

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 ІII курсу

групи КВ-01

Філіпенко Д. О.

Перевірив:

Павловський В. І.

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 занести декілька рядків даних у кожну з таблиць засобами pgAdmin 4.

*Вимоги до ER-моделі:*

1. Сутності моделі предметної галузі мають містити зв’язки типу 1:N або N:M.
2. Кількість сутностей у моделі – 3-4. Кількість атрибутів у кожній сутності: від двох до п’яти.
3. Передбачити наявність зв’язку з атрибутом.

Для побудови ER-діаграм використовувати одну із нотацій: Чена, “Пташиної лапки (Crow’s foot)”, UML.

**Опис предметної галузі «Завод»**

Обрана предметна галузь передбачає замовлення товарів з заводу фірмою. Завод має цеха, кожен цех має виготовляти декілька товарів.

**Опис сутностей предметної галузі**

При проектуванні бази даних «завод» можна виділити наступні сутності:

* + - 1. Завод (Plant), з атрибутами: код заводу (plant\_id), назва (name), адреса (address).
      2. Цеха заводу (Department plant), з атрибутами: код цеху (department\_id), назва (name).
      3. Товар (Product) , з атрибутами: код товару (product\_id), назва (name), ціна (price).
      4. Замовлення (Order) , з атрибутами: код замовлення (order\_id), дата (date).
      5. Замовник (Customer) , з атрибутами: код замовник (customer\_id), ім’я (name).

**Опис зв’язків**

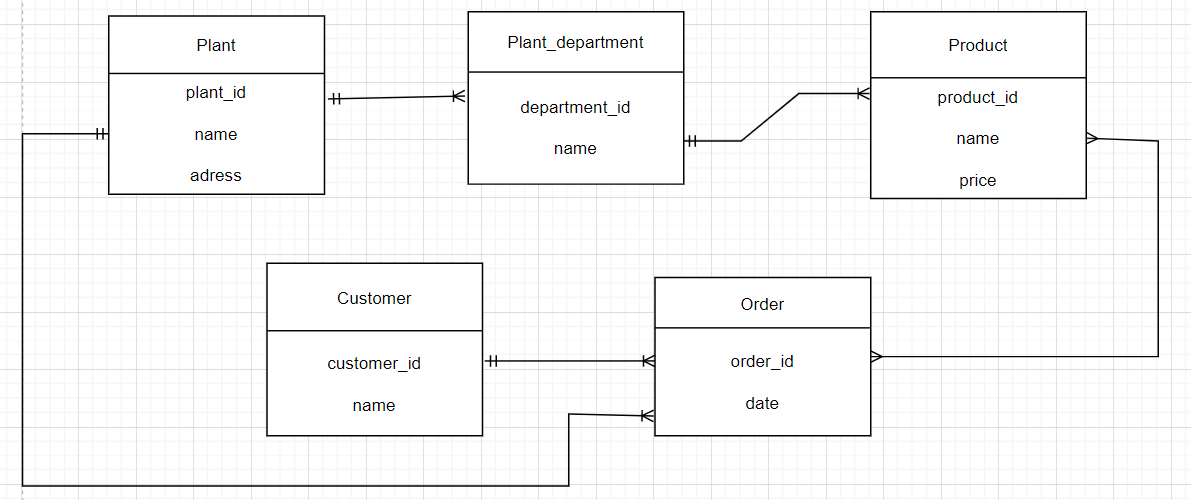
В одному заводі може бути багато цехів, зв’язок 1:N.

У одному цеху може бути багато товарів, зв’язок 1:N.

Багато товарів може міститись у одному замовленні, зв’язок M:N.

На один завод може поступати багато замовлень, зв’язок 1:N.

Один замовник може мати багато замовлень, зв’язок 1:N.

 Рисунок 1. ER-діаграма, побудована за нотацією “Пташиної лапки (Crow’s foot)”

**Перетворення концептуальної моделі у схему баз даних**

Для кожної сутності створюється таблиця. Причому кожному атрибуту сутності відповідає стовпець таблиці.

Сутність “Plant” було перетворено у таблицю “Plant”, а зв’язки 1:N цієї сутності із сутностями “Plant\_department” і “Order” зумовили появу у цих сутностях зовнішніх ключів plant\_id.

Сутність “Plant\_department” було перетворено у таблицю “Plant\_department”, а зв’язок 1:N цієї сутності із “Product ” зумовили появу у цій сутності зовнішнього ключа department\_id.

Сутності “Order” та “Product” були перетворені у таблиці з відповідними назвами, а зв’язок між ними (M:N) зумовив появу додаткової таблиці “Order\_Product”.

Сутність “Customer” було перетворено у таблицю “Customer”, а зв’язок 1:N цієї сутності із “Order” зумовив появу у цій сутності зовнішнього ключа customer\_id.

