

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ
(повна назва інституту/факультету)

КАФЕДРА інформатики та програмної інженерії
(повна назва кафедри)

КУРСОВА РОБОТА

з дисципліни «Бази даних»
(назва дисципліни)

на тему: База даних маршрутів громадського транспорту міста

Студента (ки) 2 курсу ІП-13 групи
спеціальності 121 «Інженерія програмного
забезпечення»

Дем'янчука Олександра Петровича
(прізвище та ініціали)

Керівник Ліщук О. В

(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Національна шкала _____

Кількість балів: _____ Оцінка ECTS _____

Члени комісії

(підпис)

(вчене звання, науковий ступінь, прізвище та ініціали)

(підпис)

(вчене звання, науковий ступінь, прізвище та ініціали)

(підпис)

(вчене звання, науковий ступінь, прізвище та ініціали)

Київ – 2022 рік

**Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»**

Факультет Інформатики та обчислювальної техніки
(повна назва)

Кафедра Інформатики та програмної інженерії
(повна назва)

Дисципліна Бази даних

Курс 2 Група ІІІ-13 Семестр 3

**З А В Д А Н Н Я
НА КУРСОВУ РОБОТУ СТУДЕНТУ**

Дем'янчуку Олександр Петровичу

(прізвище, ім'я, по батькові)

1. Тема роботи База даних маршрутів громадського транспорту міста

керівник роботи

Ліщук О. В

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

2. Строк подання студентом роботи 09.01.2023
3. Вихідні дані до роботи Побудовано ER-модель, створено базу даних для маршрутів громадського транспорту міста, розроблено змістовні запити, користувачів бази даних, тригери для взаємодії з БД та функції
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
 - 1) Аналіз предметного середовища
 - 2) Побудова ER-моделі
 - 3) Побудова реляційної схеми з ER-моделі
 - 4) Створення бази даних, у форматі обраної системи управління базою даних
 - 5) Створення користувачів бази даних
 - 6) Імпорт даних з використанням засобів СУБД в створену базу даних
 - 7) Створення мовою SQL запитів
 - 8) Оптимізація роботи запитів
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) ER-модель, реляційна схема БД, ілюстрації роботи запитів, тригерів, тощо
6. Дата видачі завдання 30.10.2022

КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів виконання курсового проекту | Строк виконання етапів проекту | Примітка |
|-------|--|--------------------------------|----------|
| 1 | Аналіз предметного середовища | 24.11.2022 | |
| 2 | Побудова ER-моделі | 06.12.2022 | |
| 3 | Побудова реляційної схеми з ER-моделі | 04.01.2023 | |
| 4 | Створення бази даних, у форматі обраної системи управління базою даних | 04.01.2023 | |
| 5 | Створення користувачів бази даних | 04.01.2023 | |
| 6 | Імпорт даних з використанням засобів СУБД в створену базу даних | 05.01.2023 | |
| 7 | Створення мовою SQL запитів | 07.01.2023 | |
| 8 | Оптимізація роботи запитів | 08.01.2023 | |
| 9 | Оформлення пояснювальної записки | 09.01.2022 | |
| 10 | Захист курсової роботи | 11.01.2022 | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

Студент

_____ Дем'янчук О. П.
(підпис) (прізвище та ініціали)

Керівник роботи

_____ Ліщук О. В.
(підпис) (прізвище та ініціали)

ЗМІСТ

| | |
|---|----|
| ВСТУП | 5 |
| ОСНОВНА ЧАСТИНА | |
| 1) Опис предметного середовища | 6 |
| 2) Постановка задачі | 7 |
| 3) Проектування бази даних | |
| а. Бізнес-правила | 10 |
| б. Вибір сутностей | 10 |
| в. Побудова ER-моделі | 13 |
| 4) Реляційна модель бази даних | 14 |
| 5) Реалізація бази даних | 16 |
| 6) Робота з базою даних | |
| а. Створення ролей та користувачів БД | 24 |
| б. Створення запитів | 25 |
| в. Створення процедур та функцій | 48 |
| г. Створення тригерів..... | 54 |
| д. Оптимізація запитів | 58 |
| 7) Висновок | 60 |
| 8) Список використаних джерел | 61 |

ВСТУП

Людство живе в епоху, де інформація – найцінніший продукт. Інформація знаходиться всюди: в школах, на вулиці, в магазинах, всюди є інформація, навіть у розумних побутових приладах. Важко оминати факт, що об'єми всіх даних у світі зростають до незліченних розмірів, тому необхідна система для збереження, управління та структурування великих об'ємів даних.

Для цього існують бази даних – сервери з великими обсягами пам'яті, та програмна частина баз даних – система керування базами даних (СКБД) та мова для управління БД – SQL (Structured Query Language). Ці технології зараз максимально розповсюджені в зв'язку з обсягами інформації. Сфера життя, яка розглядається у даній курсовій роботі – система маршрутів громадського транспорту міста – потребує систематизації та структуризації задля стабільного руху транспорту та можливості переглянути інформацію про маршрути пасажирами. Отже, маю на меті розробку бази даних для маршрутів громадського транспорту міста

1 — ОПИС ПРЕДМЕТНОГО СЕРЕДОВИЩА

Поглянемо на нашу столицю – Київ. Величезне місто з великою мережею маршрутів як наземних, так і метро. Система – як мурашник, активно працює зранку до ночі кожен день. Тому необхідно зберігати у структурованому вигляді інформацію, що необхідна буде водіям, відповідальним за розробку маршрутів, та навіть звичайним пасажиром.

Маємо три великі категорії транспортних маршрутів, два з яких є наземними: автобуси та трамваї, та одна підземна – метро. Кожна система має свій набір маршрутів, у кожного є інформація про початок та кінець роботи маршруту. Неодмінно, необхідно також зберігати інформацію про наземні транспортні зупинки, тобто назву зупинки, та інформацію про станцію метро: назва та наявність інклюзивних технологій для людей з інвалідністю. Кожен маршрут має список зупинок у порядку прибуття та час, необхідний на прибуття з попередньої зупинки на поточну.

Ще однією важливою характеристикою будь-якого транспортного маршруту є тижневий розклад руху, що містить інформації про інтервали руху транспорту по маршруту. Наприклад, якщо автобуси та трамваї рухаються з фіксованим інтервалом протягом доби, то інтервал руху в метро змінюється, підлаштовуючись під графік навантаженості метрополітену, тому цей фактор варто урахувати при проектуванні.

Для програмної реалізації бази даних послугуватимуся СКБД PostgreSQL, що давно себе зарекомендувала як професійний open-source інструмент управління базами даних, з великою кількістю користувачів у світі.

2 — ПОСТАНОВКА ЗАДАЧІ

Мета курсової роботи - створення бази даних маршрутів громадського транспорту міста, що включає в себе функціонал для редагування даних, та зручного перегляду інформації про маршрути для користувачів

Визначимо для початку структуру описаної бази даних. Отже, основні задачі:

Адміністратор бази даних повинен мати змогу:

- Виконувати будь які дії з редагування, перегляду даних, використання функцій.

Також працівники сфери громадського транспорту необхідні мати можливість:

- Переглянути усю наявну інформацію у базі даних про маршрути та зупинки.

Проектувальникам маршрутів громадського транспорту необхідно мати змогу:

- Переглядати та редагувати інформацію про маршрути, вносити в систему чи видаляти маршрути, редагувати список зупинок маршрутів.

Редагувальники розкладів наземних/підземних маршрутів

- Переглядати та редагувати інформацію про інтервали руху маршрутами наземного чи підземного транспорту;
- Переглядати інформацію про відповідні маршрути та зупинки, що їм належать.

Зрештою, пасажери повинні мати змогу:

- Переглядати інформацію про час і початок роботи маршруту, першу і кінцеву зупинки маршруту, список зупинок будь-якого маршруту;

- отримати інформацію про час прибуття автобусного транспорту вказаного маршруту на вказану зупинку, через інтервали, вказані у розкладі;
- отримати інформацію про час прибуття потягів вказаного гілки метро на вказану станцію, через інтервали, вказані у розкладі.

3 —ПРОЕКТУВАННЯ БАЗИ ДАНИХ

Бізнес правила

Для правильної роботи БД необхідно ввести певні бізнес-правила:

- 1) Час початку роботи транспорту не має перевищувати час закінчення його роботи та дана часова інформація має обов'язково бути:
- 2) Інформація про зупинки має містити як мінімум назву цієї зупинки, на станціях метро, за можливості, надати інформацію про інклюзивні технології на станції;
- 3) В списках зупинок для маршруту необхідна інформація про позицію зупинки/станції у маршруті та час, необхідний на прибуття до неї;
- 4) Рядки розкладу транспорту обов'язково повинні містити інформацію про час початку дії інтервалу руху, час кінця його дії, та, власне, довжину інтервалу руху.

