

Міністерство освіти і науки України
Національний університет «Одеська політехніка»

Інститут комп'ютерних систем
Кафедра інформаційних систем

Козуб Катерина Олексіївна,
студент групи AI-212

КУРСОВА РОБОТА

Розробити базу даних маршрутів громадського транспорту з можливістю
зберігання топологічної схеми маршрутів

Спеціальність:
122 Комп'ютерні науки

Освітня програма: Комп'ютерні науки

Керівник:
Глава Марія Геннадіївна,
Канд. техн. наук, доцент

Одеса – 2022

ЗМІСТ

Завдання на курсову роботу.....	3
Анотація.....	4
Вступ.....	5
1 Аналіз предметної області та постановка задачі	7
1.1 Опис предметної області бази даних маршрутів громадського транспорту.....	7
1.2 Опис користувачів системи та їх історії (User Story).....	8
1.3 Детальний опис функцій, що автоматизуються.....	9
2 Проектування бази даних маршрутів громадського транспорту.....	10
3 Вибір програмного забезпечення	14
4 Створення бази даних.....	15
4.1 Створення таблиць.....	15
4.2 Створення представлень.....	19
4.3 Створення тригерів.....	20
5 Маніпулювання даними	24
5.1 Оператори відновлення	24
5.2 Оператор вибірки	24
6 Створення користувачів і призначення прав доступу.....	33
Висновки.....	38
Перелік використаних джерел.....	39

Національний університет «Одеська політехніка»
Інститут дистанційної та заочної освіти
Кафедра інформаційних систем

ЗАВДАННЯ
НА КУРСОВУ РОБОТУ

студенту Козуб Катерина Олексіївна група AI-212

1. Тема роботи «Розробити базу даних маршрутів громадського транспорту з можливістю зберігання топологічної схеми маршрутів»

2. Термін здачі студентом закінченої роботи 25.11.2022

3. Початкові дані до проекту (роботи) Варіант 4: Вхідні дані:

Маршрути: тип, номер, назва.

Сегменти маршрутів: маршрут, код сегменту маршруту, сегмент вулиці.

Вулиці міста: код, назва.

Сегменти вулиць: вулиця, код сегменту, опис місцеположення та граничних номерів будинків.

Вихідні дані:

Вулиці, через які проходить заданий маршрут. Маршрути, що проходять по заданій вулиці та інші.

Побудова маршруту з точки А в точку Б. Виявлення дублюючих маршрутів різними видами транспорту.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які належить розробити) постановка задачі, проектування бази даних, вибір програмного забезпечення, створення бази даних, маніпулювання даними, створення користувачів і призначення прав доступу

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) _____

Завдання видано

05.09.2022

(підпис викладача)

Завдання прийнято до виконання 05.09.2022

(підпис студента)

АНОТАЦІЯ

Розглянуто на даній курсовій роботі проектування бази даних маршрутів громадського транспорту з можливістю зберігання топологічної схеми маршрутів. Запропоновано створити вхідні та вихідні дані. Розроблено також User`s story для різних користувачів також суттєві сутності та їх характеристики. Запропоновано створити за допомогою мови SQL таблиці , згідно з варіантом , та створені представлення , тригери, відновлення, вибірку та користувачів

ABSTRACT

The design of a database of public transport routes with the possibility of storing a topological scheme of routes is considered in this course work. It is proposed to create input and output data. User`s story for different users as well as essential entities and their characteristics have also been developed. It is proposed to create tables using the SQL language, according to the option, and created views, triggers, recovery, selection and custom

ВСТУП

SQL – це стандартна мова програмування, яка використовується для створення, модифікації, пошуку і вибірки інформації, що зберігається в довільній реляційній базі даних, яка управляється відповідною системою управління базами даних (СУБД). Мова SQL дуже могутня, тому її підтримують найбільш популярні СУБД, зокрема Microsoft Access, Oracle і MySQL, хоча рівень цієї підтримки істотно залежить від того, про яку саме СУБД йдеться. Треба сказати, що мови програмування, названі також формальними мовами, відрізняються від мов спілкування, названих неформальними або природними мовами, головним чином тим, що створюються під конкретну мету, повністю позбавлені двозначності, мають вельми обмежений словниковий запас і гнучкість. Таким чином, якщо ви не отримали результату від роботи своєї програми, на який розраховували при її написанні, це відбулося тому, що ваша програма містить яку-небудь помилку (логічну або синтаксичну - в останньому випадку, швидше за все, буде виведено відповідне повідомлення, що описує помилку), а не тому, що комп'ютер неправильно зрозумів ваші інструкції, формалізовані у вигляді програми. Будучи формальною мовою, SQL, як інша мова цього типу, має свої синтаксис і семантику. Синтаксис включає власне слова і символи, а також правила, по яких ці слова і символи можна використовувати при створенні команд і програм. Семантика допомагає з'ясувати реальне значення, значення будь-якої синтаксично правильної команди. Ви цілком можете написати на SQL яку-небудь команду, відповідну синтаксису мови, яка, проте, буде виражати невірне значення (тобто буде правильною синтаксично, але невірною семантично).

В курсовій роботі я розробляла базу даних на тему маршрутів громадського транспорту з можливістю зберігання топологічної схеми маршрутів. На мою думку ця база зараз актуальна. Ця база спростить життя людям, їм не потрібно буде ритися в папірах в пошуку інформації, підраховувати суму або кількість чогось. Мені здається моя база даних яку я створила буде актуальна у мої предметній області.

Отже, програмуючи на SQL, вам треба вказати тільки те, що саме необхідно зробити, а далі сама СУБД автоматично і непомітно для вас визначає і виконує ту послідовність покрокових операцій, яку вимагається виконати для досягнення бажаного результату.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

Будь-яка організація потребує своєчасного доступу до інформації. Цінність інформації в даний час дуже висока. Роль розпорядників інформації в сучасному світі найчастіше виконують бази даних. Бази даних забезпечують надійне зберігання інформації, структурованому вигляді і своєчасний доступ до неї. Практично будь-яка сучасна організація потребує бази даних, що задовольняє ті чи інші потреби зі зберігання, управління та адміністрування даних.

В даному курсовому проекті була розроблена база даних в СУБД SQL Server 2022 для автоматизованого обліку пасажирських перевезень - це така організація, яка працює з дуже великим об'ємом інформації, як про співробітників, так і про клієнтів. Для цього потрібна спільна база даних, що включає всю необхідну інформацію. Потужність бази даних обумовлена можливістю її постійного поповнення новими даними, причому в необмеженій кількості інформації. Це є дуже зручним для користувача. Таким чином, створення бази даних, яка має такі властивості, завдання досить актуальна і корисна. Програма, що працює з БД, дозволяє вести автопарк, водії, маршрут, сегмент маршрута, вулиця та сегмент вулиці.

