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-02

Шевченко І.І.

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 III;
7. Приклад вмісту БД.

**Опис предметної області «Сервіс продажу залізничних квитків»**

Обрана предметна галузь передбачає придбання і продаж квитків на потяг. Потягу прописаний певний унікальний номер та час відправлення. Потяг має декілька вагонів, вагони мають номера та належать одному з трьох типів, кожен вагон має місця з номерами.

**Опис сутностей предметної області**

Для побудови бази даних для обраної області було виділено сутності:

1)Потяг, з атрибутами: Номер потягу. Призначено для збереження інформації про наявні потяги.

2) Вагон, з атрибутами: «Тип вагону», «Номер вагону». Призначено для зберігання інформації про наявні вагони в цьому потязі. Атрибут «Тип вагону» має під атрибути «Люкс», «Плацкарт», «Купе» для визначення типу конкретного вагона.

3)Місце, з атрибутами: «Номер місця». Призначений для зберігання номера місця пасажира у конкретному вагоні.

4)Квиток, з атрибутами: «Час відправлення» - призначення для унікалізації саме цієї подорожі і, відповідно, виключення проблеми продажу багатьох квитків на одне місце. «Номер місця», «Номер потягу» та «Номер вагону» зберігаються для подальшого друку квитка.

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

1) Сутність «Потяг» має зв’язок 1:N по відношенню до вагонів, бо один потяг може мати декілька вагонів, але один вагон не може належати декільком потягам одночасно.

2) Сутність «Вагон» має зв’язок 1:N стосовно сутності «Місце», тому що вагон має багато місць, але одне місце не може належати багатьом вагонам одночасно. В загальній моделі без прив’язки до конкретної подорожі «Місце» і «Вагон» мають зв’язок M:N. Тому що багато місць можуть належати багатьом вагонам. У випадку, якщо у кожному вагоні однакова нумерація місць.

3) Сутність «Місце» має зв’язок 1:M стосовно «Квиток», бо під час однієї подорожі (атрибут «Дата і час відправлення») одне місце відповідає одному квитку. Але під час різних подорожей багато квитків можуть бути продані на одне й те ж місце.

4) Сутність «Машиніст» має зв’язок по відношенню до сутності «Потяг» M:N. У багатьох вагонах працюють декілька машиністів одночасно.

**Концептуальна модель предметної області**

**“Сервіс продажу залізничних квитків”**

Концептуальна модель наведена на рисунку 1.