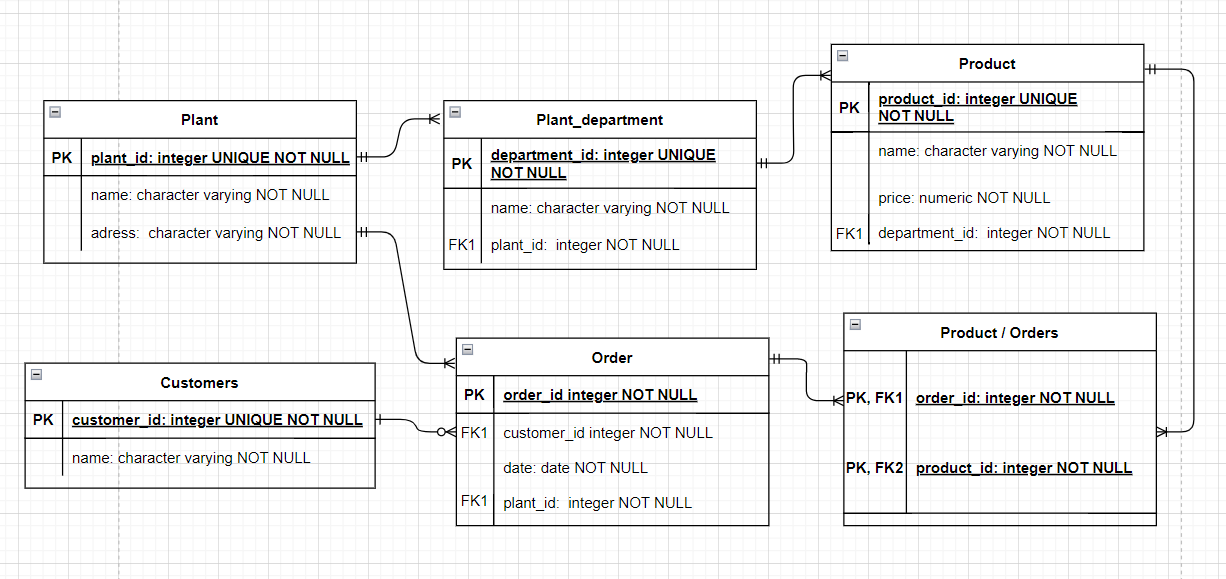


Рисунок 2. Схема бази даних, побудовано у додатку [draw.io](https://app.diagrams.net/).

Після розробки моделі предметної галузі «завод» та перетворення її у схему бази даних, було створено дану базу даних у додатку pgAdmin 4

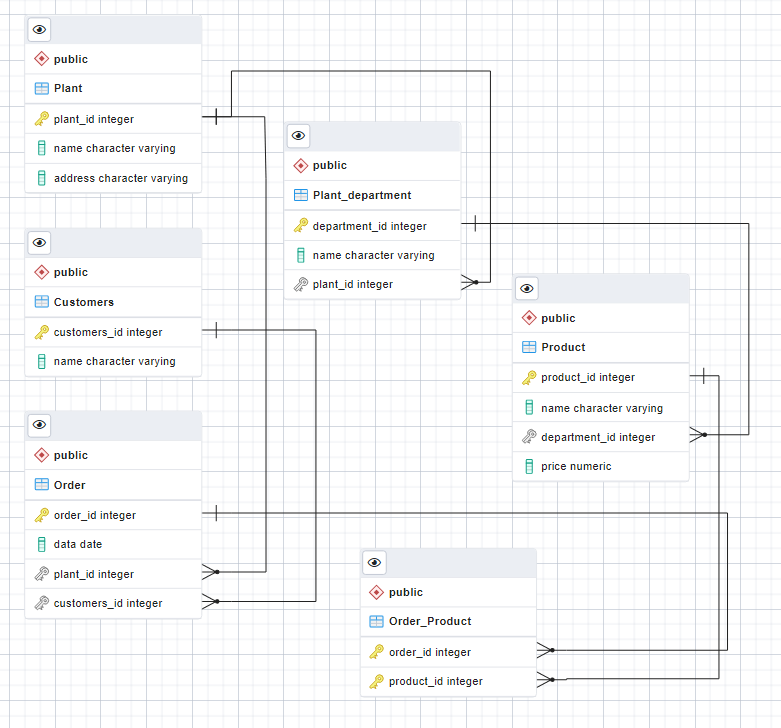


Рисунок 3. Схема бази даних, побудовано у [p](https://app.diagrams.net/)gAdmin 4.

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

Схема відповідає 1НФ, тому що:

1. В таблиці немає дубльованих рядків.
2. В кожній комірці зберігається атомарне значення.
3. В кожному стовпці зберігаються дані одного типу.