1.1 Опис предметної області бази даних маршрутів громадського транспорту

Ефективне функціонування сучасного підприємства неможливо без застосування інформаційних систем. Дана проблема актуальна як для великих підприємств, так і для підприємств середнього і навіть малого бізнесу. Інформаційні системи мають ряд істотних відмінностей від стандартних прикладних програм. Залежно від предметної області інформаційні системи можуть сильно відрізнятися за своєю архітектурою і функцій.

1.2. Опис користувачів системи та їх історії (User Story)

У результаті в БД «Маршрут» використовуються наступні вхідні дані:

- інформація про автовокзал;
- інформація про водіїв;
- інформація про маршрути;
- інформація про сегмент маршрутів;
- інформація про вулиці;
- інформація про сегмент вулиці;

Users' stories:

Автопарк

1. Я автопарк, я можу подивитися всю інформацію про водіїв.
2. Я автопарк, я можу змінювати інформацію про рух маршрутки.
3. Я автопарк, я можу змінювати інформацію про вартість проїзду.
4. Я автопарк, я можу змінювати водіїв місцями.

Водій

1. Я водій, я можу подивитися про зміни в маршруті.
2. Я водій, я можу подивитися стан маршрутки.
3. Я водій, я можу змінювати щось в салоні маршрутки.
4. Я водій, я можу дивитися за поведінкою людей в маршрутке.

Сегмент маршрутів

1. Я сегмент маршрутів, я можу змінювати час зупинки маршрутів.
2. Я сегмент маршрутів, я можу подивитися інформацію про водія.
3. Я сегмент маршрутів, я можу подивитися інформацію к якому маршруту я відповідаю.
4. Я сегмент маршрутів, я можу подивитися якій я вулиці відповідаю.

Вулиця

1. Я вулиця, я можу змінювати час прибуття маршрути.
2. Я вулиця, я можу змінювати напрямок маршрутів за якоїсь причини.
3. Я вулиця, я можу дивитися якому сегменту маршрутів я відповідаю.
4. Я вулиця, я можу подивитися який номер маршрутки відповідає вулиці

Сегмент вулиці

1. Я сегмент вулиці, я можу змінювати час зупинки маршрутів.
2. Я сегмент вулиці, я можу подивитися якій я вулиці відповідаю.
3. Я сегмент вулиці, я можу подивитися інформацію про водія.
4. Я сегмент вулиці, я можу змінювати інформацію збір людей на місці зупинки транспорту.

1.3. Детальний опис функцій, що автоматизуються

Мета проектування - забезпечення найбільш природних для людини способів збору і представлення тієї інформації, яку передбачається зберігати в створеній БД. Тому інфологічну модель намагаються будувати за аналогією з природною мовою. Основними конструктивними елементами інфологічних моделей є сутності, зв'язки між ними та їх властивості.

2 ПРОЕКТУВАННЯ БАЗИ ДАНИХ МАРШРУТІВ ГРОМАДСЬКОГО ТРАНСПОРТУ

Основними поняттями реляційних баз даних є: тип даних, домен, атрибут, кортеж, первинний ключ і відношення.

Поняття тип даних в реляційній моделі даних повністю адекватно поняттю типу даних в мовах програмування. Зазвичай всі сучасні реляційні

БД підтримують наступні типи даних:

- числові;
- символічні;
- великі двійкові об'єкти (малюнки та медіа-файли);
- бітові рядки;
- спеціалізовані числові дані (такі як «гроші»);
- спеціальні «темпоральні дані» (дата, час та часовий інтервал).

Домен - це семантичне поняття. Домен можна розглядати як підмножину значень деякого типу даних, які мають певний сенс. Домен характеризується наступними властивостями:

- домен має унікальну назву (в межах бази даних);
- домен визначений на деякому простому типі даних або на іншому домені;
- домен може мати деяку логічну умову, що дозволяє описати підмножину даних, допустимих для даного домену;
- домен несе певне смислове навантаження

Кортеж, що відповідає даній схемі відношення, - це множина пар {назва атрибута, значення}, яке містить одне входження кожної назви атрибута, що належить схемі відношення. «Значення» є припустимим значенням домену цього атрибута (або типу даних, якщо поняття домену не підтримується). Попросту кажучи, кортеж - це набір іменованих значень заданого типу.

Відношення - це множина кортежів, які відповідають одній схемі відношення. Насправді, поняття схеми відношення найближче до поняття структурного типу даних в мовах програмування.

Число атрибутів у відношенні називають степенем (або -арністю) відношення.

Первинний ключ – це поле або набір полів зі значеннями, унікальними в межах певної таблиці. За допомогою значень ключа можна посилатися на окремі записи, адже значення ключа кожного запису відрізняється. Кожна таблиця може мати лише один первинний ключ

Виділені суттєві сутності, що характеризують предметну область, та їх характеристика представлені в таблиці 2.1.

Таблиця 2.1 – Суттєві сутності, що характеризують предметну область

Сутність	Характеристика
Автопарк	Містить інформацію про водителя
Водитель	Містить інформацію про назву маршрута та номер маршрута
Маршрути	Містить інформацію про тип, назву, номер та водителя
Сегменти маршрутів	Містить інформацію про маршрут, код вулиці та код сегменту
Вулиці міста	Містить інформацію про код вулиці, сегмент маршруту, назва, код сегменту
Сегменти вулиць	Містить інформацію про сегмент вулиці та код сегменту

Виділені властивості об'єктів (атрибути), їх типи даних та ключі представлено в таблиці 2.2.