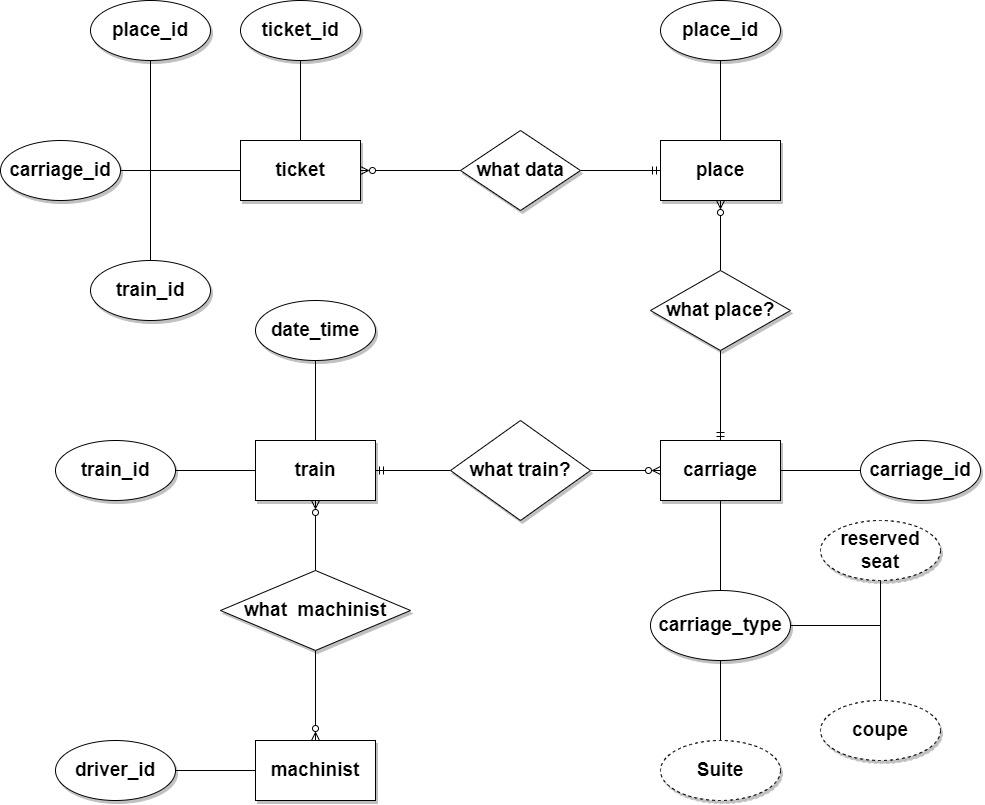
****

Рисунок 1 - ER-діаграма

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

Сутність «Потяг» перетворено на таблицю «train» із зовнішніми ключами «train\_id» і «date\_time».

Сутність «Вагон» перетворено на таблицю «carriage» із зовнішніми ключами «carriage\_type» і «carriage\_id».

Сутність «Місце» перетворено на таблицю «place» із зовнішнім ключем «place\_id».

Сутність «Квиток» перетворено на таблицю «ticket» із зовнішніми ключами «train\_id», «carriage\_id», «place\_id», «ticket\_id».

Сутність «Машиніст» перетворено на таблицю «driver» із зовнішнім ключем «driver\_id»

Створено додаткову таблицю «train/driver» через зв’язок N:M.

**Логічна модель (схема) БД « Сервіс продажу залізничних квитків »**

Логічну модель (схему) бази даних наведено на рисунку 2.

