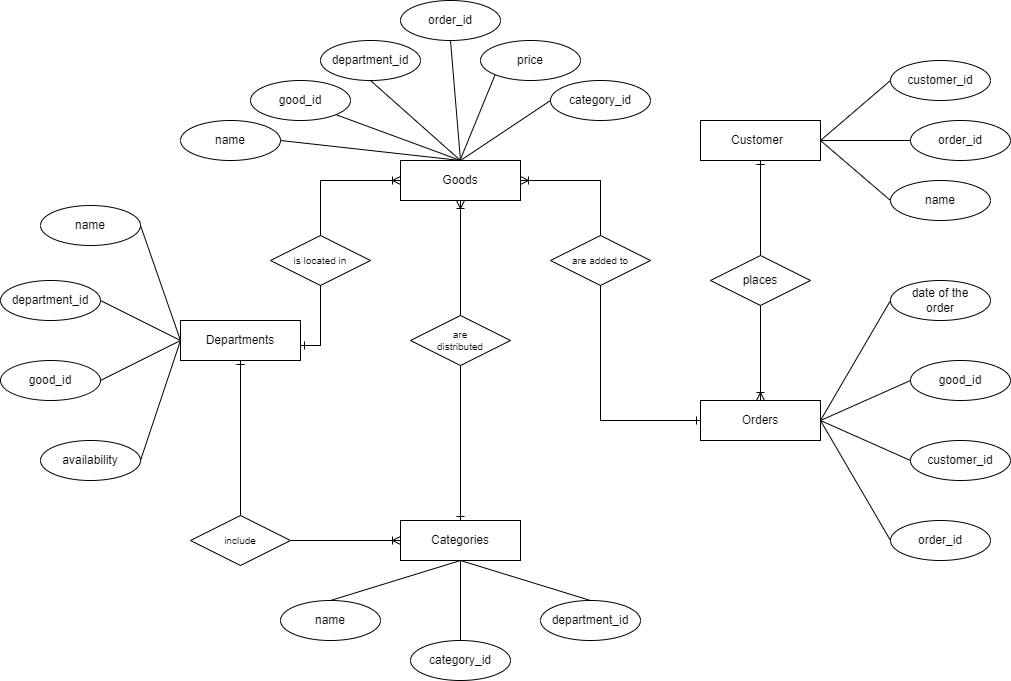


Рисунок 1 - ER-діаграма, побудована за нотацією «Пташина лапка» (інструмент: draw.io)

**Перетворення концептуальної моделі у логічну схему бази даних**

Сутність «Товари» перетворено в таблицю «goods».

Сутність «Категорії товарів» перетворено в таблицю «categories».

Сутність «Відділи» перетворено в таблицю «departments».

Сутність «Замовлення» перетворено в таблицю «orders».

Сутність «Клієнт» перетворено в таблицю «customer».

**Логічна модель (схема) БД «Магазин»**

Логічну модель (схему бази даних наведено на рисунку 2.