Таблиця 2.2 – Виділені атрибути сутностей та їх характеристики

1. Таблиця Автопарк

Сутність	Властивість	Тип даних	Ключ
Автопарк	id	Цілочислений	Первинний
	Назва	Символний	
	Керівник	Символний	

2. Таблиця Водії

Сутність	Властивість	Тип даних	Ключ
Водії	id водителя	Цілочислений	Первинний
	id Автопарка	Цілочислений	Зовнішній
	Призвіще	Символний	
	Ім'я	Символний	
	По Батькові	Символний	
	Адреса	Символний	

3. Таблиця Маршрути

Сутність	Властивість	Тип даних	Ключ
Маршрути	id Маршута	Цілочислений	Первинний
	id водителя	Цілочислений	Зовнішній
	Тип	Символний	
	Номер	Цілочислений	
	Назва	Символний	
	Код сегментів	Цілочислений	

4. Таблиця Сегмент маршрутів

Сутність	Властивість	Тип даних	Ключ
Сегменти маршрутов	id сегмента маршрута	Цілочислений	Первинний
	id Маршута	Цілочислений	Зовнішній
	id Вулиці	Цілочислений	Зовнішній
	Код сегменту	Цілочислений	

5. Таблиця Вулиць

Сутність	Властивість	Тип даних	Ключ
Вулиці	id Вулиці	Цілочислений	Первинний
	id сегмента вулиці	Цілочислений	Зовнішній
	Назва	Символьний	
	Код сегмента	Цілочислений	

6. Таблиця Сегмент Вулиці

Сутність	Властивість	Тип даних	Ключ
Сегменти вулиць	id сегмент вулицы	Цілочислений	Первинний
	Код вулицы	Цілочислений	

Визначені типи зв'язків між сутностями представлено в таблиці 2.3.

Таблиця 2.3 – Типи зв'язків між сутностями

Сутність	Тип зв'язку	Сутність
Автопарк	1 : 1	Водії
Водії	1 : 1	Маршрути
Маршрути	1 : 1	Сегмент маршрута
Вулиця	1 : 1	Сегмент маршрута
Сегмент вулиці	1 : 1	Вулиця

На основі виділених сутностей, їх атрибутів та зв'язків між сутностями створено інформаційну модель в нотації draw.io , яка представлена на рис. 2.1.

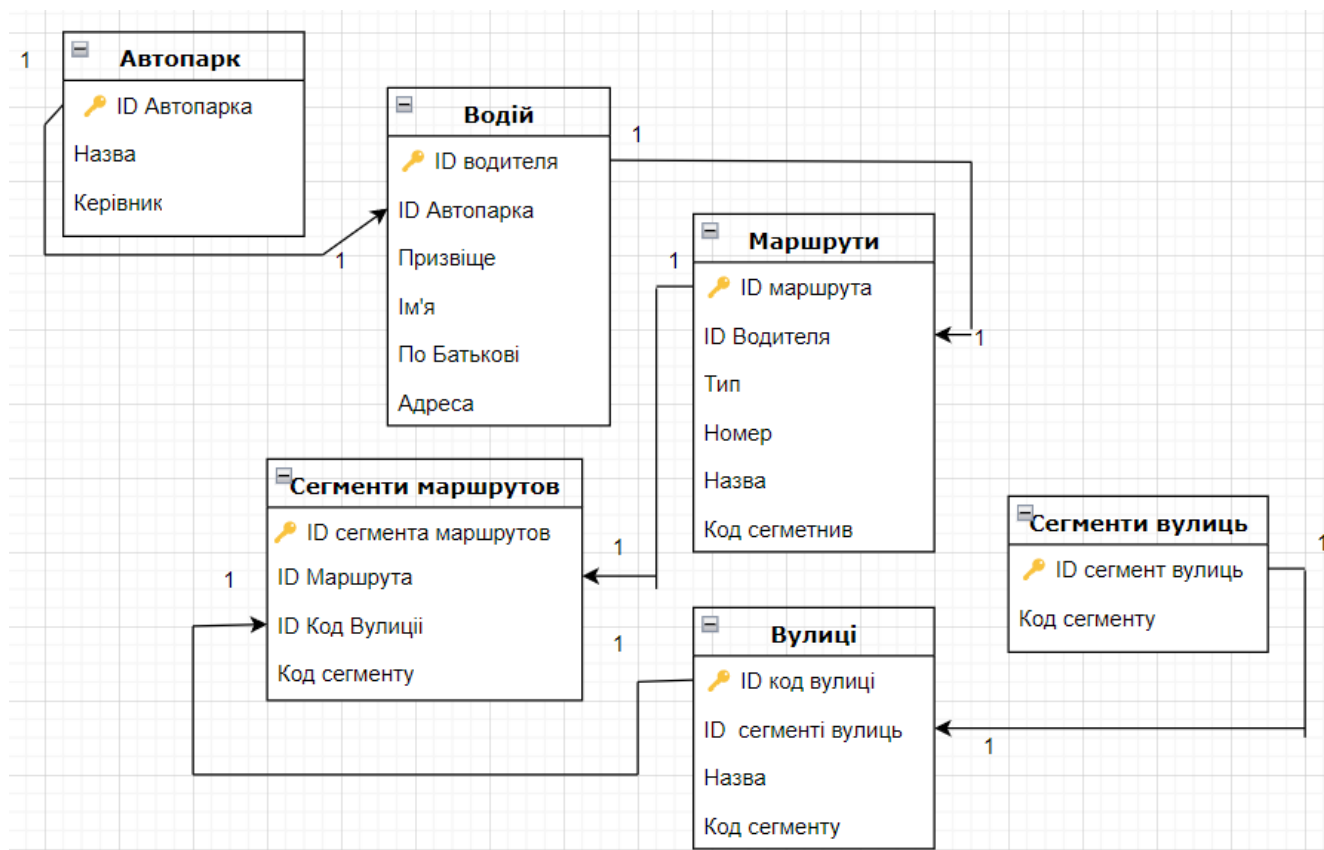


Рисунок 2.1 – Інформаційна модель в нотації draw.io

3 ВИБІР ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

База даних створювалась у pgAdmin4.

Характеристики комп'ютера: процесор - AMD Ryzen 5 4600H with Radeon Graphics 3.00 GHz; оперативна пам'ять — 8 Гб; тип системи — 64-розрядна операційна система; відеокарта – NVIDIA GTX GeForce 1650

4 СТВОРЕННЯ БАЗИ ДАНИХ

Процес створення бази даних в системі SQL -сервера складається з двох етапів: спочатку організовується сама база даних, а потім журнал транзакцій, що належить їй. Інформація розміщується у відповідних файлах, що мають розширення *.mdf (для бази даних) і *.ldf. (для журналу транзакцій). У файлі бази даних записуються відомості про основні об'єкти (таблицях, індексах, представленнях і так далі), а у файлі журналу транзакцій - про процес роботи з транзакціями (контроль цілісності даних, стани бази даних до і після виконання транзакцій).

Створення бази даних в системі SQL -сервер здійснюється командою CREATE DATABASE. Слід зазначити, що процедура створення бази даних в SQL -сервері вимагає наявності прав адміністратора сервера.

4.1 Створення таблиць

ІНСТРУКЦІЯ CREATE TABLE

Інструкція CREATE TABLE визначає нову таблицю та готує її до запису даних. Різні блоки інструкції задають елементи визначення таблиці.