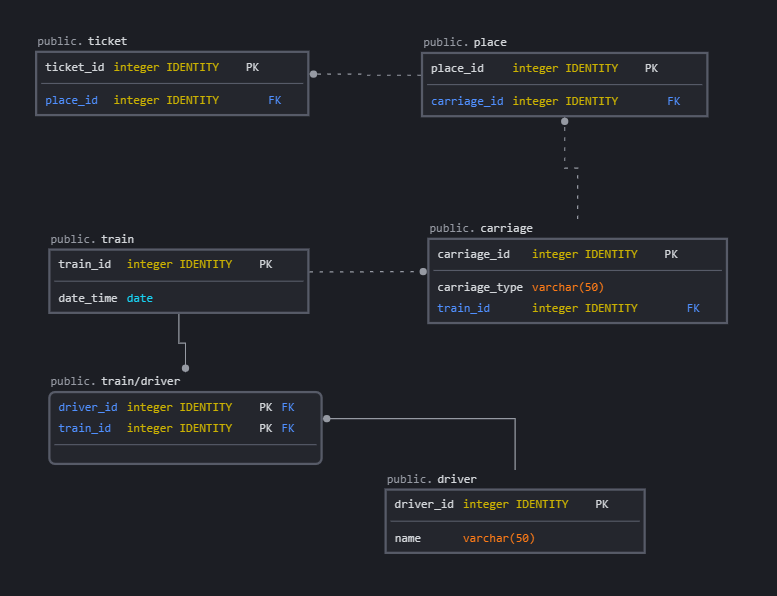
****

Рисунок 2 - Схема бази даних

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

| **Сутність** | **Атрибут** | **Тип атрибуту** |
| --- | --- | --- |
| **Train –** *містить інформацію про потяги* | **Train\_id –** *унікальний ідентифікатор потягу*  **Date\_time –** *дата та час відправлення потягу* | **Integer***(числовий)*  **Date***(дата)* |
| **Carriage** – *містить інформацію про вагони потяга* | **Carriage\_id** – *унікальний ідентифікатор вагону*  **Carriage\_type** – *Тип вагону*  **Train\_id –***унікальний ідентифікатор потягу* | **Integer***(числовий)*  **Char** *(символьний)*  **Integer***(числовий)* |
| **Place** – *містить інформацію про місце пасажира* | **Place\_id** – *унікальний ідентифікатор місця*  **Carriage\_id** – *унікальний ідентифікатор вагону* | **Integer** *(числовий)*  **Integer** *(числовий)* |
| **Ticket** – *містить збірну інформацію, призначену для друку* | **Ticket\_id –** *унікальний ідентифікатор квитка*  **Place\_id** – *унікальний ідентифікатор місця* | **Integer** *(числовий)*  **Integer***(числовий)* |
| **Driver –** *містить інформацію про машиністів* | **Name** – *ім’я машиніста* | **Char***(символьний)* |
| **Train/Driver –** *проміжна таблиця, містить id потягів та водіїв* | **Driver\_id** – *унікальний ідентифікатор машиніста*  **Train\_id -** *унікальний ідентифікатор потягу* | **Integer***(числовий)*  **Integer***(числовий)* |

**Фізична модель БД «Сервіс продажу залізничних квитків»**

**у pgAdmin4**

TRAIN:

Train\_id → date\_time (дата відправлення потягу залежить від його коду)

CARRIAGE:

Carriage\_id → carriage\_type, train\_id

Carriage\_id → carriage\_type (тип вагону залежить від його коду)

Carriage\_id → train\_id (Код потяга залежить від коду вагона)

PLACE:

Place\_id → carriage\_id (код вагону залежить від коду місця)

TICKET:

Ticket\_id → place\_id (Код місця залежить від дати відправлення)

DRIVER:

Driver\_id → name (Ім’я водія потяга)

TRAIN/DRIVER:

Train\_id → Driver\_id

Driver\_id → Train\_id

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

Схема відповідає 1НФ, тому що:

* Значення в кожній комірці є атомарними
* Кожний рядок є унікальним

Схема відповідає 2НФ, тому що:

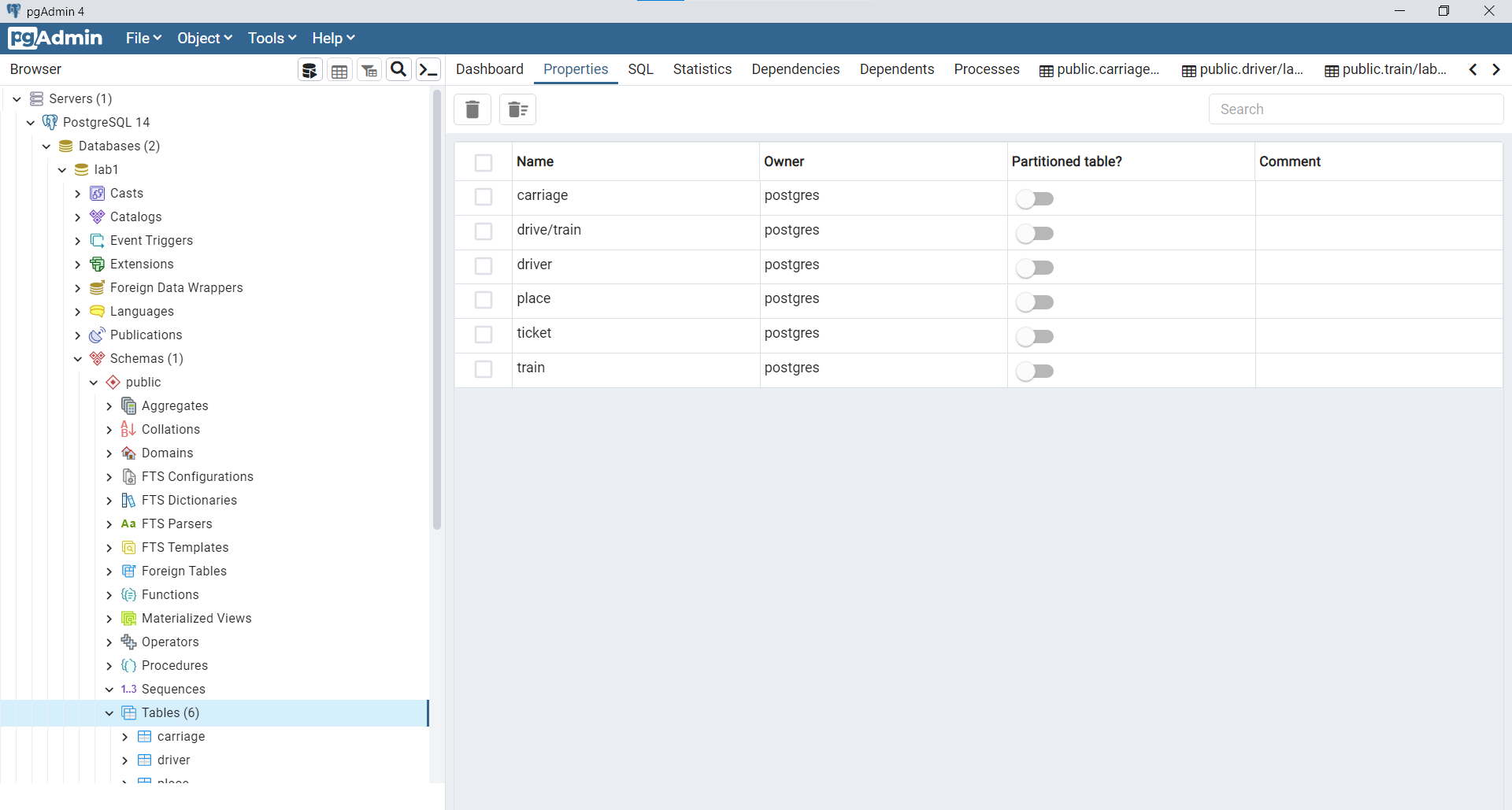
* Дотримується вимог 2НФ
* Неключовий атрибут в списку функціональних залежностей залежить від усього переліку ключа