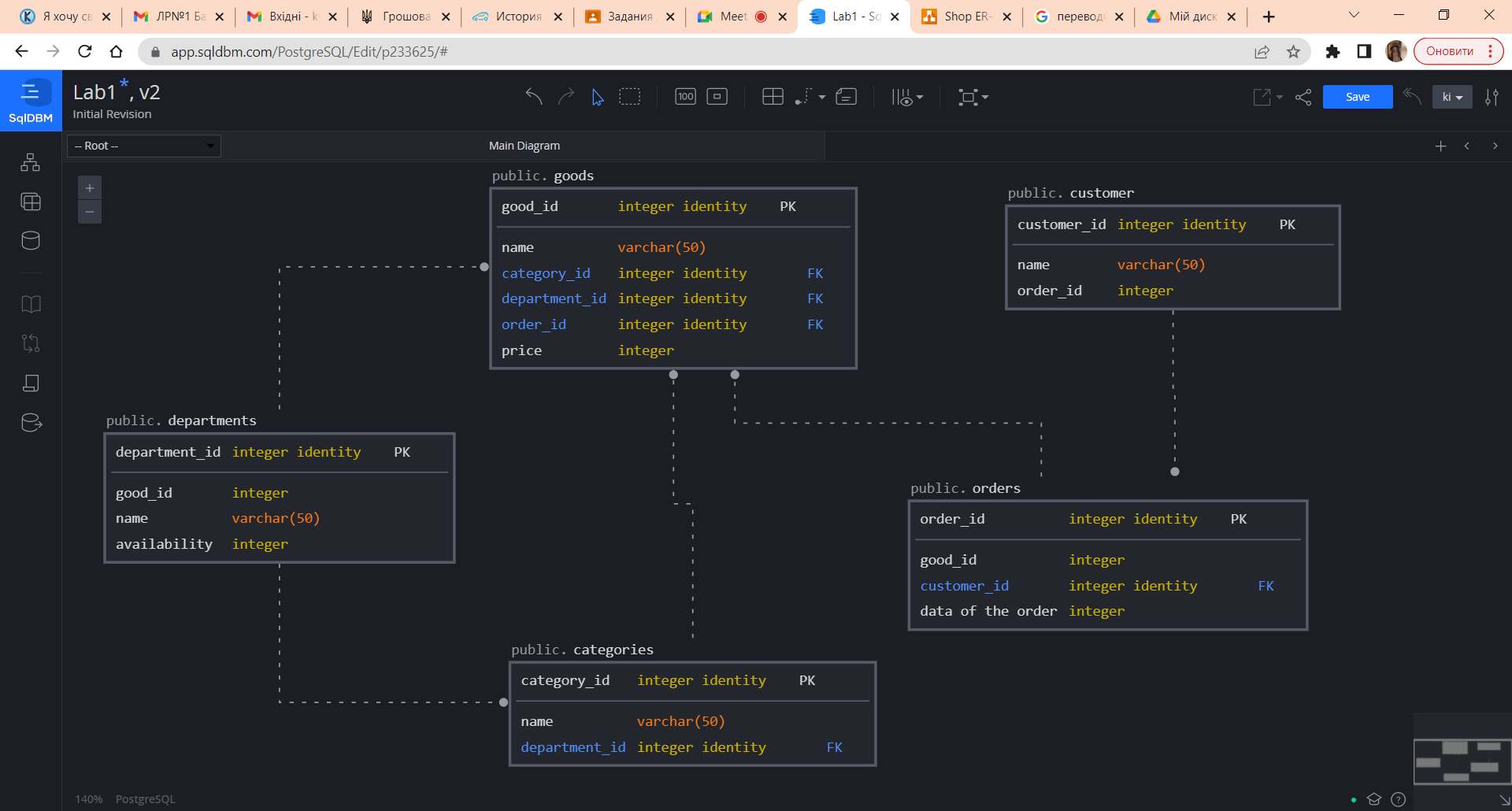


Рисунок 2 - Схема бази даних (інструмент: sqldbm.com)

**Опис об’єктів бази даних у вигляді таблиці**

|  |  |  |
| --- | --- | --- |
| Сутність | Атрибут | Тип атрибуту |
| **goods** – *містить всі товари, що є у продажі* | **good\_id***- унікальний ідентифікатор товару*  **name** *- назва товару*. *Не допускає NULL*  **price** *– вартість товару*. *Не допускає NULL*  **category\_id***- унікальний ідентифікатор категорії товару*  **department\_id***- унікальний ідентифікатор відділу, до якого відноситься товар*  **order\_id***- унікальний ідентифікатор замовлення, в яке додано товар* | **integer** *(числовий)*  **character varying** *(рядок)*  **integer** *(числовий)*  **integer** *(числовий)*  **integer** *(числовий)*  **integer** *(числовий)* |
| **categories** – *містить інформацію про категорії товарів* | **category\_id***- унікальний ідентифікатор категорії товару*  **name** *- назва категорії товару*. *Не допускає NULL*  **department\_id***- унікальний ідентифікатор відділу, до якого відноситься категорія товару* | **integer** *(числовий)*  **character varying** *(рядок)*  **integer** *(числовий)* |
| **departments** - *містить інформацію про відділи* | **department\_id***- унікальний ідентифікатор відділу, до якого відноситься товар*  **name** *- назва відділу*. *Не допускає NULL*  **good\_id***- унікальний ідентифікатор товару, який продається у відділі*  **availability** *– кількість товару в наявності. Не допускає NULL* | **integer** *(числовий)*  **character varying** *(рядок)*  **integer** *(числовий)*  **integer** *(числовий)* |
| **orders** - *містить інформацію про замовлення та його склад* | **order\_id** *- унікальний ідентифікатор замовлення*  **good\_id** *- унікальний ідентифікатор товару, який додано в замовлення*  **date of the order** *– дата оформлення замовлення*. *Не допускає NULL*  **customer\_id**- *унікальний ідентифікатор клієнта, який зробив замовлення* | **integer** *(числовий)*  **integer** *(числовий)*  **date** *(дата)*  **integer** *(числовий)* |
| **customer –** *містить інформацію про клієнта, який зробив замовлення* | **customer\_id** – *унікальний ідентифікатор клієнта*  **name** – *ПІБ клієнта. Не допускає NULL*  **order\_id** – *унікальний ідентифікатор замовлення, що зробив клаєнт* | **integer** *(числовий)*  **character varying** *(рядок)*  **integer** *(числовий)* |