Схема відповідає 2НФ, тому що:

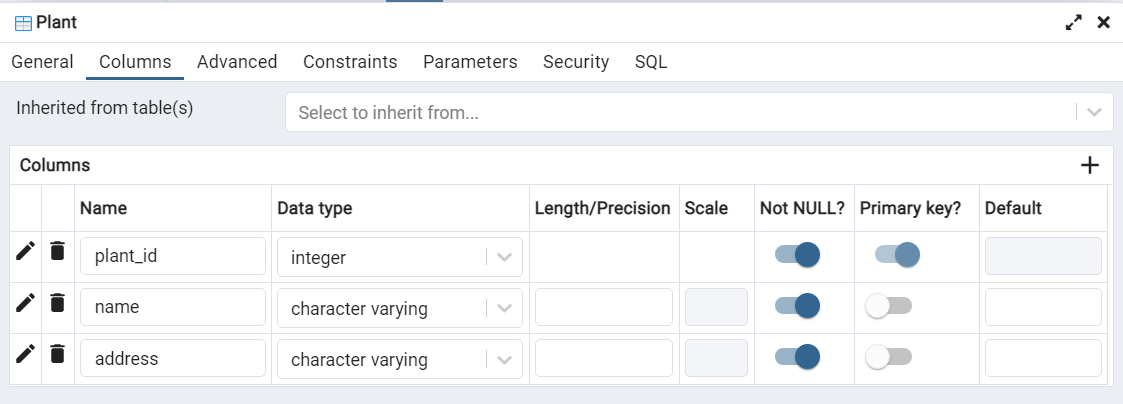
1. Вона відповідає 1НФ.
2. Має первинний ключ, а всі не ключові стовпці таблиці залежать від первинного ключа.

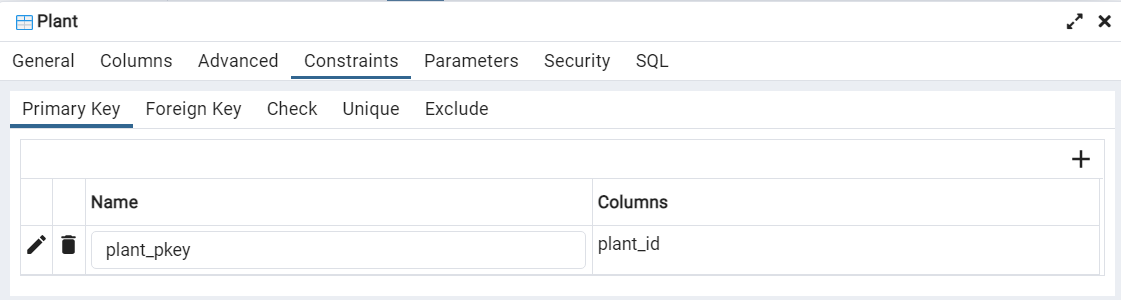
Схема відповідає 3НФ, тому що:

1. Вона відповідає 2НФ.
2. Всі не ключові атрибути таблиці залежать винятково від усього первинного ключа, а не його частини.