Схема відповідає 3НФ, тому що:

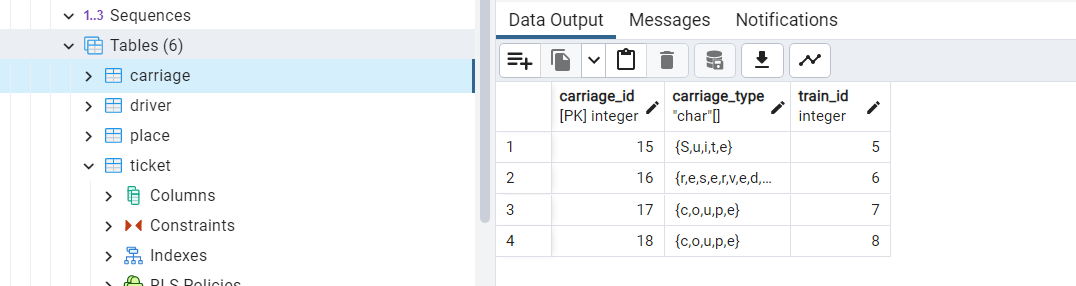
* Відповідає 2НФ
* Кожен неключовий атрибут не є транзитивно залежним від кожного кандидатного ключа

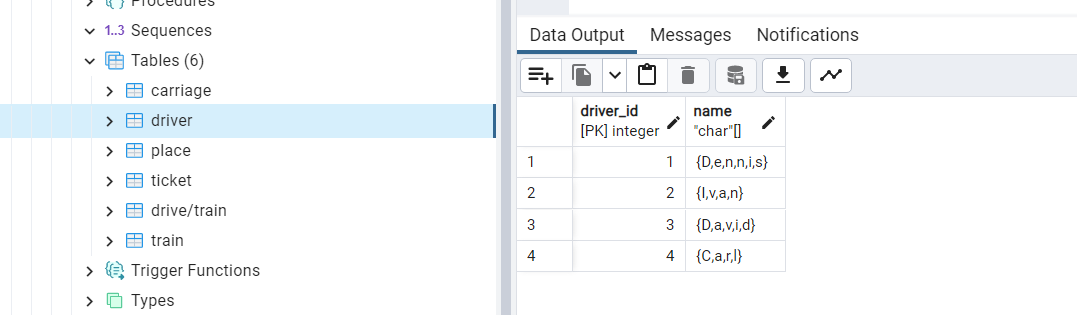
**Фізична модель БД «Сервіс продажу залізничних квитків»**