Вибір сутностей

Вибір сутностей для Баз Даних виглядатиме наступним чином:

- Дні тижня
- Розклад руху автобусів
- Розклад руху трамваїв
- Розклад руху потягів метро
- Автобусний маршрут
- Трамвайний маршрут
- Лінія метро

- Наземна зупинка
- Підземна станція
- Списки зупинок автобусних маршрутів
- Списки зупинок трамвайних маршрутів
- Списки станцій ліній метро

Атрибути цих сутностей наведені у наступній таблиці:

Таблиця 3.1 - Список сутностей та їх атрибутів

| Сутність | Атрибути |
|----------------------------|--|
| Дні тижня | ID днів тижня Тип дня тижня |
| Розклад руху автобусів | ID рядка ID типу дня тижня ID автобусного маршруту Час початку дії інтервалу руху Час закінчення дії інтервалу руху Тривалість інтервалу руху |
| Розклад руху трамваїв | ID рядка ID типу дня тижня ID трамвайного маршруту Час початку дії інтервалу руху Час закінчення дії інтервалу руху Тривалість інтервалу руху |
| Розклад руху потягів метро | ID рядка |

| | |
|-------------------------------------|--|
| | ID типу дня тижня ID трамвайного маршруту Час початку дії інтервалу руху Час закінчення дії інтервалу руху Тривалість інтервалу руху |
| Автобусний маршрут | ID маршруту Номенклатура маршруту Час початку роботи Час кінця роботи |
| Трамвайний маршрут | ID маршруту Номенклатура маршруту Час початку роботи Час кінця роботи |
| Лінія метро | ID лінії Номенклатура лінії Час початку роботи Час кінця роботи |
| Наземна зупинка | ID зупинки Назва зупинки |
| Підземна станція | ID станції Назва станції Наявність інклюзивних технологій |
| Списки зупинок автобусних маршрутів | ID автобусного маршруту ID зупинки Порядковий номер зупинки Час до прибуття |
| Списки зупинок трамвайних маршрутів | ID трамвайного маршруту ID зупинки Порядковий номер зупинки |

| | |
|----------------------------|---|
| | Час до прибуття |
| Списки станцій ліній метро | ID ліній метро ID станції Порядковий номер станції Час до прибуття |

Побудова ER-моделі

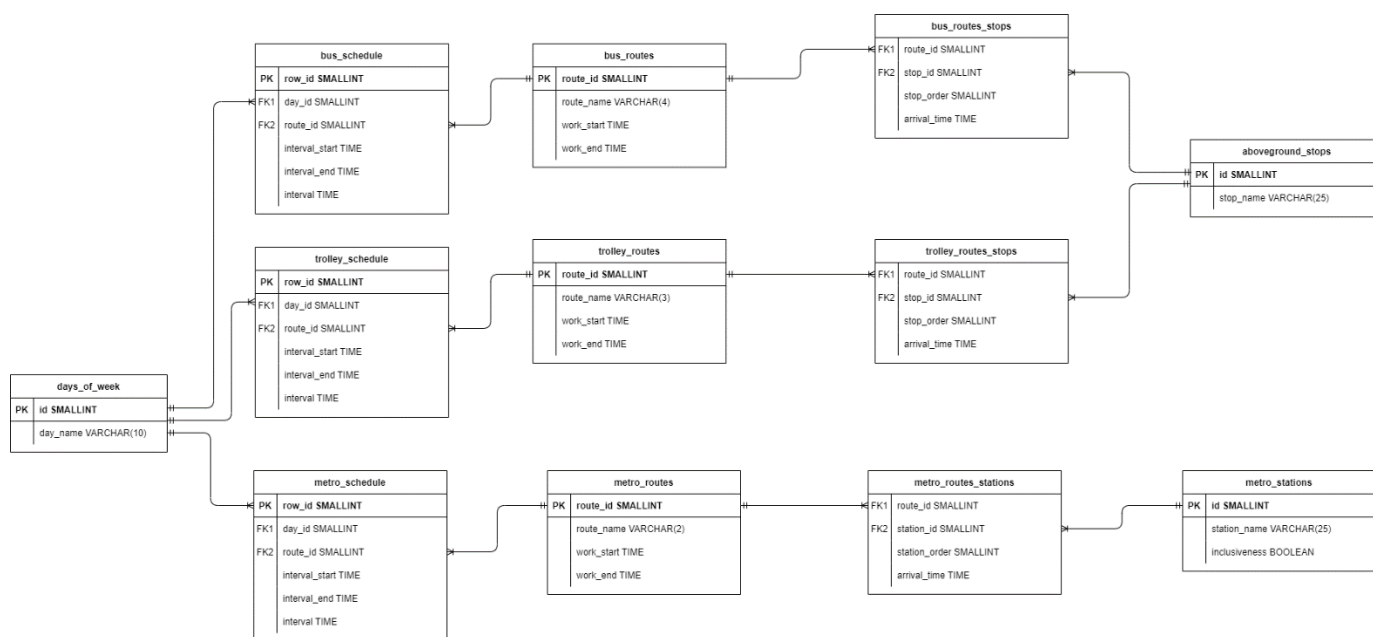


Рисунок 3.1 – ER-діаграма бази даних

4 —РЕЛЯЦІЙНА МОДЕЛЬ БАЗИ ДАНИХ

Згенерована реляційна модель БД

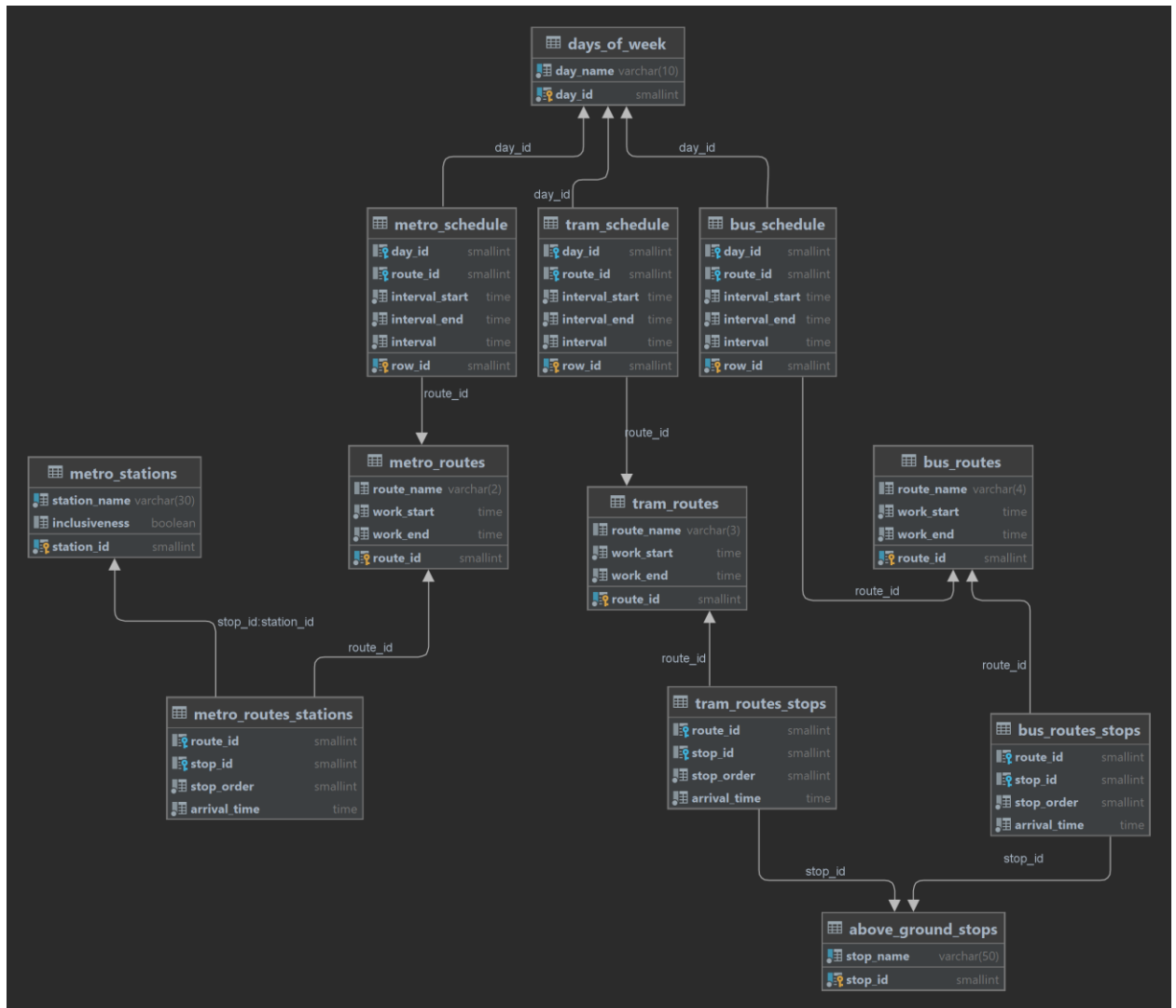


Рисунок 3.2 – Реляційна модель БД

На даній схемі видно, що база даних знаходиться у 3 нормальній формі, а саме: залежності будуються виключно від основних ключів таблиць.