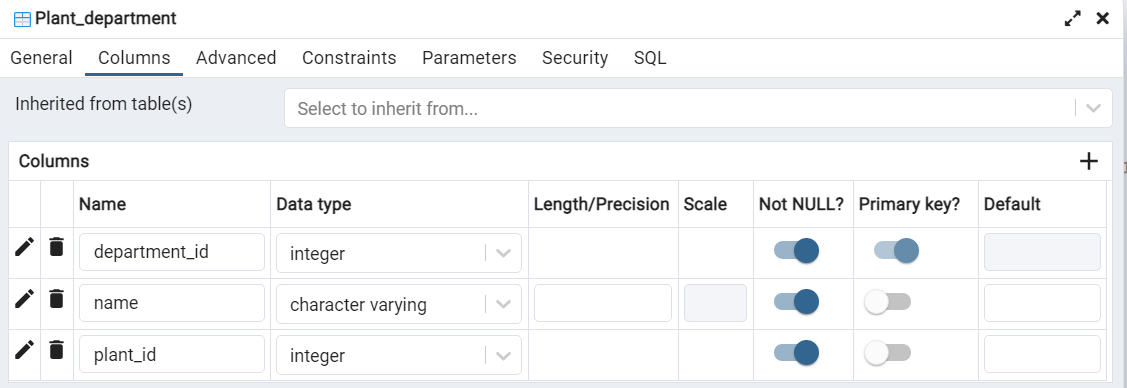
**Таблиці бази даних у pgAdmin 4**

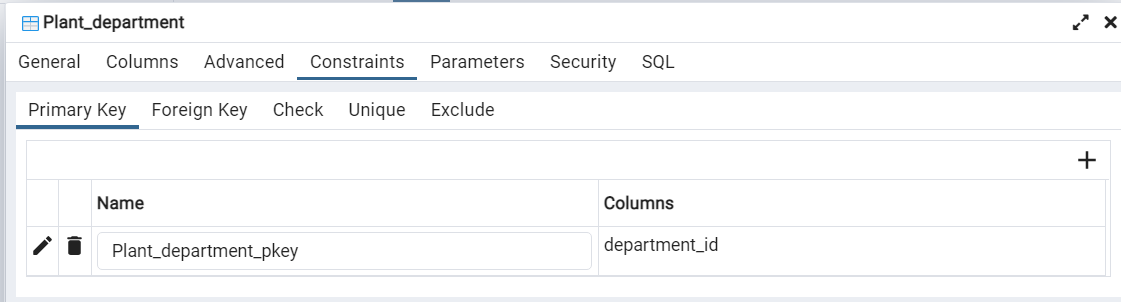
**Plant**

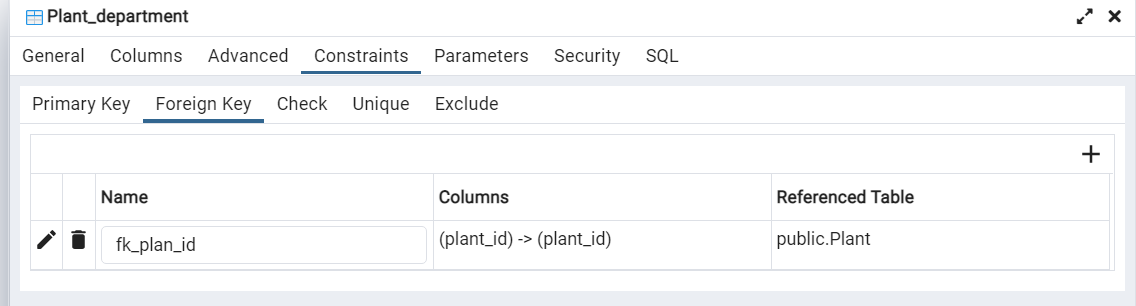
****

****

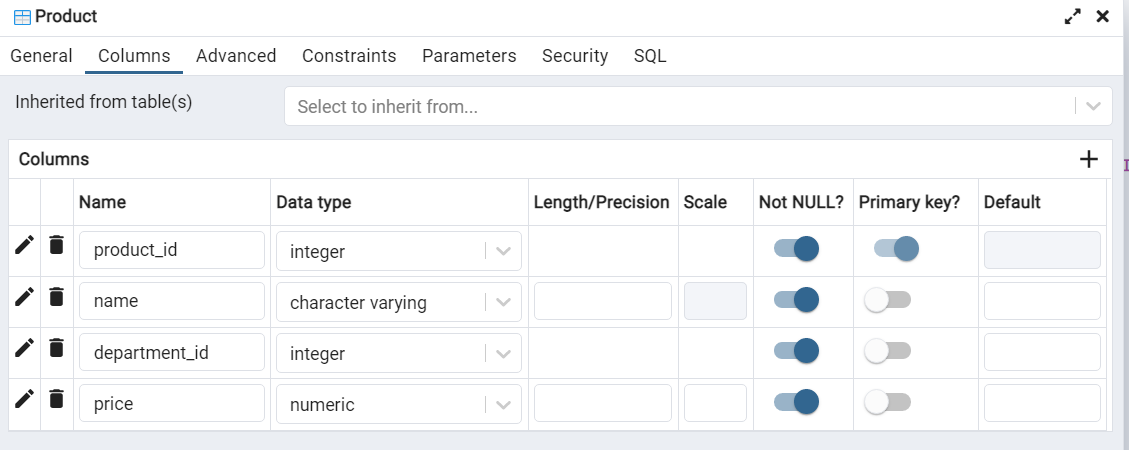