**Функціональні залежності для кожної таблиці**

GOODS:

good\_id → name, price, category\_id, department\_id, order\_id

good\_id → name (назва конкретного товару залежить від його коду)

good\_id → price (ціна товару залежить від його коду)

good\_id → category\_id (категорія товару залежить від коду товару)

good\_id → department\_id (відділ товару залежить від коду товару)

good\_id → order\_id (код замовлення залежить від коду товару)

CATEGORIES:

category\_id → name, department\_id

category\_id → name (назва категорії залежить від коду категорії)

category\_id → department\_id (відділ категорії залежить від коду категорії)

DEPARTMENTS:

department\_id → name, good\_id, availability

department\_id → name (назва відділу залежить від коду відділу)

department\_id → good\_id (код товару залежить від коду його відділу)

department\_id → availability (наявність товару залежить від коду відділу)

ORDERS:

order\_id → good\_id, date of the order

order\_id → good\_id (код товару залежить від коду замовлення)

order\_id → date of the order (дата замовлення залежить від його коду)

CUSTOMER:

customer\_id → name, order\_id

customer\_id → name (ПІБ клієнта залежить від коду клієнта)

customer\_id → order\_id (код замовлення залежить від коду клієнта)

**Відповідність схеми бази даних до третьої нормальної форми**

Схема відповідає 1НФ, тому що:

В таблиці немає дубльованих рядків.

В кожній комірці збережено скалярне значення.

В кожному стовпці збережено дані одного типу.

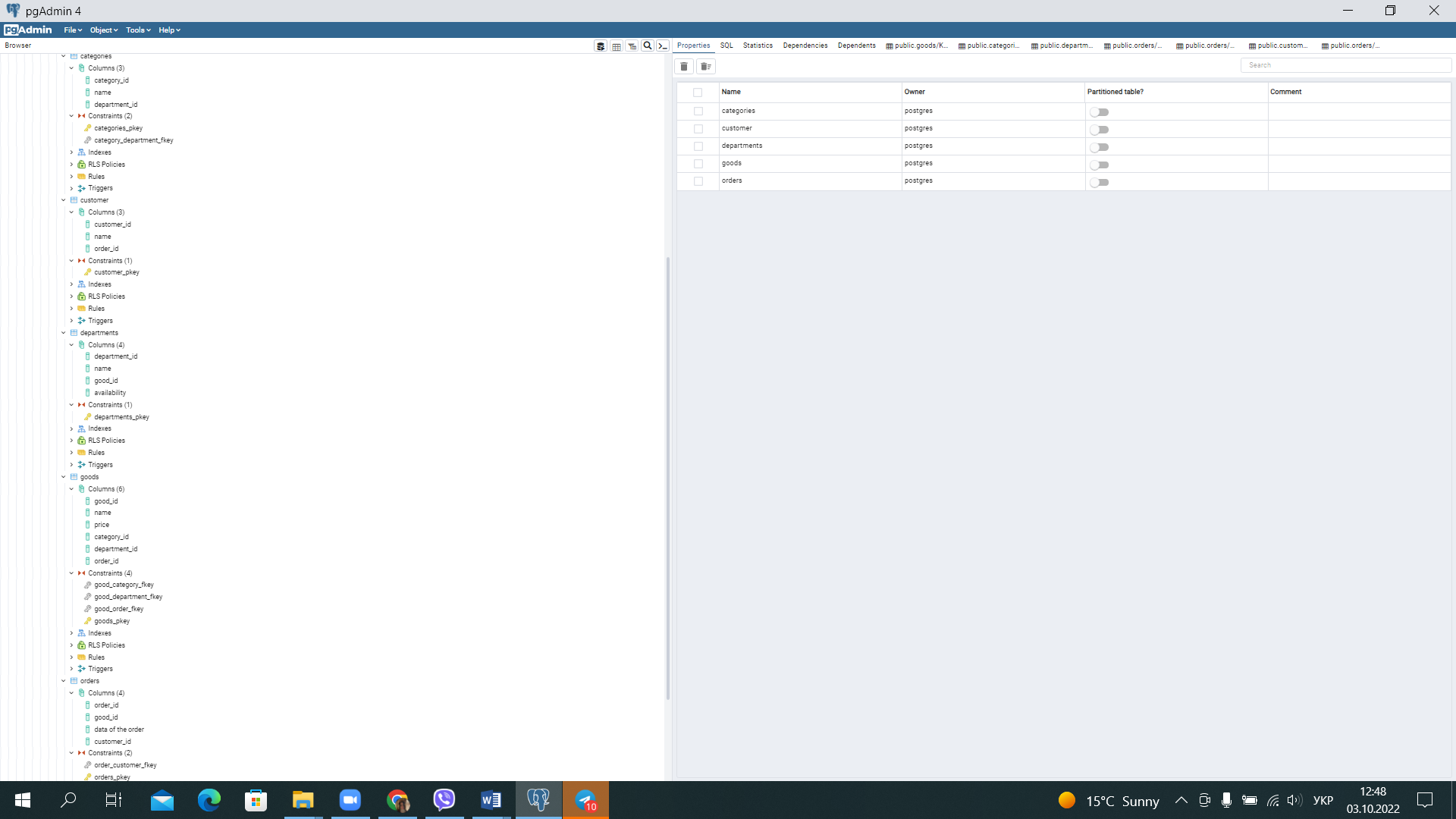
Схема відповідає 2НФ, тому що:

1. Вона відповідає 1НФ.
2. Має первинний ключ, від якого залежать всі не ключовi стовпці таблиці.