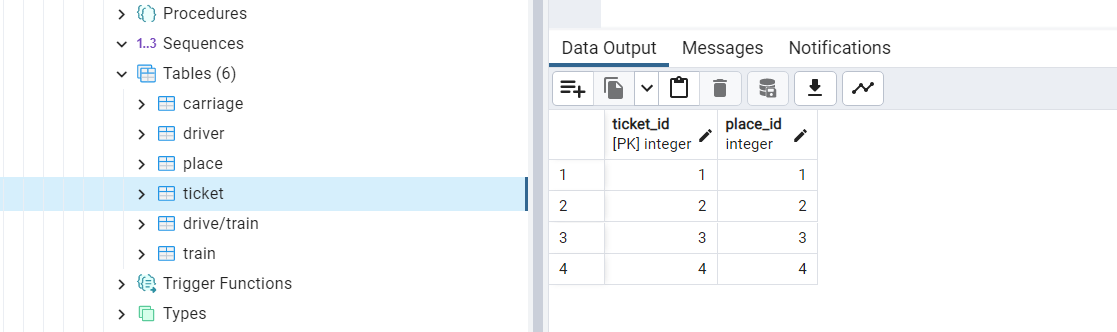
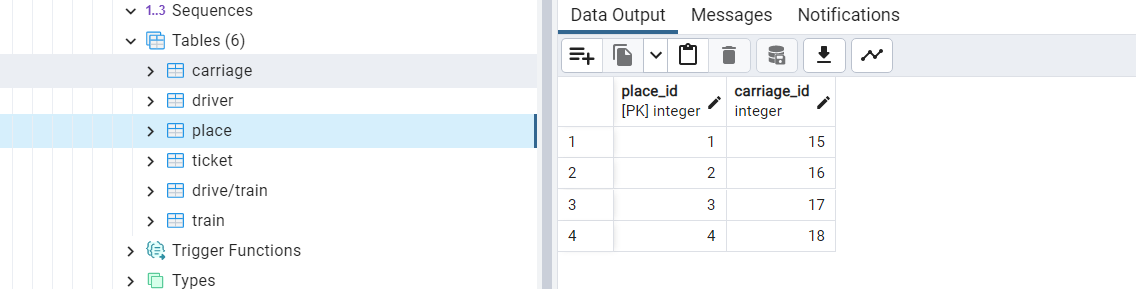
**у pgAdmin4**