**Plant\_department**

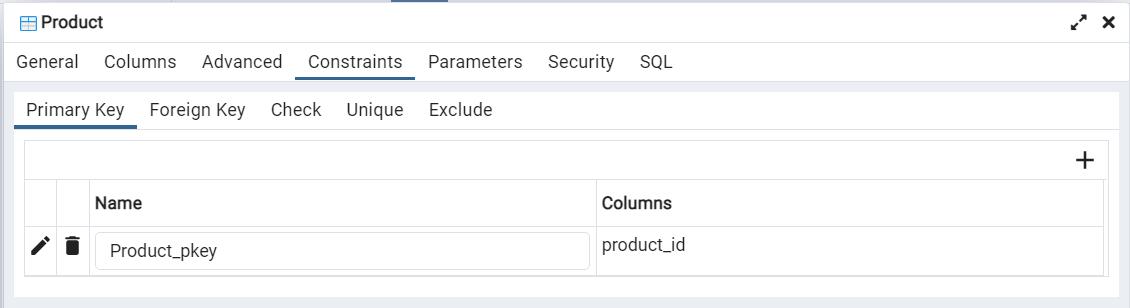
****

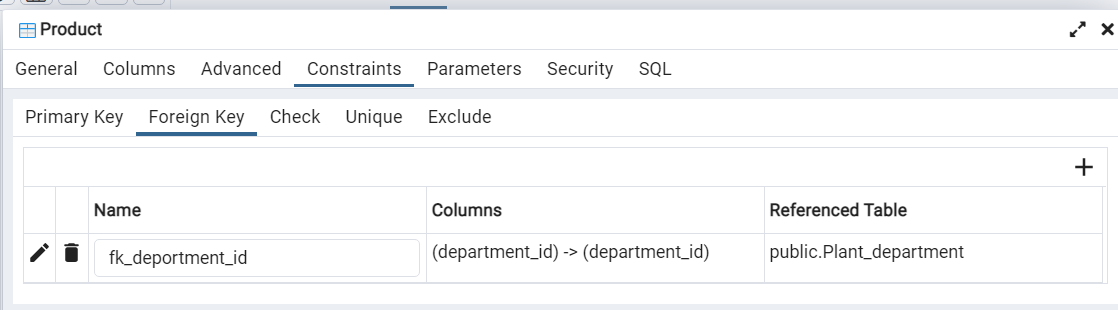
****

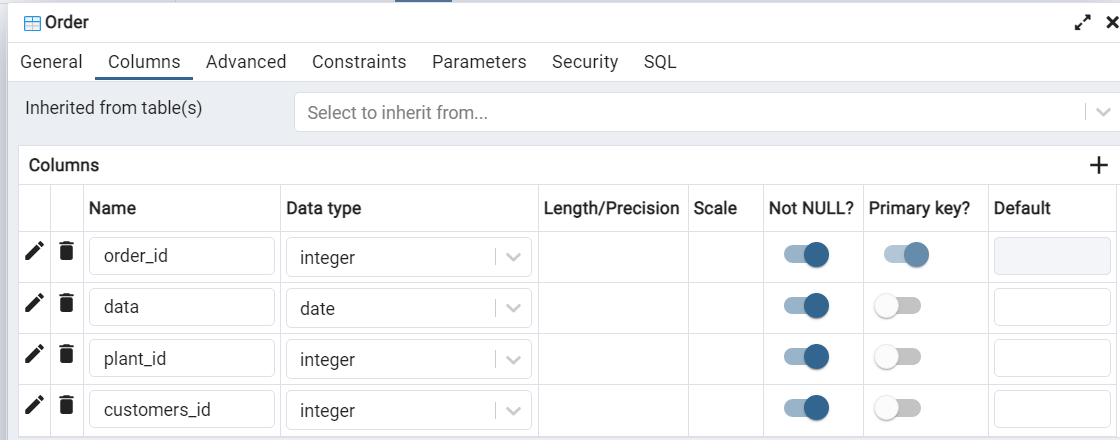
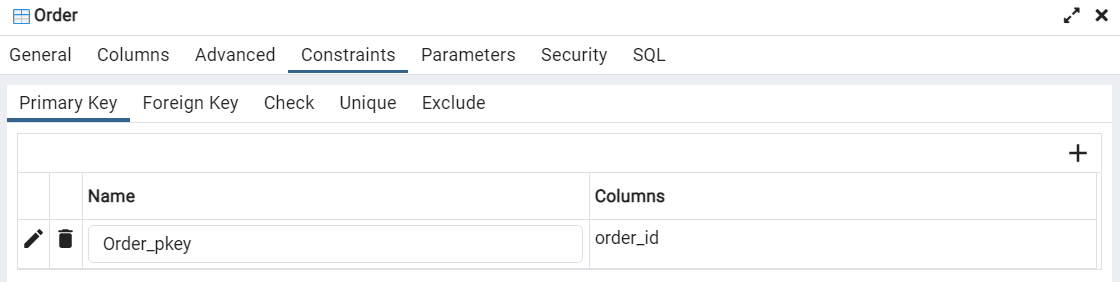
****