Синтаксична структура інструкції є такою:

<p>CREATE TABLE <i>ім'я таблиці (визначення стовпця або визначення обмежень таблиці,)</i></p>

Після виконання інструкції створюється нова таблиця. Створена таблиця є порожньою; додавати до неї записи можна за допомогою інструкції INSERT.

Визначення первинного та зовнішнього ключів

В інструкції CREATE TABLE вказується також інформація про первинний ключ та її зв'язках з іншими таблицями бази даних. Ця інформація міститься в частині PRIMARY KEY та FOREIGN KEY.

В частині PRIMARY KEY задається стовпець чи стовпці, які утворюють первинний ключ таблиці. Цей стовець чи стовпці є унікальними ідентифікаторами рядків таблиці. СУБД автоматично слідкує за тим, щоб первинний ключ кожного рядка таблиці містив унікальне значення. Крім того, у визначенні рядків первинного ключа має бути вказано, що вони не можуть містити значення NULL.

В частині FOREIGN KEY задається зовнішній ключ таблиці і визначається зв'язок, який задається. В ньому вказуються:

- стовець чи стовпці створюваної таблиці, які утворюють зовнішній ключ;
- таблиця, зв'язок з якою створюється

В цій БД я створюватиму 6 таблиць: автопарк, водії, маршрути, сегмент маршрутів, вулиця, сегмент вулиці. Створюватиму таблиці в PgAdmin 4.

```
CREATE TABLE carpark
(
  carpark_id integer PRIMARY KEY,
  name_carpark text NOT NULL,
  kerivnik text NOT NULL,
  phone int UNIQUE,
  adress varchar(30) NOT NULL
);
CREATE TABLE driver
(
  driver_id integer PRIMARY KEY,
  fk_carpark_id integer REFERENCES carpark(carpark_id),
  last_name varchar(30) NOT NULL,
  first_name varchar(30) NOT NULL,
  surname varchar(30) NOT NULL,
  phone int UNIQUE,
  adress varchar(30) NOT NULL
);
CREATE TABLE route
(
  route_id integer PRIMARY KEY,
  fk_driver_id integer REFERENCES driver(driver_id),
  typeof_route varchar(30) NOT NULL,
  number_route int NOT NULL,
  name_route varchar(30) NOT NULL,
  segment_code int NOT NULL,
  from_where_to_where text NOT NULL
```



```

);
CREATE TABLE street_segment
(
  street_segment_id integer PRIMARY KEY,
  segment_code int NOT NULL
);
CREATE TABLE street
(
  street_code_id integer PRIMARY KEY,
  fk_street_segment_id integer REFERENCES street_segment(street_segment_id),
  name_street varchar(30) NOT NULL,
  segment_code int NOT NULL
);
CREATE TABLE route_segment
(
  route_segment_id integer PRIMARY KEY,
  fk_route_id integer REFERENCES route(route_id),
  fk_street_code_id integer REFERENCES street(street_code_id),
  segment_code int NOT NULL
);
SELECT * FROM carpark

```

	carpark_id [PK] integer	name_carpark text	kerivnik text
1	4567	Автопарк 1	Іванов Ів...
2	2346	Автопарк 2	Величко ...
3	3421	Автопарк 3	Вербенко...
4	5698	Автопарк 4	Спіян Іго...
5	5909	Автопарк 5	Аверен В...

Рис. 4.1 — Дані в таблиці “Автопарк”

```
SELECT * FROM driver
```

	driver_id [PK] integer	carpark_id integer	fk_carpark_id integer	last_name text	first_name text	surname text	address text
1	34	4567	4567	Куличков	Іван	Вікторов...	Пушкінск...
2	61	2346	2346	Вілков	Артем	Ігорович	Канатка ...
3	57	3421	3421	Супрун	Миколай	Володимі...	Камінтер...
4	78	5698	5698	Баяркін	Валерії	Іванович	Спортівн...
5	90	5909	5909	Фіданюк	Віктор	Вікторов...	Костіна 1...

Рис. 4.2 — Дані в таблиці “Водії”

```
SELECT * FROM route
```

	route_id [PK] integer	driver_id integer	fk_driver_id integer	typeof_route text	number_route integer	name_route text	segment_code integer
1	31	34	34	кільцеві	175	Маршрутка...	345678
2	98	61	61	кільцеві	210	Маршрутка...	762209
3	65	57	57	кільцеві	140	Маршрутка...	650912
4	87	78	78	кільцеві	75	Маршрутка...	778091
5	21	90	90	кільцеві	125	Маршрутка...	985421

Рис. 4.3 — Дані в таблиці “Маршрути”

SELECT * FROM route_segment

	route_segment_id [PK] integer	route_id integer	fk_route_id integer	street_code_id integer	fk_street_code_id integer	segment_code integer
1	23458	31	31	8711	8711	345678
2	89122	98	98	2213	2213	762209
3	34512	65	65	8721	8721	650912
4	45611	87	87	76032	76032	778091
5	87651	21	21	99923	99923	985421

Рис. 4.4 — Дані в таблиці “Сегмент маршрутів”

SELECT * FROM street

	street_code_id [PK] integer	fk_street_segment_id integer	name_street text	segment_code integer	street_segment_id integer
1	8711	12	Генерала П...	700895	12
2	2213	23	Канатная	89331	23
3	8721	67	О.Вишні	981134	67
4	76032	76	І.Франка	561221	76
5	99923	77	Горького	91113	77

Рис. 4.5 — Дані в таблиці “Вулиця”

SELECT * FROM street_segment

	street_segment_id [PK] integer	segment_code integer
1	12	700895
2	23	89331
3	67	981134
4	76	561221
5	77	91113

Рис. 4.6 — Дані в таблиці “Сегмент вулиць”

4.2 Створення представлень

Синтаксис створення представлень:

CREATE VIEW назва_представлення AS [запит]

На даному разі я користуюся оператором CREATE VIEW за для створення представлення driver_carpark. При створенні я брала дані з таблиці Водії та Автопарк

SQL Код:

```
CREATE VIEW driver_carpark AS
SELECT last_name, kerivnik
FROM driver,carpar
```

Результат виконання команди:

	last_name text	kerivnik text
1	Куличков	Іванов Ів...
2	Вілков	Іванов Ів...
3	Супрун	Іванов Ів...
4	Баяркін	Іванов Ів...
5	Фіданюк	Іванов Ів...

Рис. 4.7 — Дані у “Водії_Автопарк”

На даному разі я користуюся оператором CREATE VIEW за для створення представлення driver_and_routes. При створенні я брала дані з таблиці Водії та Маршрути.