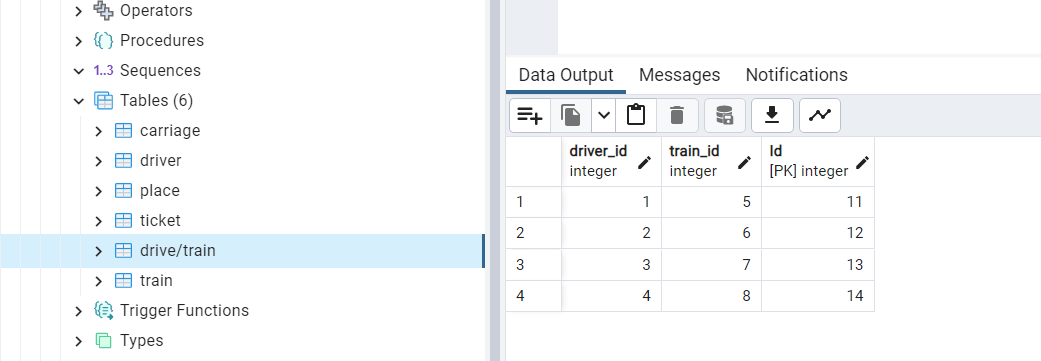
****

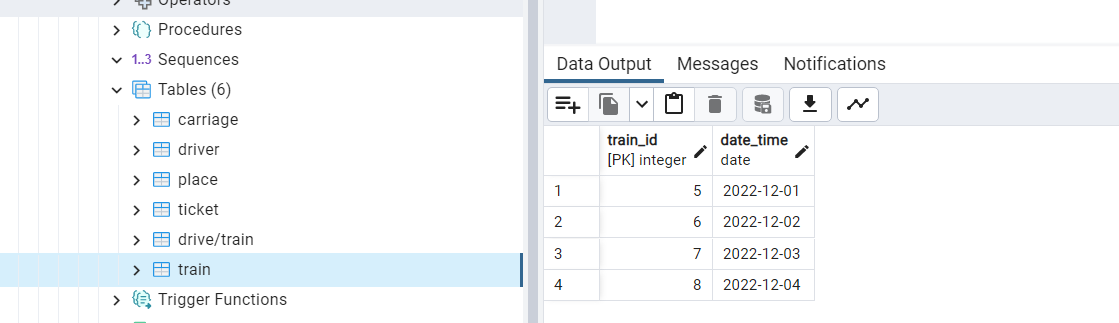
**Фотографії вмісту таблиць**

****

****

****

****

****

**SQL-текст опису БД «Сервіс продажу залізничних квитків»**

-- Database: lab1

-- DROP DATABASE IF EXISTS lab1;

CREATE DATABASE lab1

WITH

OWNER = postgres

ENCODING = 'UTF8'

LC\_COLLATE = 'Russian\_Russia.1251'

LC\_CTYPE = 'Russian\_Russia.1251'

TABLESPACE = pg\_default

CONNECTION LIMIT = -1

IS\_TEMPLATE = False;

CREATE TABLE IF NOT EXISTS public.carriage

(

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

train\_id integer NOT NULL,

carriage\_type "char"[] NOT NULL,

CONSTRAINT carriage\_pkey PRIMARY KEY (carriage\_id),

CONSTRAINT carriage\_train\_id\_fkey FOREIGN KEY (train\_id)

REFERENCES public.train (train\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

NOT VALID

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public.carriage

OWNER to postgres;

CREATE TABLE IF NOT EXISTS public.driver

(

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

name "char"[] NOT NULL,

CONSTRAINT driver\_pkey PRIMARY KEY (driver\_id)

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public.driver

OWNER to postgres;

CREATE TABLE IF NOT EXISTS public."train/driver"

(

id integer NOT NULL GENERATED ALWAYS AS IDENTITY ( INCREMENT 1 START 1 MINVALUE 1 MAXVALUE 2147483647 CACHE 1 ),

train\_id integer NOT NULL,

driver\_id integer NOT NULL,

CONSTRAINT "train/driver\_pkey" PRIMARY KEY (id),

CONSTRAINT "train/driver\_driver\_id\_fkey" FOREIGN KEY (driver\_id)

REFERENCES public.driver (driver\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION,

CONSTRAINT "train/driver\_train\_id\_fkey" FOREIGN KEY (train\_id)

REFERENCES public.train (train\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public."train/driver"

OWNER to postgres;

CREATE TABLE IF NOT EXISTS public.place

(

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

carriage\_id integer NOT NULL,

CONSTRAINT place\_pkey PRIMARY KEY (place\_id),

CONSTRAINT place\_carriage\_id\_fkey FOREIGN KEY (carriage\_id)

REFERENCES public.carriage (carriage\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

NOT VALID

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public.place

OWNER to postgres;

CREATE TABLE IF NOT EXISTS public.ticket

(

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

place\_id integer NOT NULL,

CONSTRAINT ticket\_pkey PRIMARY KEY (ticket\_id),

CONSTRAINT ticket\_place\_id\_fkey FOREIGN KEY (place\_id)

REFERENCES public.place (place\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

NOT VALID

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public.ticket

OWNER to postgres;

CREATE TABLE IF NOT EXISTS public.train

(

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

datetime date NOT NULL,

CONSTRAINT train\_pkey PRIMARY KEY (train\_id)

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public.train

OWNER to postgres;