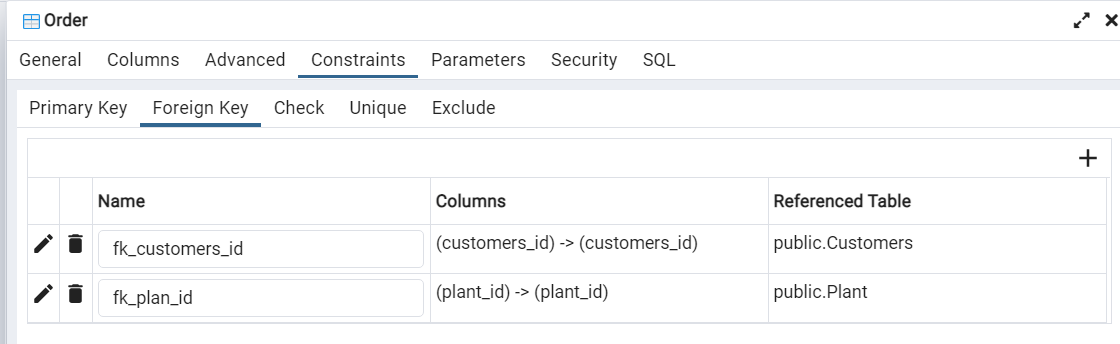
**Product**

****

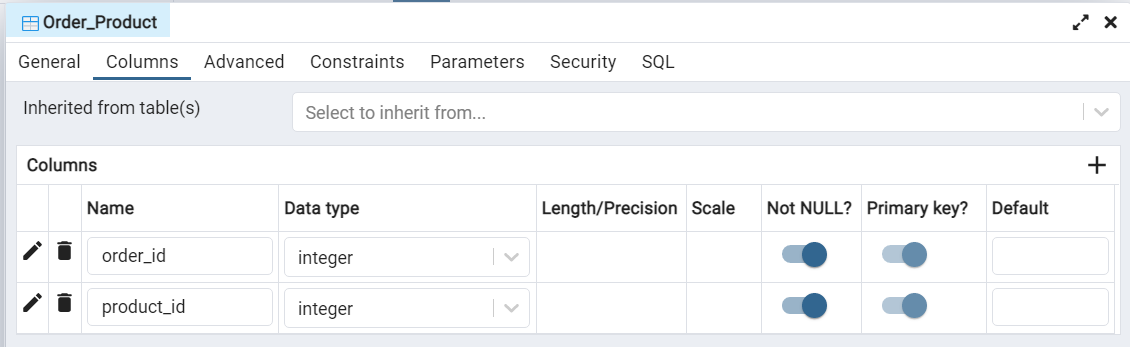
****

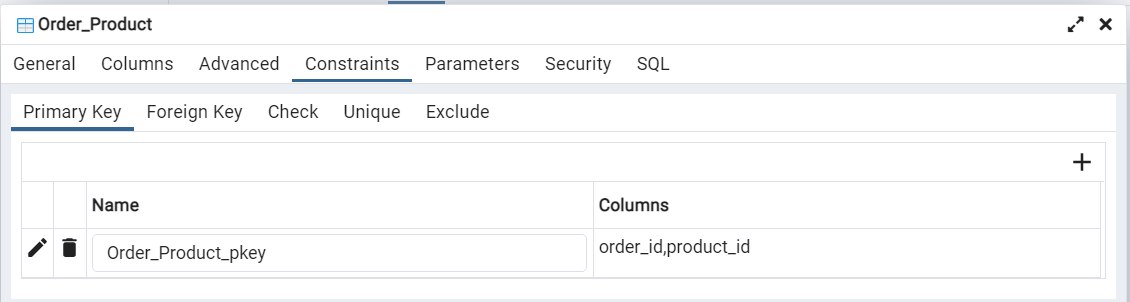
****

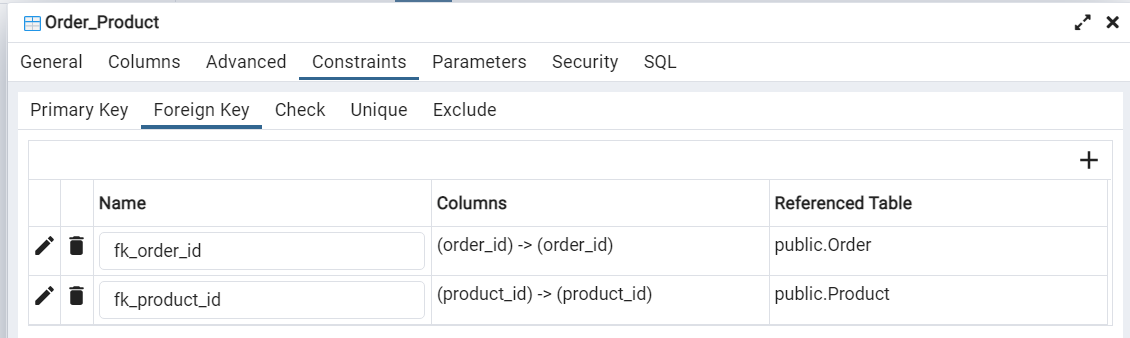
**Order  
**

****

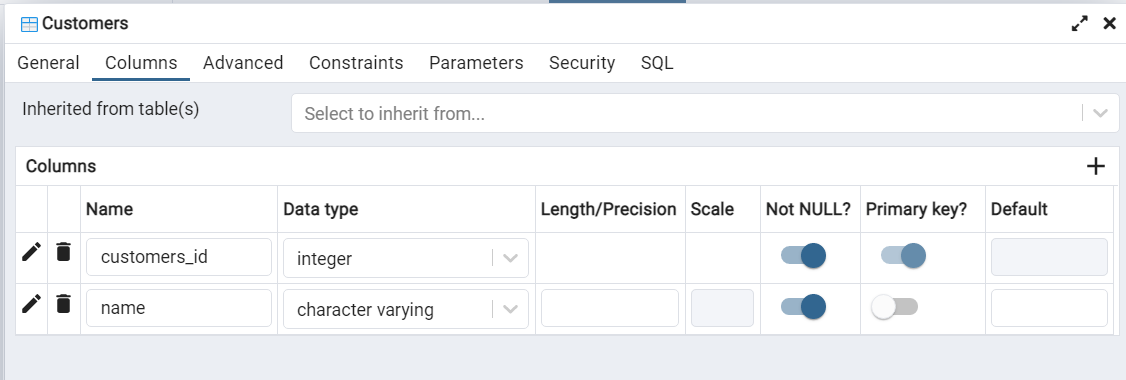
**Order\_Product**

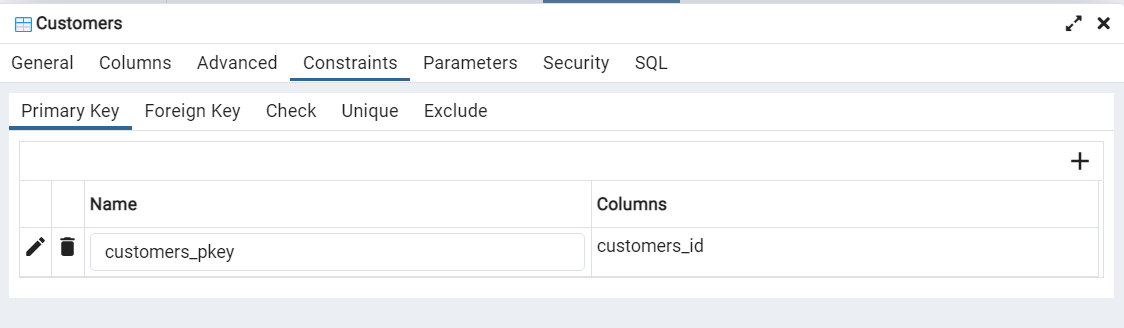
****

****

****

**Customers**

****

****

**SQL-текст опису БД «Магазин»**

-- Database: Plant

-- DROP DATABASE IF EXISTS "Plant";

**CREATE** **DATABASE** "Plant"

**WITH**

OWNER = postgres

ENCODING = 'UTF8'

LC\_COLLATE = 'Russian\_Ukraine.1252'

LC\_CTYPE = 'Russian\_Ukraine.1252'

**TABLESPACE** = pg\_default

**CONNECTION** **LIMIT** = -1

IS\_TEMPLATE = **False**;

-- SCHEMA: public

-- DROP SCHEMA IF EXISTS public ;

**CREATE** **SCHEMA** **IF** **NOT** **EXISTS** **public**

**AUTHORIZATION** postgres;

**COMMENT** **ON** **SCHEMA** **public**

**IS** 'standard public schema';

**GRANT** **ALL** **ON** **SCHEMA** **public** **TO** **PUBLIC**;

**GRANT** **ALL** **ON** **SCHEMA** **public** **TO** postgres;

-- Table: public.Customers

-- DROP TABLE IF EXISTS public."Customers";

**CREATE** **TABLE** **IF** **NOT** **EXISTS** **public**."Customers"

(

customers\_id **integer** **NOT** **NULL** **GENERATED** ALWAYS **AS** **IDENTITY** ( **INCREMENT** 1 **START** 1 **MINVALUE** 1 **MAXVALUE** 2147483647 **CACHE** 1 ),