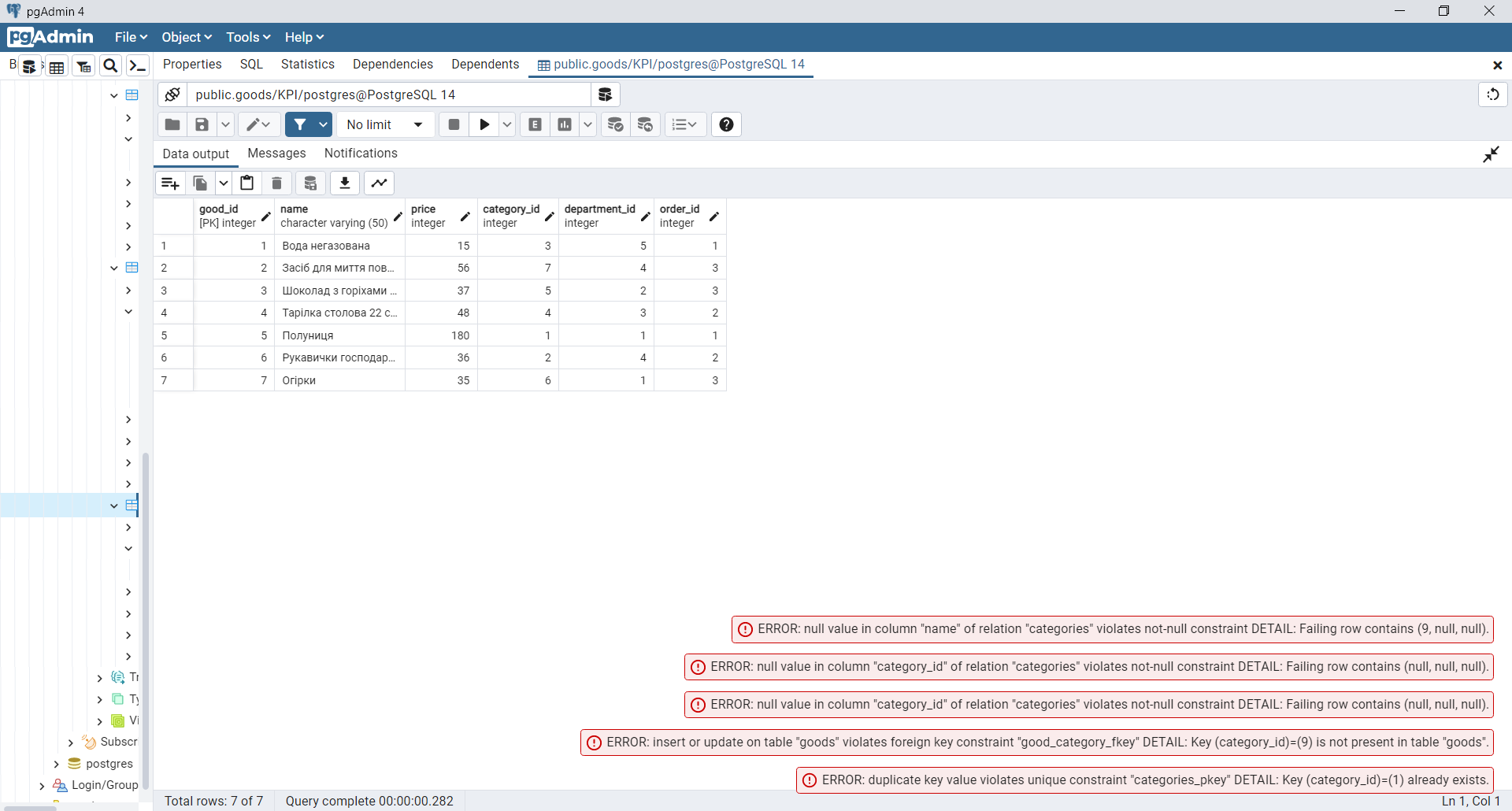
Схема відповідає 3НФ, тому що:

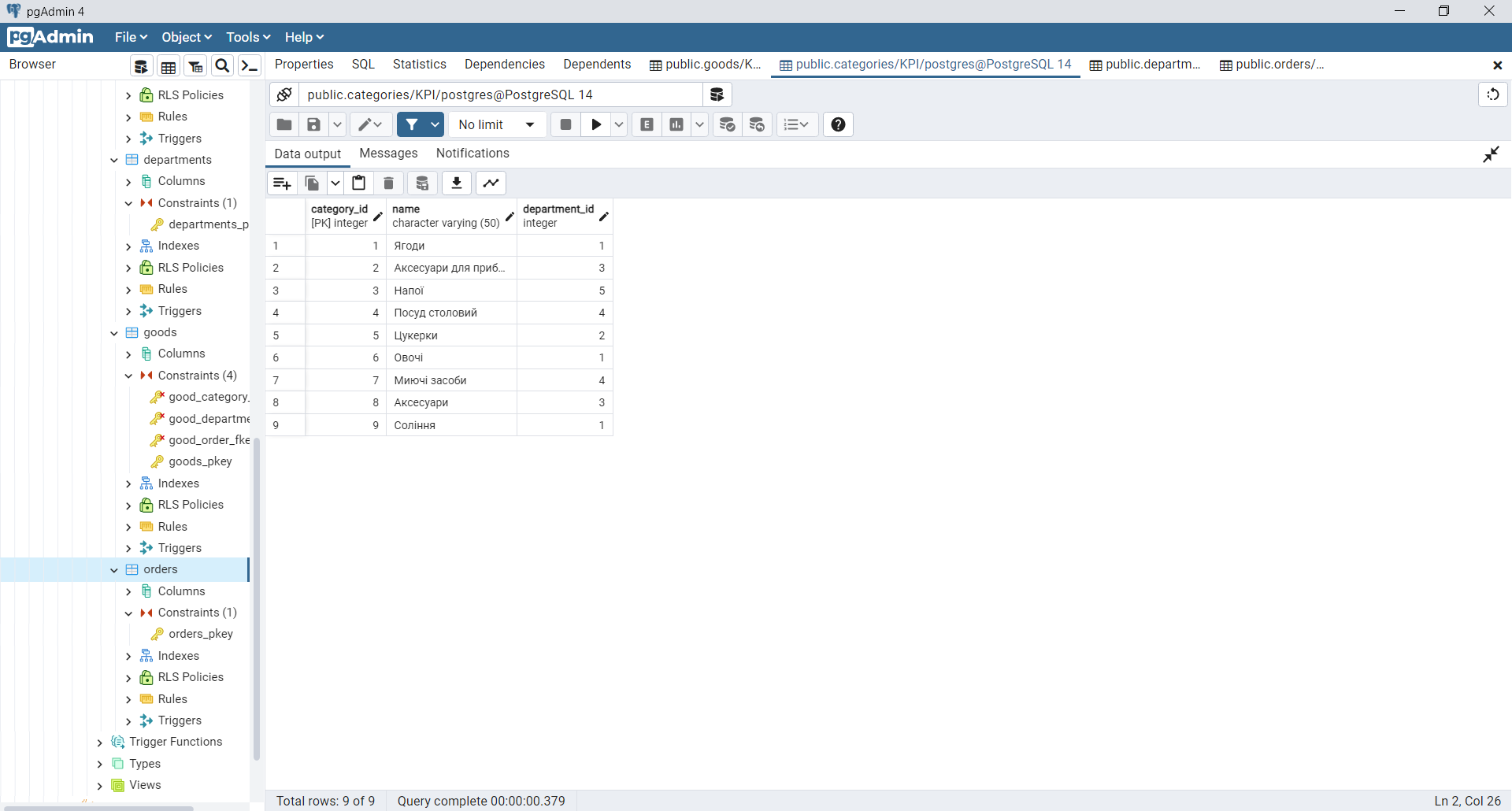
1. Вона відповідає 2НФ.
2. Кожен не ключовий атрибут залежить від первинного ключа, бо всі не ключові атрибути таблиці залежать від всього первинного ключа.