SQL Код:

```
CREATE VIEW driver_and_routes AS
SELECT first_name, name_route
FROM driver, route
```

Результат виконання команди:

	first_name text	name_route text
1	Іван	Маршрутка...
2	Артем	Маршрутка...
3	Миколай	Маршрутка...
4	Валерії	Маршрутка...
5	Віктор	Маршрутка...

Рис. 4.8 — Дані у “Водії_и_Маршрути”

Видаляю представлення за допомогою оператора DROP VIEW

SQL Код:

DROP VIEW driver_and_routes,driver_carpark

Результат виконання команди:

DROP VIEW

Рис. 4.8 — Дані видалені

4.3 Створення тригерів

Синтаксис створення тригерів та тригерних функцій:

```
CREATE OR REPLACE FUNCTION назва_тригерної_функції() RETURNS
trigger
AS $$ BEGIN
програма;
RETURN значення; END;$$
LANGUAGE 'plpgsql';
```

Замінюємо в таблиці Автопарк первинний ключ на 0 , за умовою , що значення нових даних <= 2346 (використовуємо тригер на операцію INSERT) :

SQL Код:

```
CREATE OR REPLACE FUNCTION check_carparkA()
RETURNS TRIGGER AS
$$
BEGIN
IF new.carpark_id <= 2346 THEN
UPDATE carpark SET carpark_id = 0 WHERE carpark_id=new.carpark_id;
END IF;
RETURN new;
END;
$$
LANGUAGE plpgsql;
CREATE TRIGGER check_carparkA
AFTER INSERT ON carpark FOR EACH ROW EXECUTE PROCEDURE
check_carparkA();
INSERT INTO carpark VALUES
(0,'Автопарк 2','Величко Володимір')
```

	carpark_id [PK] integer	name_carpark text	kerivnik text
1	4567	Автопарк 1	Іванов Ів...
2	2346	Автопарк 2	Величко ...
3	3421	Автопарк 3	Вербенко...
4	5698	Автопарк 4	Спіян Іго...
5	5909	Автопарк 5	Аверен В...
6	0	Автопарк 2	Величко ...

Рис. 4.10 — Результат тригера

За допомогою модифікації даних DELETE, зменшуємо послідовність на один із заданої таблиці

SQL Код:

```
CREATE OR REPLACE FUNCTION carparkA_deleted() RETURNS trigger
AS $$
DECLARE
card_id INTEGER;
BEGIN
SELECT carpark_id INTO old_id FROM old WHERE carpark_id =
old.carpark_id;
ALTER SEQUENCE s_carpark RESTART WITH old_id;
RETURN old;
```

```

END;
$$ LANGUAGE 'plpgsql';
CREATE OR REPLACE TRIGGER carparkA_deleted
AFTER DELETE ON carpark
FOR EACH STATEMENT
EXECUTE PROCEDURE carparkA_deleted()

```

	carpark_id [PK] integer	name_carpark text	kerivnik text
1	4567	Автопарк 1	Іванов Іван
2	2346	Автопарк 2	Величко Володим...
3	3421	Автопарк 3	Вербенко Віктор
4	5698	Автопарк 4	Спіян Ігорь
5	5909	Автопарк 5	Аверен Володимір

Рис. 4.11 — Результат тригера

За допомогою SELECT створюємо функцію , яка підраховує кількість сегментів вулиць

SQL Код:

```

REATE FUNCTION street_sum() RETURNS bigint as
$$
BEGIN
RETURN SUM(segment_code)
FROM street_segment;
END;
$$
LANGUAGE 'plpgsql';
SELECT street_sum

```

	street_sum bigint
1	2423694

Рис. 4.12 — Результат тригера

В цьому прикладі я намагаюсь замінити в таблиці сегмент маршрутів значення сегмент кода на 0, якщо сегмент кода більше або дорівнює 89122.

SQL Код:

```

CREATE FUNCTION checkk_routeser_id() RETURNS trigger

```

```

AS $$ BEGIN
IF new.route_segment_id >= 89122 THEN
UPDATE route_segment SET segment_code= '0' WHERE route_segment_id
=new.route_segment_id ;
END IF;
RETURN new;
END;
$$ LANGUAGE 'plpgsql';
CREATE TRIGGER checkk_routeser_id
AFTER UPDATE OF route_segment_id ON route_segment
FOR EACH ROW
EXECUTE PROCEDURE checkk_routeser_id();
Update route_segment SET route_segment_id = 89122
WHERE route_segment_id = 89122;

```

	route_segment_id [PK] integer	route_id integer	fk_route_id integer	street_code_id integer	fk_street_code_id integer	segment_code integer
1	23458	31	31	8711	8711	345678
2	34512	65	65	8721	8721	650912
3	45611	87	87	76032	76032	778091
4	87651	21	21	99923	99923	985421
5	89122	98	98	2213	2213	0

Рис. 4.13 — Результат тригера

5 МАНІПУЛЮВАННЯ ДАНИМИ

5.1 Оператори відновлення

ОБНОВЛЕННЯ ІСНУЮЧИХ ДАНИХ (ІНСТРУКЦІЯ UPDATE)

Інструкція UPDATE обновляє значення одного чи декількох стовпців у вибраних рядках однієї таблиці. Синтаксична структура інструкції є такою:

UPDATE ім'я таблиці SET ім'я стовпця=вираз,....

WHERE умова відбору

В інструкції вказується цільова таблиця, яка має бути модифікована. В блоці SET вказується, які стовпці мають бути модифікованими. Блок WHERE відбирає рядки таблиці для обновлення. Якщо цей блок відсутній, то обновляться всі рядки таблиці. Також, допустимим є використання вкладених запитів для задання умов відбору рядків.

Змінюємо driver_id в таблиці маршрути

UPDATE route SET driver_id = '98'

WHERE driver_id = (SELECT MIN(driver_id)FROM route)

	route_id [PK] integer	driver_id integer	fk_driver_id integer	typeof_route text	number_route integer	name_route text	segment_code integer
1	65	57	57	кільцеві	140	Маршрутка...	650912
2	87	78	78	кільцеві	75	Маршрутка...	778091
3	21	90	90	кільцеві	125	Маршрутка...	985421
4	31	98	34	кільцеві	175	Маршрутка...	345678
5	98	61	61	кільцеві	210	Маршрутка...	762209

Рис. 5.1.1 — Результат операції

5.2 Оператор вибірки

Існує багато умов відбору, які дозволяють ефективно створювати різні типи запитів. Основними умовами відбору є:

1. Порівняння. Значення одного виразу порівнюється із значенням іншого виразу для кожного рядка даних. Існує шість різних способів порівняння виразів:

=, <>, <, <=, >, >=

Результатом виконання СУБД порівняння двох виразів може бути:

- якщо порівняння істинне, то результат перевірки має значення TRUE;
- якщо порівняння хибне, то результат перевірки має значення FALSE;
- якщо хоча б один з двох виразів має значення NULL, то результатом перевірки буде NULL.