name **character** **varying** **COLLATE** pg\_catalog."default" **NOT** **NULL**,

**CONSTRAINT** customers\_pkey **PRIMARY** **KEY** (customers\_id)

)

**TABLESPACE** pg\_default;

**ALTER** **TABLE** **IF** **EXISTS** **public**."Customers"

OWNER **to** postgres;

-- Table: public.Order

-- DROP TABLE IF EXISTS public."Order";

**CREATE** **TABLE** **IF** **NOT** **EXISTS** **public**."Order"

(

order\_id **integer** **NOT** **NULL** **GENERATED** ALWAYS **AS** **IDENTITY** ( **INCREMENT** 1 **START** 1 **MINVALUE** 1 **MAXVALUE** 2147483647 **CACHE** 1 ),

**data** **date** **NOT** **NULL**,

plant\_id **integer** **NOT** **NULL**,

customers\_id **integer** **NOT** **NULL**,

**CONSTRAINT** "Order\_pkey" **PRIMARY** **KEY** (order\_id),

**CONSTRAINT** fk\_customers\_id **FOREIGN** **KEY** (customers\_id)

**REFERENCES** **public**."Customers" (customers\_id) **MATCH** SIMPLE

**ON** **UPDATE** **NO** **ACTION**

**ON** **DELETE** **NO** **ACTION**

**NOT** VALID,

**CONSTRAINT** fk\_plan\_id **FOREIGN** **KEY** (plant\_id)

**REFERENCES** **public**."Plant" (plant\_id) **MATCH** SIMPLE

**ON** **UPDATE** **NO** **ACTION**

**ON** **DELETE** **NO** **ACTION**

**NOT** VALID

)