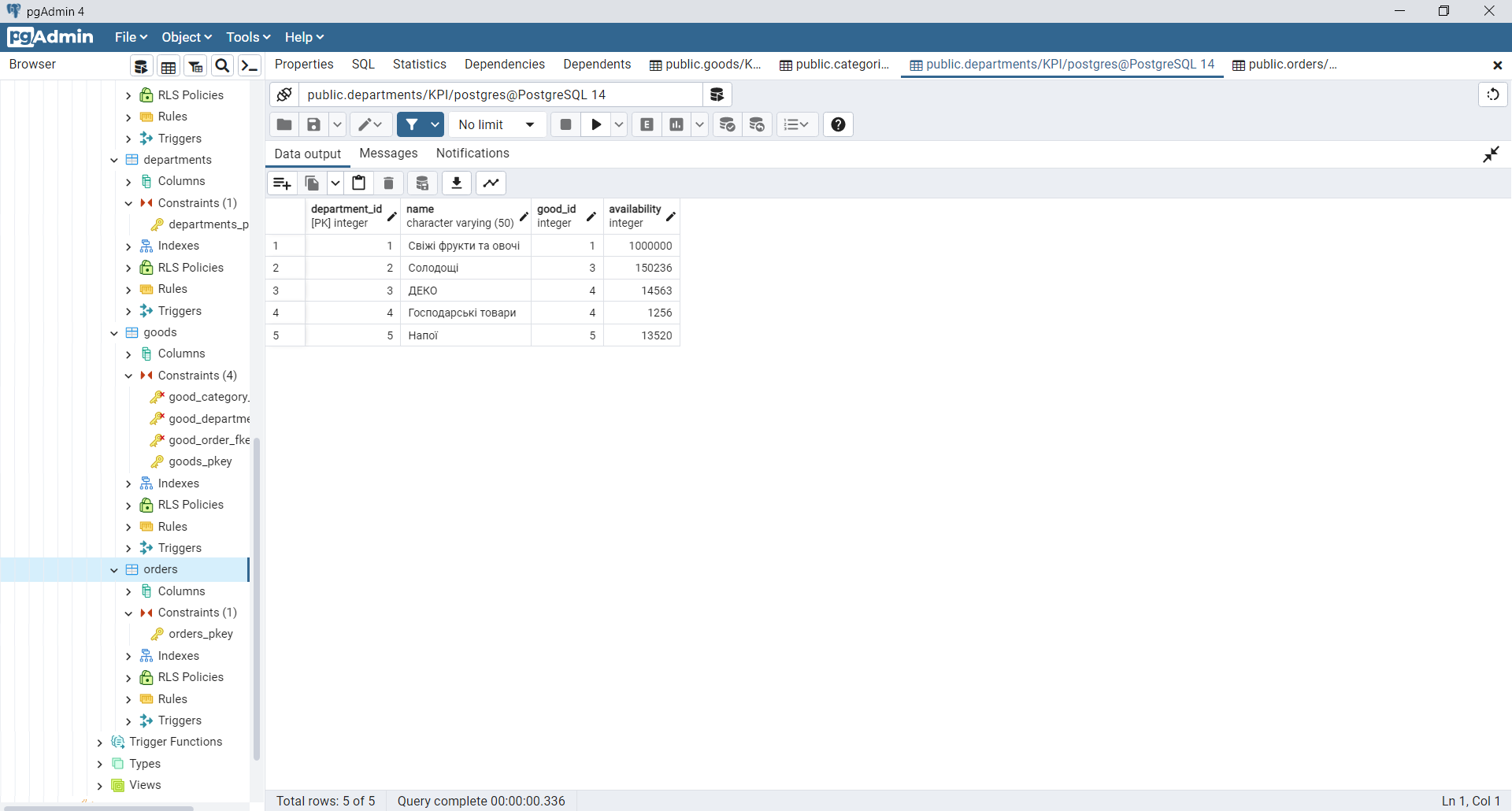
**Фізична модель БД «Магазин» у pgAdmin4**