Для того, щоб дізнатися, скільки сегментів кода більше 0 в таблиці сегмент маршрутів, потрібно написати наступний фрагмент коду:

```
SELECT * FROM route_segment
WHERE segment_code > 0
```

	route_segment_id [PK] integer	route_id integer	fk_route_id integer	street_code_id integer	fk_street_code_id integer	segment_code integer
1	23458	31	31	8711	8711	345678
2	34512	65	65	8721	8721	650912
3	45611	87	87	76032	76032	778091
4	87651	21	21	99923	99923	985421

Рис. 5.2.1 — Результат операції

Для того, щоб дізнатися, скільки сегментів кода більше і дорівнює 561221 в таблиці сегмент вулиць, потрібно написати наступний фрагмент коду:

```
SELECT * FROM street_segment
WHERE segment_code >= 561221
```

	street_segment_id [PK] integer	segment_code integer
1	12	700895
2	67	981134
3	76	561221

Рис. 5.2.2 — Результат операції

2. Перевірка на належність діапазону значень. Перевіряється чи потрапляє вказане значення в визначений діапазон. Схематично таку форму умови відбору можна зобразити так:

вираз, що перевіряється BETWEEN нижня межа AND верхня межа

або

вираз, що перевіряється NOT BETWEEN нижня межа AND верхня межа
При такій перевірці верхня та нижня межі вважаються частиною діапазону.

В деяких СУБД визначені такі правила обробки значення NULL в перевірці BETWEEN:

- якщо вираз, що перевіряється має значення NULL або якщо обидва виразів, які визначають діапазон, рівні NULL, то і перевірка BETWEEN повертає NULL;

- якщо вираз, що визначає нижню межу діапазону, має значення NULL, то перевірка BETWEEN повертає FALSE, коли значення, що перевіряється більше, ніж верхня межа діапазону, і NULL в протилежному випадку;

- якщо вираз, що визначає верхню межу діапазону, має значення NULL, то перевірка BETWEEN повертає FALSE, коли значення, що перевіряється менше, ніж нижня межа діапазону, і NULL в протилежному випадку.

Для виводу проміжку з 4567 по 5909 з таблиці автопарк, потрібно написати наступний фрагмент коду :

```
SELECT * FROM carpark
WHERE carpark_id BETWEEN 4567 AND 5909
```

	carpark_id [PK] integer	name_carpark text	kerivnik text
1	4567	Автопарк 1	Іванов Іван
2	5698	Автопарк 4	Спiян Ігорь
3	5909	Автопарк 5	Аверен Володимiр

Рис. 5.2.3 — Результат операції

Для виводу проміжку з 2213 по 8721 з таблиці вулиці, потрібно написати наступний фрагмент коду :

```
SELECT * FROM street
WHERE street_code_id BETWEEN 2213 AND 8721
```

	street_code_id [PK] integer	fk_street_segment_id integer	name_street text	segment_code integer	street_segment_id integer
1	8711	12	Генерала Петро...	700895	12
2	2213	23	Канатная	89331	23
3	8721	67	О.Вишні	981134	67

Рис. 5.2.4 — Результат операції

3. Перевірка на входження до множини. Перевіряється, чи співпадає значення виразу з одним із значень заданої множини. Схематично таку форму умови відбору можна зобразити так:

вираз, що перевіряється IN (список констант через кому)
або

вираз, що перевіряється NOT IN (список констант через кому)

Для виводу інформації ‘Автопарк 1’ і ‘Автопарк 5’ з таблиці автопарк, потрібно написати наступний фрагмент коду :

```
SELECT * FROM carpark
WHERE name_carpark IN('Автопарк 1','Автопарк 5')
```

	carpark_id [PK] integer	name_carpark text	kerivnik text
1	4567	Автопарк 1	Іванов Іван
2	5909	Автопарк 5	Аверен Володимир

Рис. 5.2.5 — Результат операції

Для виводу інформації про всі номери маршруток окрім 210 і 75 з таблиці маршрути, потрібно написати наступний фрагмент коду :

```
SELECT * FROM route
WHERE number_route NOT IN('210','75')
```

	route_id [PK] integer	driver_id integer	fk_driver_id integer	typeof_route text	number_route integer	name_route text	segment_code integer
1	31	34	34	кільцеві	175	Маршрутка...	345678
2	65	57	57	кільцеві	140	Маршрутка...	650912
3	21	90	90	кільцеві	125	Маршрутка...	985421

Рис. 5.2.6 — Результат операції

4. Перевірка на відповідність шаблону. Перевіряється чи відповідає рядкове значення, яке міститься в стовпці певному шаблону. Схематично таку форму умови відбору можна зобразити так:

ім'я стовпця LIKE шаблон
або
ім'я стовпця NOT LIKE шаблон

Шаблон являє собою рядок, в який може входити один або більше підстановочних знаків. В SQL використовуються такі підстановочні знаки:

- 1) % - співпадає з будь-якою послідовністю з нуля чи більше символів;
- 2) _ (символ підкреслення) - співпадає з будь-яким окремим символом.

Для виводу інформацій про I_____ I___ з таблиці автопарк, потрібно написати наступний фрагмент коду :

```
SELECT * FROM carpark
WHERE kerivnik LIKE 'I_____ I___'
```

	carpark_id [PK] integer	name_carpark text	kerivnik text
1	4567	Автопарк 1	Іванов Ів...

Рис. 5.2.7 — Результат операції

Для виводу інформацій про всі названня маршруток окрім Маршрутка з таблиці маршруток, потрібно написати наступний фрагмент коду :

```
SELECT * FROM route
WHERE name_route NOT LIKE 'Маршрутка 3'
```

	route_id [PK] integer	driver_id integer	fk_driver_id integer	typeof_route text	number_route integer	name_route text	segment_code integer
1	31	34	34	кільцеві	175	Маршрутка...	345678
2	87	78	78	кільцеві	75	Маршрутка...	778091
3	21	90	90	кільцеві	125	Маршрутка...	985421
4	98	61	61	кільцеві	210	Маршрутка...	762209

Рис. 5.2.8 — Результат операції

5. Перевірка на рівність значенню NULL. Значення NULL дозволяє застосовувати тризначну логіку в умовах відбору. У випадку, коли необхідно явно перевірити значення стовпців на рівність NULL використовується така структура умови:

ім'я стовпця IS NULL
або
ім'я стовпця IS NOT NULL

Дана перевірка завжди повертає значення TRUE або FALSE.