Після детального аналізу ER-моделі, було програмно реалізовано базу даних з таблицями, що відповідають сутностям діаграми. Кожна таблиця має власний первинний ключ – поле, що матиме обмеження PRIMARY KEY. Зв'язуються таблиці шляхом додавання обмеження FOREIGN KEY на відповідні поля. У обмеженні FOREIGN KEY використані різні механізми видалення зовнішніх ключів: NO ACTION для збереження цілісності таблиць розкладу транспорту, та CASCADE, для інших зовнішніх ключів у базі, де

логіка каскадного видалення цілком співпадає з логікою реального світу. Кожна таблиця містить такі поля, які не можуть бути порожніми згідно з бізнес-правилами – це реалізується обмеженням поля NOT NULL. В базі даних також реалізована перевірка коректності вводу для деяких полів за допомогою обмеження CHECK.

5 — РЕАЛІЗАЦІЯ БАЗИ ДАНИХ

Код створення бази даних, таблиць БД та налаштування зв'язків між ними

```
drop database if exists coursework;
```

```
create database coursework;
```

```
DROP TABLE IF EXISTS bus_routes_stops;
```

```
DROP TABLE IF EXISTS tram_routes_stops;
```

```
DROP TABLE IF EXISTS metro_routes_stations;
```

```
DROP TABLE IF EXISTS bus_schedule;
```

```
DROP TABLE IF EXISTS tram_schedule;
```

```
DROP TABLE IF EXISTS metro_schedule;
```

```
DROP TABLE IF EXISTS bus_routes;
```

```
DROP TABLE IF EXISTS tram_routes;
```

```
DROP TABLE IF EXISTS metro_routes;
```

```
DROP TABLE IF EXISTS above_ground_stops;
```

```
DROP TABLE IF EXISTS metro_stations;
```

```
DROP TABLE IF EXISTS days_of_week;
```

```
CREATE TABLE days_of_week
```

```
(
```

```
    day_id smallint,
```

```
    day_name varchar(10) NOT NULL UNIQUE,
```

```
    PRIMARY KEY (day_id)
```


);

CREATE TABLE above_ground_stops

(

stop_id smallserial,

stop_name varchar(50) NOT NULL UNIQUE,

PRIMARY KEY (stop_id)

);

CREATE TABLE metro_stations

(

station_id smallserial,

station_name varchar(30) NOT NULL UNIQUE,

inclusiveness BOOLEAN,

PRIMARY KEY (station_id)

);

CREATE TABLE bus_routes

(

route_id smallserial,

route_name varchar(4),

work_start time NOT NULL,

```
work_end time NOT NULL,  
  
PRIMARY KEY (route_id),  
  
CONSTRAINT bus_time_check CHECK (work_start < work_end)  
  
);
```

```
CREATE TABLE tram_routes  
  
(  
  
    route_id smallserial,  
  
    route_name varchar(3),  
  
    work_start time NOT NULL,  
  
    work_end time NOT NULL,  
  
    PRIMARY KEY (route_id),  
  
    CONSTRAINT tram_time_check CHECK (work_start < work_end)  
  
);
```

```
CREATE TABLE metro_routes  
  
(  
  
    route_id smallserial,  
  
    route_name varchar(2),  
  
    work_start time NOT NULL,  
  
    work_end time NOT NULL,  
  
    PRIMARY KEY (route_id),
```

```
CONSTRAINT metro_time_check CHECK (work_start < work_end)

);

CREATE TABLE bus_schedule

(
    row_id smallserial,
    day_id smallint,
    route_id smallint,
    interval_start time NOT NULL,
    interval_end time NOT NULL,
    interval time NOT NULL,
    PRIMARY KEY (row_id),
    FOREIGN KEY (day_id) REFERENCES days_of_week (day_id) ON DELETE
    NO ACTION,
    FOREIGN KEY (route_id) REFERENCES bus_routes (route_id) ON DELETE
    CASCADE
);
```

```
CREATE TABLE tram_schedule

(
    row_id smallserial,
    day_id smallint,
    route_id smallint,
```

```

interval_start time NOT NULL,

interval_end time NOT NULL,

interval time NOT NULL,

PRIMARY KEY (row_id),

FOREIGN KEY (day_id) REFERENCES days_of_week (day_id) ON DELETE
NO ACTION,

FOREIGN KEY (route_id) REFERENCES tram_routes (route_id) ON
DELETE CASCADE

);

```

```

CREATE TABLE metro_schedule

```

```

(

row_id smallserial,

day_id smallint,

route_id smallint,

interval_start time NOT NULL,

interval_end time NOT NULL,

interval time NOT NULL,

PRIMARY KEY (row_id),

FOREIGN KEY (day_id) REFERENCES days_of_week (day_id) ON DELETE
NO ACTION,

FOREIGN KEY (route_id) REFERENCES metro_routes (route_id) ON
DELETE CASCADE

```

);

```
CREATE TABLE bus_routes_stops(  
    route_id smallint,  
    stop_id smallint,  
    stop_order smallint NOT NULL,  
    arrival_time time NOT NULL,  
    FOREIGN KEY (route_id) REFERENCES bus_routes(route_id) ON DELETE  
    CASCADE,  
    FOREIGN KEY (stop_id) REFERENCES above_ground_stops(stop_id) ON  
    DELETE CASCADE  
);
```

```
CREATE TABLE tram_routes_stops(  
    route_id smallint,  
    stop_id smallint,  
    stop_order smallint NOT NULL,  
    arrival_time time NOT NULL,  
    FOREIGN KEY (route_id) REFERENCES tram_routes(route_id) ON DELETE  
    CASCADE,  
    FOREIGN KEY (stop_id) REFERENCES above_ground_stops(stop_id) ON  
    DELETE CASCADE  
);
```

```

CREATE TABLE metro_routes_stations(

    route_id smallint,

    stop_id smallint,

    stop_order smallint NOT NULL,

    arrival_time time NOT NULL,

    FOREIGN KEY (route_id) REFERENCES metro_routes(route_id) ON
DELETE CASCADE,

    FOREIGN KEY (stop_id) REFERENCES metro_stations(station_id) ON
DELETE CASCADE

);

```

Код імпортування даних в БД

У даній курсовій роботі було використано метод імпорту даних з файлів формату .csv. Перевагами такого методу є:

- Зручність роботи з форматом даних, у кожному офісному пакеті є інструмент для роботи з такими файлами;
- Забезпечення збереження даних: базу даних можна буде наповнити даними заново з легкістю, або додати нові таким ж зручним методом.

Код використання даного методу у роботі:

```

COPY days_of_week(day_id, day_name) FROM
'D:\Labs\kursach_db\csv\days_of_week.csv' DELIMITER ';' CSV HEADER;

COPY metro_stations(station_name, inclusiveness) FROM
'D:\Labs\kursach_db\csv\metro_stations.csv' DELIMITER ';' CSV HEADER;

```

COPY above_ground_stops(stop_name) FROM

'D:\Labs\kursach_db\csv\above_ground_stops.csv' DELIMITER ';' CSV
HEADER;

COPY metro_routes(route_name, work_start, work_end) FROM

'D:\Labs\kursach_db\csv\metro_routes.csv' DELIMITER ';' CSV HEADER;

COPY bus_routes(route_name, work_start, work_end) FROM

'D:\Labs\kursach_db\csv\bus_routes.csv' DELIMITER ';' CSV HEADER;

COPY tram_routes(route_name, work_start, work_end) FROM

'D:\Labs\kursach_db\csv\tram_routes.csv' DELIMITER ';' CSV HEADER;

COPY bus_schedule(day_id, route_id, interval_start, interval_end, interval)

FROM 'D:\Labs\kursach_db\csv\bus_schedule.csv' DELIMITER ';' CSV
HEADER;

COPY tram_schedule(day_id, route_id, interval_start, interval_end, interval)

FROM 'D:\Labs\kursach_db\csv\tram_schedule.csv' DELIMITER ';' CSV
HEADER;

COPY metro_schedule(day_id, route_id, interval_start, interval_end, interval)

FROM 'D:\Labs\kursach_db\csv\metro_schedule.csv' DELIMITER ';' CSV
HEADER;

COPY bus_routes_stops(route_id, stop_id, stop_order, arrival_time) FROM

'D:\Labs\kursach_db\csv\bus_routes_stops.csv' DELIMITER ';' CSV HEADER;