**TABLESPACE** pg\_default;

**ALTER** **TABLE** **IF** **EXISTS** **public**."Order"

OWNER **to** postgres;

-- Table: public.Order\_Product

-- DROP TABLE IF EXISTS public."Order\_Product";

**CREATE** **TABLE** **IF** **NOT** **EXISTS** **public**."Order\_Product"

(

order\_id **integer** **NOT** **NULL**,

product\_id **integer** **NOT** **NULL**,

**CONSTRAINT** "Order\_Product\_pkey" **PRIMARY** **KEY** (order\_id, product\_id),

**CONSTRAINT** fk\_order\_id **FOREIGN** **KEY** (order\_id)

**REFERENCES** **public**."Order" (order\_id) **MATCH** SIMPLE

**ON** **UPDATE** **NO** **ACTION**

**ON** **DELETE** **NO** **ACTION**,

**CONSTRAINT** fk\_product\_id **FOREIGN** **KEY** (product\_id)

**REFERENCES** **public**."Product" (product\_id) **MATCH** SIMPLE

**ON** **UPDATE** **NO** **ACTION**

**ON** **DELETE** **NO** **ACTION**

)

**TABLESPACE** pg\_default;

**ALTER** **TABLE** **IF** **EXISTS** **public**."Order\_Product"

OWNER **to** postgres;

-- Table: public.Plant

-- DROP TABLE IF EXISTS public."Plant";

**CREATE** **TABLE** **IF** **NOT** **EXISTS** **public**."Plant"

(

plant\_id **integer** **NOT** **NULL** **GENERATED** ALWAYS **AS** **IDENTITY** ( **INCREMENT** 1 **START** 1 **MINVALUE** 1 **MAXVALUE** 2147483647 **CACHE** 1 ),

name **character** **varying** **COLLATE** pg\_catalog."default" **NOT** **NULL**,

address **character** **varying** **COLLATE** pg\_catalog."default" **NOT** **NULL**,

**CONSTRAINT** "Plant\_pkey" **PRIMARY** **KEY** (plant\_id)

)

**TABLESPACE** pg\_default;

**ALTER** **TABLE** **IF** **EXISTS** **public**."Plant"

OWNER **to** postgres;

-- Table: public.Plant\_department

-- DROP TABLE IF EXISTS public."Plant\_department";

**CREATE** **TABLE** **IF** **NOT** **EXISTS** **public**."Plant\_department"

(

department\_id **integer** **NOT** **NULL** **GENERATED** ALWAYS **AS** **IDENTITY** ( **INCREMENT** 1 **START** 1 **MINVALUE** 1 **MAXVALUE** 2147483647 **CACHE** 1 ),

name **character** **varying** **COLLATE** pg\_catalog."default" **NOT** **NULL**,

plant\_id **integer** **NOT** **NULL**,

**CONSTRAINT** "Plant\_department\_pkey" **PRIMARY** **KEY** (department\_id),

**CONSTRAINT** fk\_plan\_id **FOREIGN** **KEY** (plant\_id)

**REFERENCES** **public**."Plant" (plant\_id) **MATCH** SIMPLE

**ON** **UPDATE** **NO** **ACTION**

**ON** **DELETE** **NO** **ACTION**

**NOT** VALID

)

**TABLESPACE** pg\_default;

**ALTER** **TABLE** **IF** **EXISTS** **public**."Plant\_department"

OWNER **to** postgres;

-- Table: public.Product

-- DROP TABLE IF EXISTS public."Product";

**CREATE** **TABLE** **IF** **NOT** **EXISTS** **public**."Product"

(

product\_id **integer** **NOT** **NULL** **GENERATED** ALWAYS **AS** **IDENTITY** ( **INCREMENT** 1 **START** 1 **MINVALUE** 1 **MAXVALUE** 2147483647 **CACHE** 1 ),

name **character** **varying** **COLLATE** pg\_catalog."default" **NOT** **NULL**,

department\_id **integer** **NOT** **NULL**,

price **numeric** **NOT** **NULL**,

**CONSTRAINT** "Product\_pkey" **PRIMARY** **KEY** (product\_id),

**CONSTRAINT** fk\_deportment\_id **FOREIGN** **KEY** (department\_id)

**REFERENCES** **public**."Plant\_department" (department\_id) **MATCH** SIMPLE

**ON** **UPDATE** **NO** **ACTION**

**ON** **DELETE** **NO** **ACTION**

**NOT** VALID

)

**TABLESPACE** pg\_default;

**ALTER** **TABLE** **IF** **EXISTS** **public**."Product"

OWNER **to** postgres;