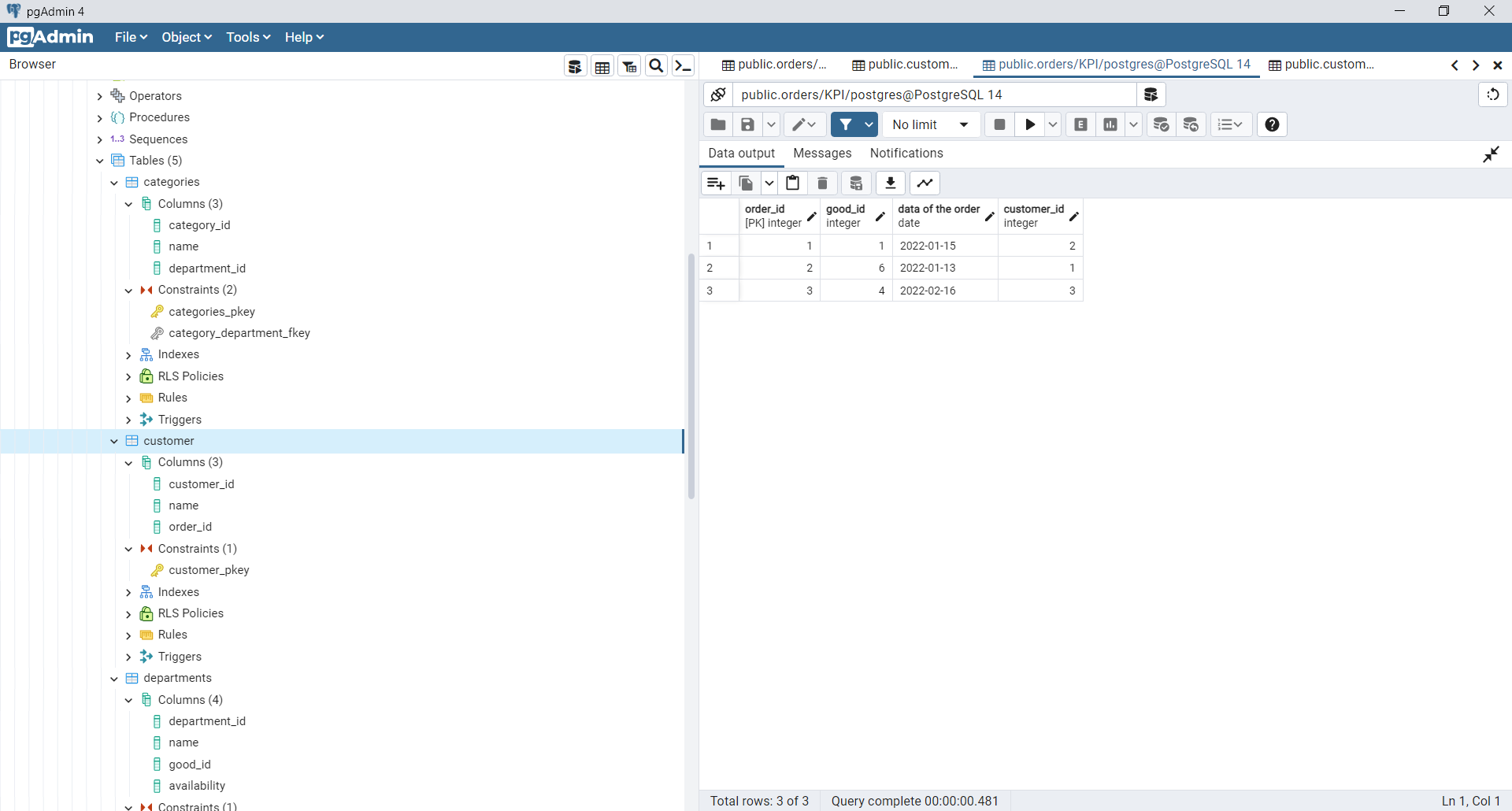


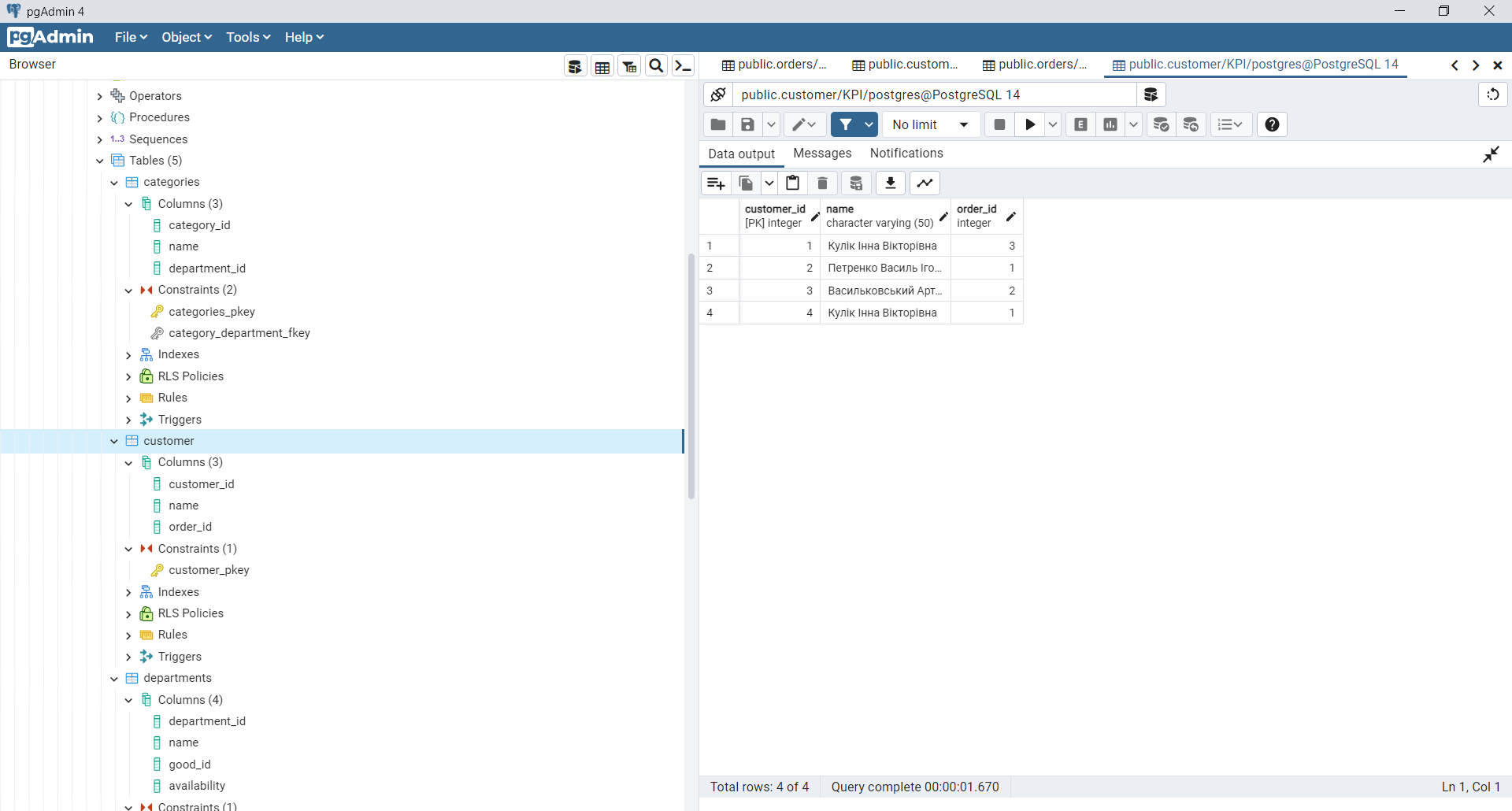
**Фотографії вмісту таблиць**











**SQL-текст опису БД «Магазин»**

SET statement\_timeout = 0;

SET lock\_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 EXTENSION IF NOT EXISTS plpgsql WITH SCHEMA pg\_catalog;

COMMENT ON EXTENSION plpgsql IS 'PL/pgSQL procedural language';

SET default\_tablespace = '';

SET default\_with\_oids = false;

CREATE TABLE public.goods (

good\_id integer NOT NULL,

name character varying(50) NOT NULL,

category\_id integer NOT NULL,

department\_id integer NOT NULL,

order\_id integer NOT NULL,

price integer NOT NULL,

);

ALTER TABLE public.goods OWNER TO postgres;

CREATE TABLE public.categories (

category\_id integer NOT NULL,

name character varying(50) NOT NULL,

department\_id integer NOT NULL,

);

ALTER TABLE public.categories OWNER TO postgres;

CREATE TABLE public.departments (

department\_id integer NOT NULL,

name character varying(50) NOT NULL,

good\_id integer NOT NULL,

availability integer NOT NULL

);

ALTER TABLE public.departments OWNER TO postgres;

CREATE TABLE public.orders (

order\_id integer NOT NULL,

good\_id integer NOT NULL,

data of the order date NOT NULL

customer\_id integer NOT NULL

);

ALTER TABLE public.orders OWNER TO postgres;

CREATE TABLE public.customer (

customer\_id integer NOT NULL,

name character varying(50) NOT NULL,