COPY tram_routes_stops(route_id, stop_id, stop_order, arrival_time) FROM

'D:\Labs\kursach_db\csv\tram_routes_stops.csv' DELIMITER ';' CSV HEADER;

COPY metro_routes_stations(route_id, stop_id, stop_order, arrival_time) FROM

'D:\Labs\kursach_db\csv\metro_routes_stations.csv' DELIMITER ';' CSV
HEADER;

6 — РОБОТА З БАЗОЮ ДАНИХ

a) Створення ролей та користувачів БД

```
CREATE ROLE db_admin PASSWORD '1234';
```

```
CREATE ROLE db_viewer PASSWORD '1234';
```

```
CREATE ROLE abg_route_editor PASSWORD '1234';
```

```
CREATE ROLE metro_route_editor PASSWORD '1234';
```

```
CREATE ROLE abg_schedule_editor PASSWORD '1234';
```

```
CREATE ROLE metro_schedule_editor PASSWORD '1234';
```

```
CREATE ROLE passenger;
```

```
GRANT ALL ON ALL TABLES IN SCHEMA public TO db_admin;
```

```
GRANT ALL ON ALL FUNCTIONS IN SCHEMA public TO db_admin;
```

```
GRANT SELECT ON ALL TABLES IN SCHEMA public TO db_viewer;
```

```
GRANT ALL ON bus_routes, tram_routes, bus_routes_stops, tram_routes_stops  
TO abg_route_editor;
```

```
GRANT SELECT ON above_ground_stops, bus_first_last, bus_route_list,  
tram_first_last, tram_route_list TO abg_route_editor;
```

```
GRANT ALL ON metro_routes, metro_routes_stations TO metro_route_editor;
```

```
GRANT SELECT ON metro_stations, metro_first_last, metro_route_list TO  
metro_route_editor;
```



```
GRANT ALL ON bus_schedule, tram_schedule TO abg_schedule_editor;
```

```
GRANT SELECT ON bus_routes, tram_routes, bus_routes_stops,  
tram_routes_stops TO abg_schedule_editor;
```

```
GRANT ALL ON metro_schedule TO metro_schedule_editor;
```

```
GRANT SELECT ON metro_routes, metro_routes_stations TO  
metro_schedule_editor;
```

```
GRANT SELECT ON bus_routes, tram_routes,  
  
    bus_route_list, tram_route_list, metro_route_list,  
  
    bus_first_last, tram_first_last,  
  
    bus_a_to_b, tram_a_to_b,  
  
    bus_schedule, tram_schedule, metro_schedule  
TO passenger;
```

```
GRANT EXECUTE ON FUNCTION display_bus_arrivals(integer, integer) TO  
passenger;
```

```
GRANT EXECUTE ON FUNCTION display_metro_arrivals(integer, integer) TO  
passenger;
```

б) Створення запитів

1. Представлення, що містить інформацію про автобусні маршрути з їх першими і останніми зупинками

```
CREATE OR REPLACE VIEW bus_first_last
```

AS

SELECT bus_routes.route_id id,

bus_routes.route_name route,

ags1.stop_id s1,

ags1.stop_name first_stop,

ags2.stop_id s2,

ags2.stop_name last_stop

FROM bus_routes

JOIN above_ground_stops ags1 ON ags1.stop_id = (SELECT
bus_routes_stops.stop_id

FROM bus_routes_stops

WHERE bus_routes_stops.route_id =
bus_routes.route_id

AND stop_order = 1)

JOIN above_ground_stops ags2 ON ags2.stop_id = (SELECT
bus_routes_stops.stop_id

FROM bus_routes_stops

WHERE bus_routes_stops.route_id =
bus_routes.route_id

AND stop_order = (SELECT
COUNT(stop_id)

FROM bus_routes_stops

WHERE
bus_routes_stops.route_id = bus_routes.route_id));

```

SELECT *

FROM bus_first_last;

SELECT *

FROM bus_first_last

WHERE first_stop = 'Symyrenka St';

```

| | id | route | s1 | first_stop | s2 | last_stop |
|---|----|-------|----|-------------------|----|----------------------|
| 1 | 1 | 1T | 1 | Symyrenka St | 24 | Starovokzalna St |
| 2 | 2 | 2 | 25 | Bulhakova St | 47 | Sholudenka St |
| 3 | 3 | 2T | 1 | Symyrenka St | 49 | Kiltseva Road |
| 4 | 4 | 3T | 49 | Kiltseva Road | 24 | Starovokzalna St |
| 5 | 5 | 5T | 50 | Senzha Lyfaria St | 73 | Troieshchyna Station |
| 6 | 6 | 6 | 74 | Darnytska Square | 95 | Churylivska St |

Рисунок 6.1 – Результат виконання запиту

| Output coursework.public.bus_first_last | | | | | | |
|---|----|-------|----|--------------|----|------------------|
| | id | route | s1 | first_stop | s2 | last_stop |
| 1 | 1 | 1T | 1 | Symyrenka St | 24 | Starovokzalna St |
| 2 | 3 | 2T | 1 | Symyrenka St | 49 | Kiltseva Road |

Рисунок 6.2 – Результат виконання запиту

2. Запит для виведення трамвайних маршрутів з їх першими і останніми зупинками

```
CREATE OR REPLACE VIEW tram_first_last
```

```
AS
```

```

SELECT tram_routes.route_id    id,

       tram_routes.route_name AS route,

       ags1.stop_id            s1,

       ags1.stop_name          first_stop,

```

```

        ags2.stop_id      s2,

        ags2.stop_name    last_stop

FROM tram_routes

        JOIN above_ground_stops ags1 ON ags1.stop_id = (SELECT
tram_routes_stops.stop_id

                                FROM tram_routes_stops

                                WHERE tram_routes_stops.route_id =

tram_routes.route_id

                                AND stop_order = 1)

        JOIN above_ground_stops ags2 ON ags2.stop_id = (SELECT
tram_routes_stops.stop_id

                                FROM tram_routes_stops

                                WHERE tram_routes_stops.route_id =

tram_routes.route_id

                                AND stop_order = (SELECT

COUNT(stop_id)

                                FROM tram_routes_stops

                                WHERE

tram_routes_stops.route_id = tram_routes.route_id));

SELECT *

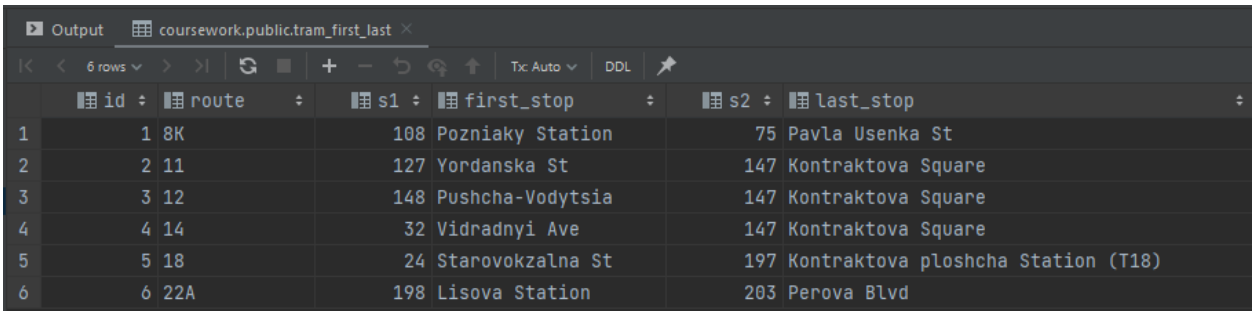
FROM tram_first_last;

SELECT *

FROM tram_first_last

```

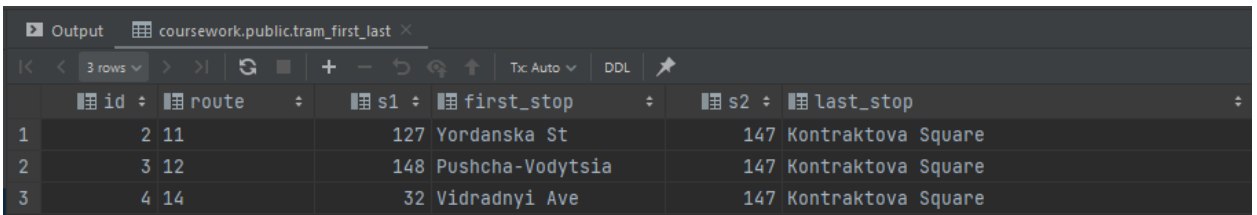
WHERE last_stop = 'Kontraktova Square';