Перераховані прості умови відбору, після застосування до деякого рядка повертають значення TRUE, FALSE або NULL. За допомогою правил логіки ці прості умови можна об'єднувати в більш складні, використовуючи при цьому логічні операції AND, OR, NOT. Їх таблиці істинності наведені нижче:

Таблиця 1

Таблиця істинності оператора AND

AND	TRUE	FALSE	NULL
TRUE	TRUE	FALSE	NULL
FALSE	FALSE	FALSE	FALSE
NULL	NULL	FALSE	NULL

Таблиця 2

Таблиця істинності оператора OR

OR	TRUE	FALSE	NULL
TRUE	TRUE	TRUE	TRUE
FALSE	TRUE	FALSE	NULL
NULL	TRUE	NULL	NULL

Таблиця 3

Таблиця істинності оператора NOT

NOT	TRUE	FALSE	NULL
	FALSE	TRUE	NULL

Оператор NOT володіє найвищим пріоритетом, наступний пріоритет має оператор AND, найнижчий – OR

Для виводу інформації де атрибут має значення не нуль з таблиці маршрути, потрібно написати наступний фрагмент коду :

```
SELECT * FROM route
WHERE segment_code IS NOT NULL
```

	route_id [PK] integer	driver_id integer	fk_driver_id integer	typeof_route text	number_route integer	name_route text	segment_code integer
1	31	34	34	кільцеві	175	Маршрутка...	345678
2	65	57	57	кільцеві	140	Маршрутка...	650912
3	87	78	78	кільцеві	75	Маршрутка...	778091
4	21	90	90	кільцеві	125	Маршрутка...	985421
5	98	61	61	кільцеві	210	Маршрутка...	762209

Рис. 5.2.9 — Результат операції

Для виводу інформації де атрибут має значення нуль з таблиці маршрути, потрібно написати наступний фрагмент коду :

```
SELECT * FROM route
WHERE segment_code IS NULL
```

	route_id [PK] integer	driver_id integer	fk_driver_id integer	typeof_route text	number_route integer	name_route text	segment_code integer
--	--------------------------	----------------------	-------------------------	----------------------	-------------------------	--------------------	-------------------------

Рис. 5.2.10 — Результат операції

СТАТИСТИЧНІ ФУНКЦІЇ

Для проведення підсумків по інформації, яка міститься в базі даних, застосовуються статистичні (агрегатні) функції. Статистична функція приймає в якості аргументу будь-який стовпець даних, а повертає одне значення. Існують такі статистичні функції:

Таблиця 4

Статистичні функції

Функція	Значення
SUM()	обчислює суму всіх значень стовпця
AVG()	обчислює середнє всіх значень стовпця
MIN()	знаходить найменше серед всіх значень стовпця
MAX()	знаходить найбільше серед всіх значень стовпця
COUNT()	підраховує кількість значень, що містить стовпець

Аргументом статистичної функції може бути ім'я стовпця або арифметичний вираз.

Структура статистичних функцій наведена нижче:

- SUM (вираз) або SUM (DISTINCT ім'я стовпця)
- AVG (вираз) або AVG (DISTINCT ім'я стовпця)
- MIN (вираз)
- MAX (вираз)
- COUNT (вираз) або COUNT (DISTINCT ім'я стовпця)

Для виводу інформації про середнє значення з таблиці вулиця, потрібно написати наступний фрагмент коду :

```
SELECT AVG (segment_code)
FROM street
```


	avg numeric 
1	484738.8000

Рис. 5.2.11 — Результат операції

Для виводу інформації про суму з таблиці сегмент вулиць, потрібно написати наступний фрагмент коду:

```
SELECT SUM(segment_code)
FROM street_segment
```


	sum bigint 
1	2423694

Рис. 5.2.12 — Результат операції

Для виводу інформації про мінімальне значення з таблиці сегментмаршрутов, потрібно написати наступний фрагмент коду:

```
SELECT MIN(segment_code)
FROM route_segment
```

	min integer 
1	0

Рис. 5.2.13 — Результат операції

Для виводу інформації про максимального значення з таблиці сегмент маршрутів, потрібно написати наступний фрагмент коду:

```
SELECT MAX(segment_code)
FROM route_segment
```


	max integer 
1	985421

Рис. 5.2.14 — Результат операції

Для виводу інформації про підраховуємо кількість route_id (оператор DISTINCT обмежує повторення значень)

```
SELECT COUNT( DISTINCT route_id )
FROM route
```


	count bigint 
1	5

Рис. 5.2.15 — Результат операції

6 СТВОРЕННЯ КОРИСТУВАЧІВ І ПРИЗНАЧЕННЯ ПРАВ ДОСТУПУ

1. Створіть трьох користувачів.

В цьому прикладі я створі трьох користувачі (worker1, worker2, worker3)

SQL Код:

```
CREATE USER worker1;
CREATE USER worker2;
CREATE USER worker3;
```

CREATE ROLE

Рис. 6.1 — Результат створення

2. Надайте право першому користувачу на зміну декількох стовпців будь-якої таблиці.

В цьому прикладі я надаю право першому користувачу (worker1) на зміну декількох стовпців (ім'я та адреси) у таблиці водій.

SQL Код:

```
GRANT UPDATE (first_name, address) ON driver TO worker1
```

GRANT

Рис. 6.2 — Результат надання права

Перевірка:

```
SET ROLE 'worker1';
SELECT SESSION_USER, CURRENT_USER;
```

	session_user name	current_user name
1	postgres	worker1

Рис. 6.3 — Результат перевірки

Update driver SET first_name = 'Іван' WHERE address = 'Пушкінская 3';

UPDATE 1

Рис. 6.4 — Результат зміни

	driver_id [PK] integer	carpark_id integer	fk_carpark_id integer	last_name text	first_name text	surname text	address text
1	34	4567	4567	Куличков	Іван	Вікторов...	Пушкінск...
2	61	2346	2346	Вілков	Артем	Ігорович	Канатка ...
3	57	3421	3421	Супрун	Миколай	Володимі...	Камінтер...
4	78	5698	5698	Баяркін	Валерії	Іванович	Спортівн...
5	90	5909	5909	Фіданюк	Віктор	Вікторов...	Костіна 1...