order\_id integer NOT NULL,

);

ALTER TABLE public.categories OWNER TO postgres;

ALTER TABLE ONLY public.goods

ADD CONSTRAINT goods\_pkey PRIMARY KEY (good\_id);

ALTER TABLE ONLY public.categories

ADD CONSTRAINT categories\_pkey PRIMARY KEY (category\_id);

ALTER TABLE ONLY public.departments

ADD CONSTRAINT departments\_pkey PRIMARY KEY (department\_id);

ALTER TABLE ONLY public.orders

ADD CONSTRAINT orders\_pkey PRIMARY KEY (order\_id);

ALTER TABLE ONLY public.customer

ADD CONSTRAINT customer\_pkey PRIMARY KEY (customer\_id);

ALTER TABLE ONLY public.goods

ADD CONSTRAINT good\_category\_fkey FOREIGN KEY (category\_id) REFERENCES public.categories(category\_id);

ALTER TABLE ONLY public.goods

ADD CONSTRAINT good\_department\_fkey FOREIGN KEY (department\_id) REFERENCES public.departments(department\_id);

ALTER TABLE ONLY public.goods

ADD CONSTRAINT good\_order\_fkey FOREIGN KEY (order\_id) REFERENCES public.orders(order\_id);

ALTER TABLE ONLY public.categories

ADD CONSTRAINT category\_department\_fkey FOREIGN KEY (department\_id) REFERENCES public.departments(department\_id);

ALTER TABLE ONLY public.customer

ADD CONSTRAINT order\_customer\_fkey FOREIGN KEY (customer\_id) REFERENCES public.customer(customer\_id);

REVOKE ALL ON SCHEMA public FROM PUBLIC;

REVOKE ALL ON SCHEMA public FROM postgres;

GRANT ALL ON SCHEMA public TO postgres;

GRANT ALL ON SCHEMA public TO PUBLIC;