The screenshot shows a database query result with 6 rows. The columns are: id, route, s1, first_stop, s2, and last_stop. The data is as follows:

| | id | route | s1 | first_stop | s2 | last_stop |
|---|----|-------|-----|------------------|-----|------------------------------------|
| 1 | 1 | 8K | 108 | Pozniaky Station | 75 | Pavla Usenka St |
| 2 | 2 | 11 | 127 | Yordanska St | 147 | Kontraktova Square |
| 3 | 3 | 12 | 148 | Pushcha-Vodytsia | 147 | Kontraktova Square |
| 4 | 4 | 14 | 32 | Vidradnyi Ave | 147 | Kontraktova Square |
| 5 | 5 | 18 | 24 | Starovokzalna St | 197 | Kontraktova ploshcha Station (T18) |
| 6 | 6 | 22A | 198 | Lisova Station | 203 | Perova Blvd |

Рисунок 6.3 – Результат виконання запиту



The screenshot shows a database query result with 3 rows. The columns are: id, route, s1, first_stop, s2, and last_stop. The data is as follows:

| | id | route | s1 | first_stop | s2 | last_stop |
|---|----|-------|-----|------------------|-----|--------------------|
| 1 | 2 | 11 | 127 | Yordanska St | 147 | Kontraktova Square |
| 2 | 3 | 12 | 148 | Pushcha-Vodytsia | 147 | Kontraktova Square |
| 3 | 4 | 14 | 32 | Vidradnyi Ave | 147 | Kontraktova Square |

Рисунок 6.4 – Результат виконання запиту

3. Представлення для виведення інформації про лінії метро з їх першими і останніми станціями

CREATE OR REPLACE VIEW metro_first_last

AS

SELECT metro_routes.route_id id,

metro_routes.route_name AS route,

ms1.station_id s1,

ms1.station_name AS first_stop,

ms2.station_id s2,

ms2.station_name AS last_stop

FROM metro_routes

JOIN metro_stations ms1 ON ms1.station_id = (SELECT
metro_routes_stations.stop_id

```

FROM metro_routes_stations

WHERE metro_routes_stations.route_id =

metro_routes.route_id

AND stop_order = 1)

JOIN metro_stations ms2 ON ms2.station_id = (SELECT
metro_routes_stations.stop_id

FROM metro_routes_stations

WHERE metro_routes_stations.route_id =

metro_routes.route_id

AND stop_order = (SELECT COUNT(stop_id)

FROM metro_routes_stations

WHERE

metro_routes_stations.route_id = metro_routes.route_id));

SELECT route, first_stop, last_stop

FROM metro_first_last;

SELECT mfl.id, mfl.route, mfl.first_stop, mfl.last_stop, ms.inclusiveness inclusive

FROM metro_first_last mfl

JOIN metro_stations ms ON ms.station_id = mfl.s2;

```

The screenshot shows a database query result with 3 rows. The columns are route, first_stop, and last_stop. The data is as follows:

| | route | first_stop | last_stop |
|---|-------|-----------------|------------------|
| 1 | M1 | Akademmistechko | Lisova |
| 2 | M2 | Heroiv Dnipra | Teremky |
| 3 | M3 | Syrets | Chervoniy Khutir |

Рисунок 6.5 – Результат виконання запиту

The screenshot shows a database query result with 3 rows. The columns are id, route, first_stop, last_stop, and inclusive. The data is as follows:

| | id | route | first_stop | last_stop | inclusive |
|---|----|-------|-----------------|------------------|-----------|
| 1 | 1 | M1 | Akademmistechko | Lisova | • true |
| 2 | 2 | M2 | Heroiv Dnipra | Teremky | • true |
| 3 | 3 | M3 | Syrets | Chervoniy Khutir | • true |

Рисунок 6.6– Результат виконання запиту

4. Запит для виведення автобусних маршрутів та кількість зупинок кожного

```
SELECT bus_routes.route_id AS n, bus_routes.route_name AS route,
COUNT(brs.stop_order) AS stops_amount
```

```
FROM bus_routes
```

```
JOIN bus_routes_stops brs ON bus_routes.route_id = brs.route_id
```

```
GROUP BY bus_routes.route_id
```

```
ORDER BY bus_routes.route_id;
```

The screenshot shows a database query result with 6 rows. The columns are n, route, and stops_amount. The data is as follows:

| | n | route | stops_amount |
|---|---|-------|--------------|
| 1 | 1 | 1T | 24 |
| 2 | 2 | 2 | 24 |
| 3 | 3 | 2T | 11 |
| 4 | 4 | 3T | 16 |
| 5 | 5 | 5T | 29 |
| 6 | 6 | 6 | 23 |

Рисунок 6.7 – Результат виконання запиту

5. Представлення для виведення списку зупинок певного трамвайного маршруту

```
CREATE OR REPLACE VIEW tram_route_list
```

```
AS
```

```
SELECT tr.route_id r_id, tram_routes_stops.stop_order AS n, ags.stop_name  
stop_name
```

```
FROM tram_routes_stops
```

```
    JOIN above_ground_stops ags ON tram_routes_stops.stop_id = ags.stop_id
```

```
    JOIN tram_routes tr ON tr.route_id = tram_routes_stops.route_id
```

```
ORDER BY r_id, n;
```

```
SELECT n, stop_name
```

```
FROM tram_route_list
```

```
WHERE r_id = 3;
```


| | n | stop_name |
|----|----|--------------------------|
| 1 | 1 | Pushcha-Vodytsia |
| 2 | 2 | 13-a liniia |
| 3 | 3 | 11-a liniia |
| 4 | 4 | 9-a liniia |
| 5 | 5 | 7-a liniia |
| 6 | 6 | 6-a liniia |
| 7 | 7 | Park Pushcha-Vodytsia |
| 8 | 8 | Hospital invalidiv VVV |
| 9 | 9 | 3-ia liniia |
| 10 | 10 | 2-a liniia |
| 11 | 11 | 1-a liniia |
| 12 | 12 | Miska St |
| 13 | 13 | Upravlinnia lisnytstva |
| 14 | 14 | Lisnytstvo |
| 15 | 15 | Hurtozhytok |
| 16 | 16 | Tarasa Shevchenka Square |
| 17 | 17 | Poliarna St |

Рисунок 6.8 – Результат виконання запити

6. Представлення для виведення списку зупинок певного автобусного маршруту

```
CREATE OR REPLACE VIEW bus_route_list
```

```
AS
```

```
SELECT br.route_id b_id, bus_routes_stops.stop_order n, ags.stop_name  
stop_name
```

```
FROM bus_routes_stops
```

```
JOIN bus_routes br ON br.route_id = bus_routes_stops.route_id
```

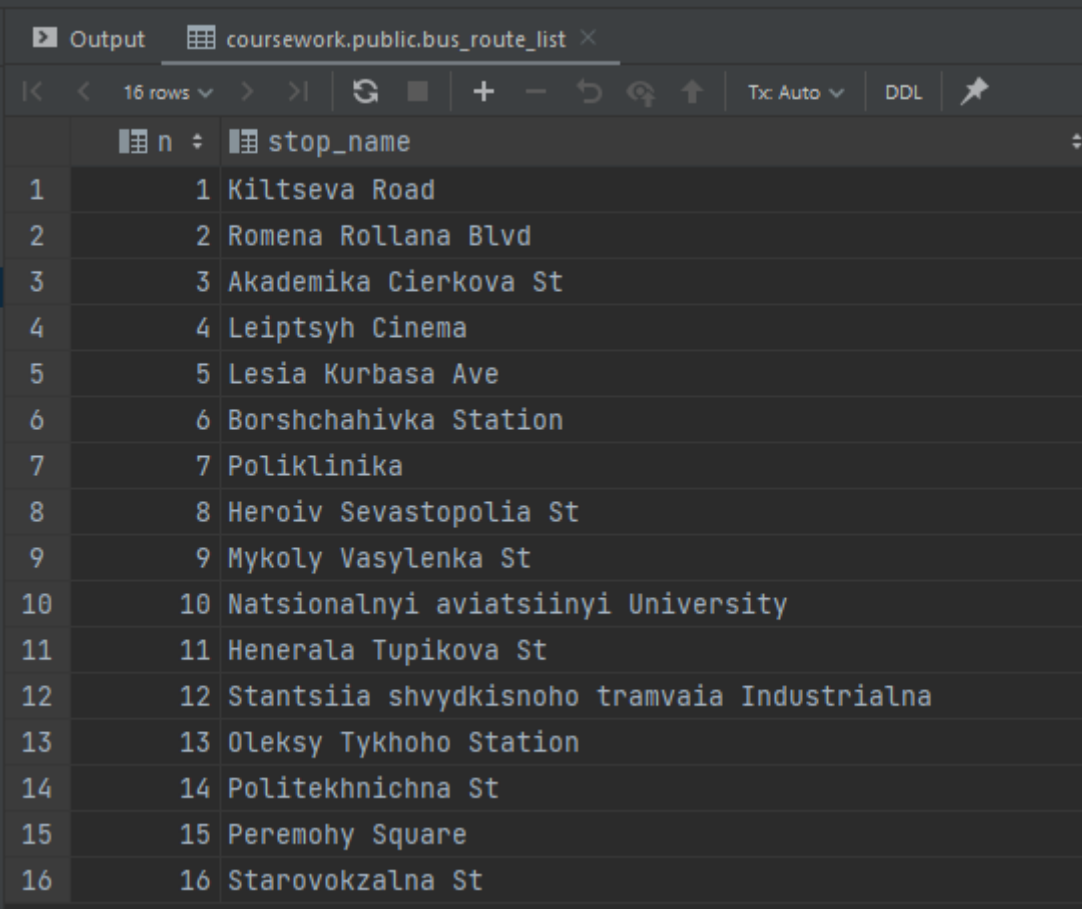
```
JOIN above_ground_stops ags ON ags.stop_id = bus_routes_stops.stop_id
```

```
ORDER BY b_id, n;
```

```
SELECT n, stop_name
```

```
FROM bus_route_list
```

```
WHERE b_id = 4;
```