Рис. 6.5 — Результат таблиці

3.Надайте право всім користувачам системи на перегляд будь-якої таблиці

В цьому прикладі я надаю усім користувачам системи на перегляд усіх таблиць
 GRANT SELECT ON carpark,driver,route,route_segment,street,street_segment TO
 PUBLIC

GRANT

Рис. 6.6 — Результат

Перевірка:

SELECT * FROM route_segment

	route_segment_id [PK] integer	route_id integer	fk_route_id integer	street_code_id integer	fk_street_code_id integer	segment_code integer
1	23458	31	31	8711	8711	345678
2	34512	65	65	8721	8721	650912
3	45611	87	87	76032	76032	778091
4	87651	21	21	99923	99923	985421
5	89122	98	98	2213	2213	0

Рис. 6.5 — Результат таблиці

4.Надайте право другому користувачу вставляти або модифікувати значення таблиці з правом передавати іншим користувачам вказані права.

В цьому прикладі я надаю право другому користувачу (worker2) модифікувати значення в таблиці учень з правом передавати worker3 права на оновлення(модифікування) стовпців ім'я та адресу.

```
GRANT UPDATE (first_name, address) ON driver TO worker2 WITH
GRANT OPTION
```

GRANT

Рис. 6. 7 Результат надання права

Перевірка:

```
SET ROLE 'worker2';
```

```
SELECT SESSION_USER, CURRENT_USER;
```

	session_user name	current_user name
1	postgres	worker2

Рис. 6.8 — Результат перевірки

```
GRANT UPDATE (first_name, address) ON driver TO worker3
```

GRANT

Рис. 6. 9 — Результат надання права

```
Update driver SET address = 'Пушкінская 24' WHERE first_name = 'Іван'
```

	driver_id [PK] integer	carpark_id integer	fk_carpark_id integer	last_name text	first_name text	surname text	address text
1	61	2346	2346	Вілков	Артем	Ігорович	Канатка ...
2	57	3421	3421	Супрун	Миколай	Володимі...	Камінтер...
3	78	5698	5698	Баяркін	Валерії	Іванович	Спортівн...
4	90	5909	5909	Фіданюк	Віктор	Вікторов...	Костіна 1...
5	34	4567	4567	Куличков	Іван	Вікторов...	Пушкінск...

Рис. 6. 10 — Результат

5.Надайте третьому користувачу права адміністратора (всі права на всі таблиці).

В цьому прикладі я надаю третьому користувачу (worker3) права адміністратора (всі права на всі таблиці).

```
GRANT ALL PRIVILEGES ON carpark,driver,route,route_segment,
street,street_segment TO worker3
```

GRANT

Рис. 6. 11 — Результат надання права

Перевірка:

```
SET ROLE 'worker3';
SELECT SESSION_USER, CURRENT_USER;
```

	session_user name	current_user name
1	postgres	worker3

Рис. 6.12 — Результат перевірки

Update driver SET address = 'Канатна 234' WHERE first_name = 'Артем'

	driver_id [PK] integer	carpark_id integer	fk_carpark_id integer	last_name text	first_name text	surname text	address text
1	57	3421	3421	Супрун	Миколай	Володимі...	Камінтер...
2	78	5698	5698	Баяркін	Валерії	Іванович	Спортівн...
3	90	5909	5909	Фіданюк	Віктор	Вікторов...	Костіна 1...
4	34	4567	4567	Куличков	Іван	Вікторов...	Пушкінск...
5	61	2346	2346	Вілков	Артем	Ігорович	Канатна ...

Рис. 6. 13 — Результат

6.Зніміть привілей INSERT для третього користувача для будь-якої таблиці.

В цьому прикладі я знімаю привілей INSERT (внесення нових даних) для третього користувача (worker1) для таблиці придмети.

```
REVOKE INSERT ON street_segment FROM worker1;
```

REVOKE

Рис. 6. 14 — Результат

Перевірка:

```
SET ROLE 'worker1';
SELECT SESSION_USER, CURRENT_USER ;
```

	session_user name	current_user name
1	postgres	worker1

Рис. 6.15 — Результат перевірки

INSERT INTO driver

VALUES(1,'4599','1111','Лунев','Руслан','Івановичь','Успенская 22')

ERROR: ОШИБКА: нет доступа к таблице driver

Рис. 6.16 — Результат

7.Розподіліть інші привілеї на свій розсуд.

В цьому прикладі я розподілив привілеї, призначивши команду

```
DELETE, INSERT та REFERENCES першому користувачу
GRANT REFERENCES ON carpark TO worker3;
GRANT DELETE ON carpark TO worker3;
GRANT INSERT ON carpark TO worker3;
```

GRANT

Рис. 6.17 — Результат

Перевірка:

```
SET ROLE 'worker3';
SELECT SESSION_USER, CURRENT_USER;
```

	session_user name	current_user name
1	postgres	worker3

Рис. 6.18 — Результат

ВИСНОВКИ

В виконанні курсової роботи з предмету бази даних і знать. Ми закріпили всі знання які вивчали на десцеплінні БД. Всі запити та створення в PgAdmin 4. За допомогою мови програмування SQL ми навчилися створювати бази даних для різних областей сфери життя. З початку потрібно створіти таблиці та заповнити їх. А все потім створювати с ними різні операції, тригерів та користувачів.

Завдання за варіантом було створити базу даних маршрутів громадського транспорту з можливістю зберігання топологічної схеми маршрутів. Ця робота була успішно виконана та отримані знання закріплені завдяки практиці.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Глава М.Г. Методичні вказівки до виконання курсової роботи з курсу “Організація баз даних та знань” для студентів всіх форм навчання спеціальності 122 «Комп’ютерні науки» / Укл.: М.Г. Глава. – Одеса, 2022. – 27 с. – Режим доступу: <http://library.opu.ua>.
2. Малахов Є.В., Блажко О.А., Глава М.Г. Проектування БД та їх реалізація засобами стандартного SQL та PostgreSQL: Навч. посібник для студ. вищих навч. закладів. – О.: ВМВ, 2012. – 248 с.
3. Глава М.Г. Організація баз даних та знань: Конспект лекцій. [Електронний ресурс] – Режим доступу: <http://library.opu.ua>.
4. Глава М.Г. Методичні вказівки до виконання лабораторних робіт з дисципліни „Організація баз даних та знань “. – Режим доступу: <http://library.opu.ua>.
5. Глава М.Г. Методичні вказівки до самостійної роботи з дисципліни „Організація баз даних та знань “. – Режим доступу: <http://library.opu.ua>.
6. https://knowledge.allbest.ru/programming/3c0b65635b2ad68a5c53b88521206c36_0.html
7. <https://dspace.uzhnu.edu.ua/jspui/bitstream/lib/8868/1/sql.pdf>