| | n | stop_name |
|----|----|--|
| 1 | 1 | Kiltseva Road |
| 2 | 2 | Romena Rollana Blvd |
| 3 | 3 | Akademika Cierkova St |
| 4 | 4 | Leiptsyh Cinema |
| 5 | 5 | Lesia Kurbasa Ave |
| 6 | 6 | Borshchahivka Station |
| 7 | 7 | Poliklinika |
| 8 | 8 | Heroiv Sevastopolia St |
| 9 | 9 | Mykoly Vasylenska St |
| 10 | 10 | Natsionalnyi aviatsiinyi University |
| 11 | 11 | Henerala Tupikova St |
| 12 | 12 | Stantsiia shvydkisnoho tramvaia Industrialna |
| 13 | 13 | Oleksy Tykhoho Station |
| 14 | 14 | Politekhnichna St |
| 15 | 15 | Peremohy Square |
| 16 | 16 | Starovokzalna St |

Рисунок 6.9 – Результат виконання запиту

7. Представлення для виведення списку зупинок певної лінії метро

```
CREATE OR REPLACE VIEW metro_route_list
```

```
AS
```

```
SELECT mr.route_id m_id, mrs.stop_order n, ms.station_name stop_name,
ms.inclusiveness inclusive
```

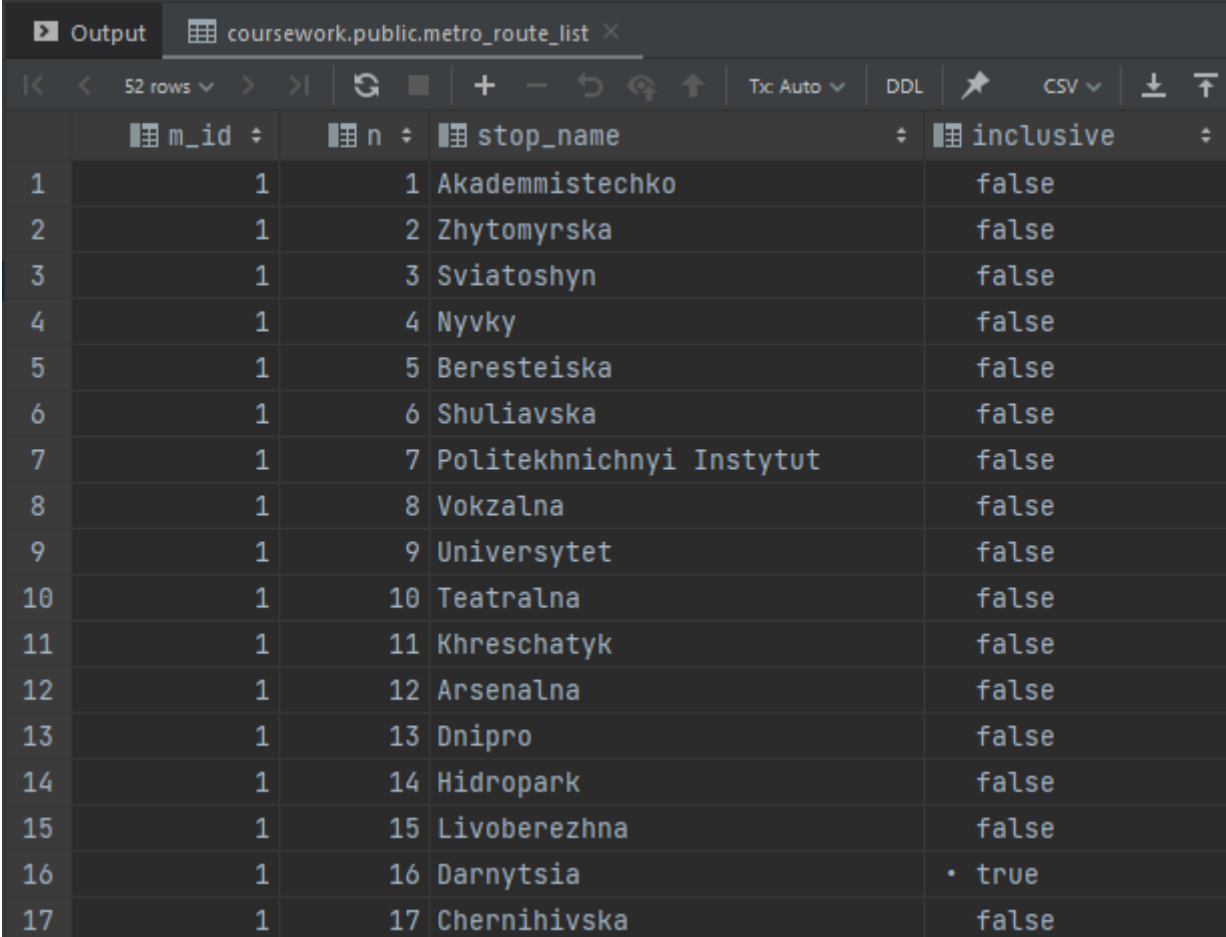
```
FROM metro_routes_stations mrs
```

```
JOIN metro_routes mr ON mr.route_id = mrs.route_id
```

```
JOIN metro_stations ms ON ms.station_id = mrs.stop_id
```

```
ORDER BY m_id, n;
```

```
SELECT * FROM metro_route_list;
```



| | m_id | n | stop_name | inclusive |
|----|------|----|--------------------------|-----------|
| 1 | 1 | 1 | Akademmistechko | false |
| 2 | 1 | 2 | Zhytomyrska | false |
| 3 | 1 | 3 | Sviatoshyn | false |
| 4 | 1 | 4 | Nyvky | false |
| 5 | 1 | 5 | Beresteiska | false |
| 6 | 1 | 6 | Shuliavska | false |
| 7 | 1 | 7 | Politekhnychnyi Instytut | false |
| 8 | 1 | 8 | Vokzalna | false |
| 9 | 1 | 9 | Universytet | false |
| 10 | 1 | 10 | Teatralna | false |
| 11 | 1 | 11 | Khreschatyk | false |
| 12 | 1 | 12 | Arsenalna | false |
| 13 | 1 | 13 | Dnipro | false |
| 14 | 1 | 14 | Hidropark | false |
| 15 | 1 | 15 | Livoberezhna | false |
| 16 | 1 | 16 | Darnytsia | true |
| 17 | 1 | 17 | Chernihivska | false |

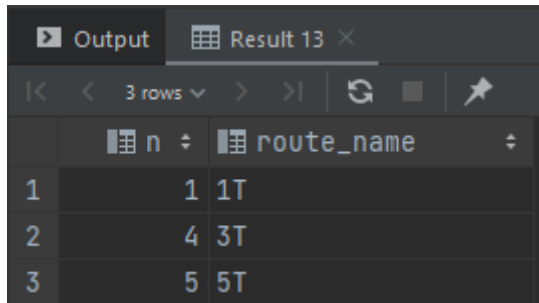
Рисунок 6.10 – Результат виконання запиту

8. Запит для виведення усіх автобусних маршрутів, що проходять через певну зупинку

```

SELECT bus_routes_stops.route_id AS n, br.route_name
FROM bus_routes_stops
      JOIN bus_routes br ON bus_routes_stops.route_id = br.route_id
WHERE stop_id = 15;

```



| | n | route_name |
|---|---|------------|
| 1 | 1 | 1T |
| 2 | 4 | 3T |
| 3 | 5 | 5T |

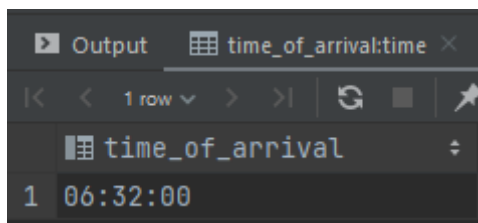
Рисунок 6.11 – Результат виконання запиту

9. Запит для виведення часу першого прибуття на першу зупинку вказаного автобусного маршруту

```

SELECT bus_routes.work_start + bus_routes_stops.arrival_time::interval AS
time_of_arrival
FROM bus_routes
      JOIN bus_routes_stops ON bus_routes.route_id = bus_routes_stops.route_id
AND bus_routes_stops.stop_order = 1
WHERE bus_routes.route_id = 2;

```



| | time_of_arrival |
|---|-----------------|
| 1 | 06:32:00 |

Рисунок 6.12 – Результат виконання запиту

10. Запит для виведення часу першого прибуття на кінцеву зупинку трамвайних маршрутів

```

SELECT tr.route_name route, (tr.work_start + SUM(trs.arrival_time))
time_of_arrival

FROM tram_routes tr

        JOIN tram_routes_stops trs ON tr.route_id = trs.route_id AND stop_order =
(SELECT COUNT(trs1.stop_order)

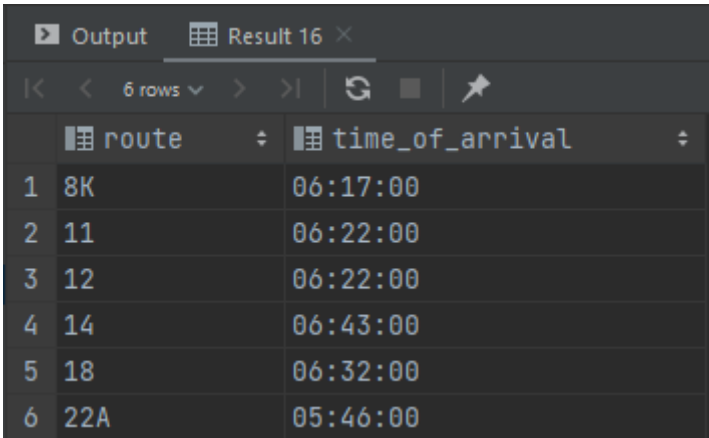
                                FROM tram_routes_stops trs1

                                WHERE trs1.route_id =

tr.route_id)

GROUP BY tr.route_id;

```



The screenshot shows a database query result with 6 rows. The columns are 'route' and 'time_of_arrival'. The data is as follows:

| | route | time_of_arrival |
|---|-------|-----------------|
| 1 | 8K | 06:17:00 |
| 2 | 11 | 06:22:00 |
| 3 | 12 | 06:22:00 |
| 4 | 14 | 06:43:00 |
| 5 | 18 | 06:32:00 |
| 6 | 22A | 05:46:00 |

Рисунок 6.13 – Результат виконання запиту

11. Представлення для виведення часу першого прибуття на вказану зупинку вказаного автобусного маршруту

```

CREATE VIEW bus_nth_stop_first_arrival
AS

SELECT bus_routes.route_id                r_id,

        bus_routes.route_name              r_name,

        brs1.stop_order                    n_stop,

        (bus_routes.work_start + SUM(brs2.arrival_time)) AS time_of_arrival

```

```

FROM bus_routes

    JOIN bus_routes_stops brs1 ON brs1.route_id = bus_routes.route_id

    JOIN bus_routes_stops brs2 ON brs2.stop_order <= brs1.stop_order

GROUP BY r_id, n_stop;

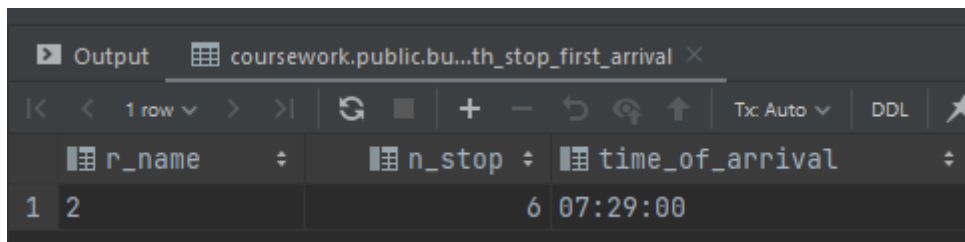
SELECT r_name, n_stop, time_of_arrival

FROM bus_nth_stop_first_arrival

WHERE r_id = 2

    AND n_stop = 6;

```



| | r_name | n_stop | time_of_arrival |
|---|--------|--------|-----------------|
| 1 | 2 | 6 | 07:29:00 |

Рисунок 6.14 – Результат виконання запиту

12. Запит для виведення часу першого прибуття на вказану зупинку
вказаного трамвайного маршруту

```

CREATE VIEW tram_nth_stop_first_arrival

AS

SELECT tram_routes.route_id                r_id,

    tram_routes.route_name                r_name,

    trs1.stop_order                      n_stop,

    (tram_routes.work_start + SUM(trs2.arrival_time)) AS time_of_arrival

FROM tram_routes

```

```

JOIN tram_routes_stops trs1 ON trs1.route_id = tram_routes.route_id

JOIN tram_routes_stops trs2 ON trs2.stop_order <= trs1.stop_order

GROUP BY r_id, n_stop;

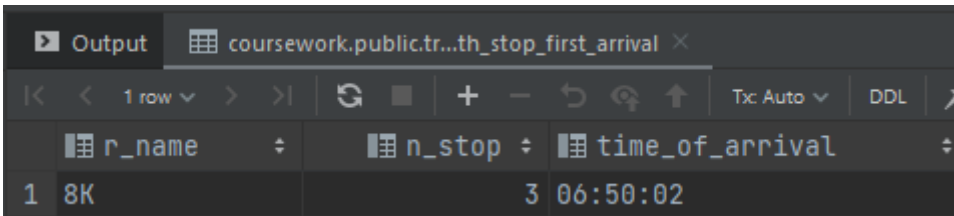
SELECT r_name, n_stop, time_of_arrival

FROM tram_nth_stop_first_arrival

WHERE r_id = 1

AND n_stop = 3;

```



| | r_name | n_stop | time_of_arrival |
|---|--------|--------|-----------------|
| 1 | 8K | 3 | 06:50:02 |

Рисунок 6.15 – Результат виконання запиту

13. Запит для виведення часу першого прибуття на вказану станцію вказаної лінії метро

```

CREATE VIEW metro_nth_stop_first_arrival

AS

SELECT metro_routes.route_id                r_id,

       metro_routes.route_name              r_name,

       mrs1.stop_order                      n_stop,

       (metro_routes.work_start + SUM(mrs2.arrival_time)) AS time_of_arrival

FROM metro_routes

       JOIN metro_routes_stations mrs1 ON mrs1.route_id = metro_routes.route_id

       JOIN metro_routes_stations mrs2 ON mrs2.stop_order <= mrs1.stop_order

```

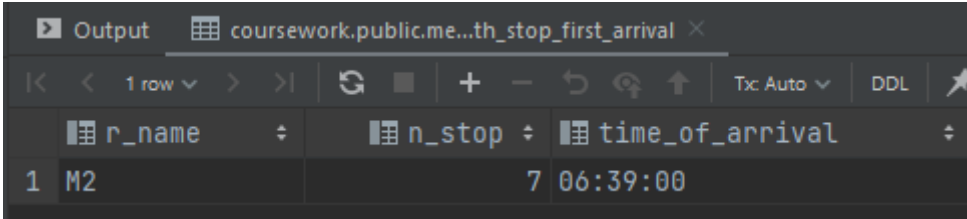
```
GROUP BY r_id, n_stop;
```

```
SELECT r_name, n_stop, time_of_arrival
```

```
FROM metro_nth_stop_first_arrival
```

```
WHERE r_id = 2
```

```
AND n_stop = 7;
```



| | r_name | n_stop | time_of_arrival |
|---|--------|--------|-----------------|
| 1 | M2 | 7 | 06:39:00 |

Рисунок 6.16 – Результат виконання запиту

14. Запит для виведення списку ліній метро і інклюзивних зупинок на них

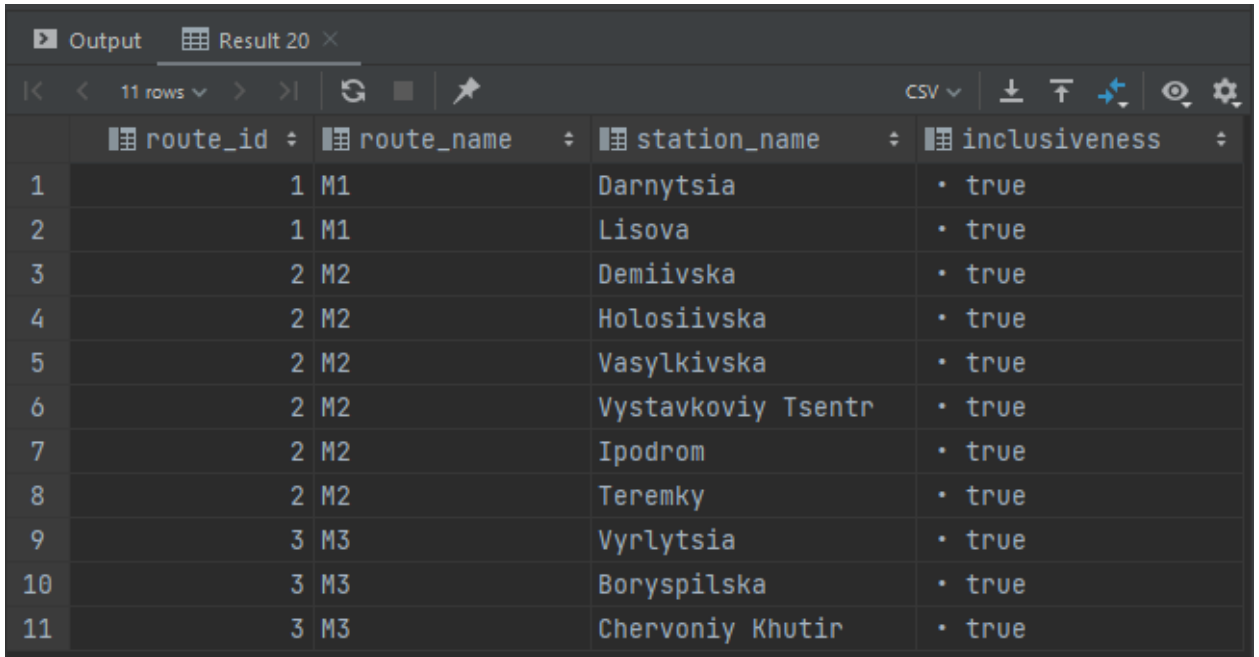
```
SELECT mrs.route_id, mr.route_name, ms.station_name, ms.inclusiveness
```

```
FROM metro_routes_stations mrs
```

```
JOIN metro_routes mr ON mr.route_id = mrs.route_id
```

```
JOIN metro_stations ms ON ms.station_id = mrs.stop_id
```

```
WHERE ms.inclusiveness = TRUE;
```

The screenshot shows a database query result with 11 rows. The columns are: route_id, route_name, station_name, and inclusiveness. The data is as follows:

| | route_id | route_name | station_name | inclusiveness |
|----|----------|------------|--------------------|---------------|
| 1 | 1 | M1 | Darnytsia | • true |
| 2 | 1 | M1 | Lisova | • true |
| 3 | 2 | M2 | Demiivska | • true |
| 4 | 2 | M2 | Holosiivska | • true |
| 5 | 2 | M2 | Vasylkivska | • true |
| 6 | 2 | M2 | Vystavkoviy Tsentr | • true |
| 7 | 2 | M2 | Ipodrom | • true |
| 8 | 2 | M2 | Teremky | • true |
| 9 | 3 | M3 | Vyrlytsia | • true |
| 10 | 3 | M3 | Boryspilska | • true |
| 11 | 3 | M3 | Chervoniy Khutir | • true |

Рисунок 6.17 – Результат виконання запиту

15. Представлення для виведення списку автобусних маршрутів, в яких можна потрапити з зупинки А на зупинку Б

```
CREATE VIEW bus_A_to_B
```

```
AS
```

```
SELECT br.route_id,
```

```
    br.route_name,
```

```
    brs1.stop_id stop_1,
```

```
    ags1.stop_name A_name,
```

```
    brs2.stop_id stop_2,
```

```
    ags2.stop_name B_name
```

```
FROM bus_routes br
```

```
    JOIN bus_routes_stops brs1 ON br.route_id = brs1.route_id
```

```
    JOIN bus_routes_stops brs2 ON br.route_id = brs2.route_id
```

```
    JOIN above_ground_stops ags1 ON ags1.stop_id = brs1.stop_id
```

```

JOIN above_ground_stops ags2 ON ags2.stop_id = brs2.stop_id

WHERE brs1.stop_order < brs2.stop_order;

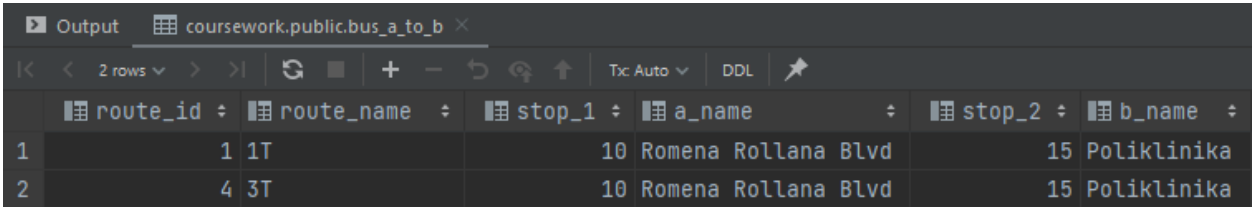
SELECT *

FROM bus_A_to_B

WHERE stop_1 = 10

AND stop_2 = 15;

```



The screenshot shows a database query output window with the title 'coursework.public.bus_a_to_b'. The output table has six columns: route_id, route_name, stop_1, a_name, stop_2, and b_name. There are two rows of data displayed.

| | route_id | route_name | stop_1 | a_name | stop_2 | b_name |
|---|----------|------------|--------|---------------------|--------|-------------|
| 1 | 1 | 1T | 10 | Romena Rollana Blvd | 15 | Poliklinika |
| 2 | 4 | 3T | 10 | Romena Rollana Blvd | 15 | Poliklinika |

Рисунок 6.18 – Результат виконання запиту

16. Представлення для виведення списку трамвайних маршрутів, в яких можна потрапити з зупинки А на зупинку Б

```
CREATE VIEW tram_a_to_b
```

```
AS
```

```
SELECT tr.route_id,
```

```
tr.route_name,
```

```
trs1.stop_id stop_1,
```

```
ags1.stop_name A_name,
```

```
trs2.stop_id stop_2,
```

```
ags2.stop_name B_name
```

```
FROM tram_routes tr
```

```
JOIN tram_routes_stops trs1 ON tr.route_id = trs1.route_id
```

```

JOIN tram_routes_stops trs2 ON tr.route_id = trs2.route_id

JOIN above_ground_stops ags1 ON ags1.stop_id = trs1.stop_id

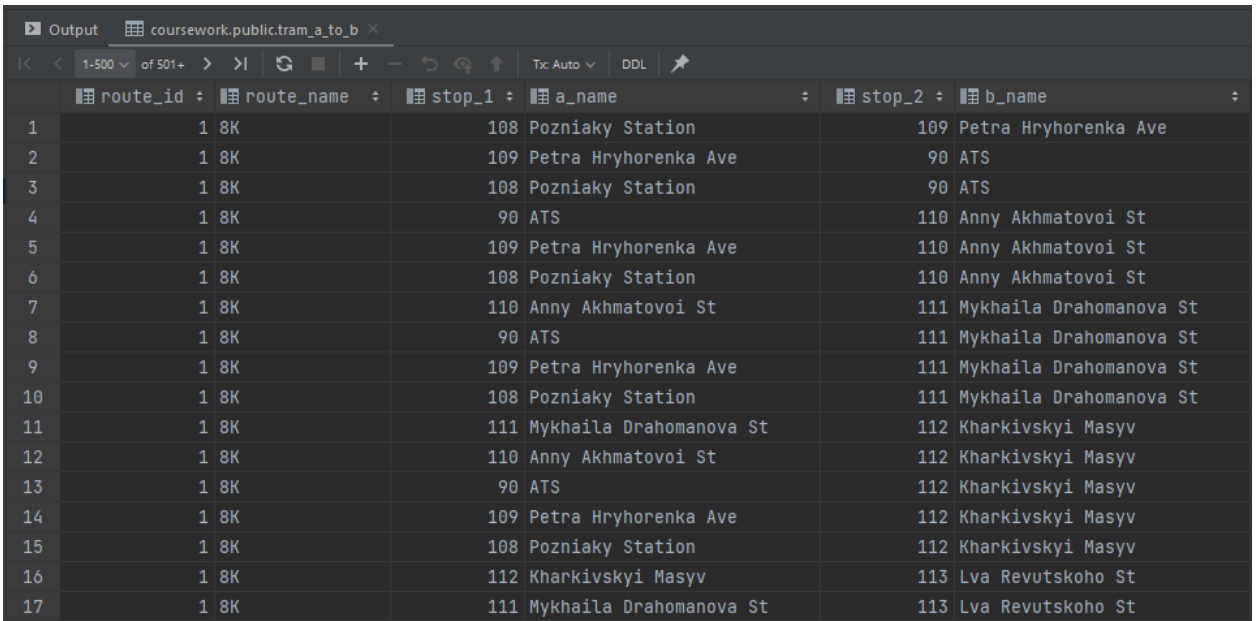
JOIN above_ground_stops ags2 ON ags2.stop_id = trs2.stop_id

WHERE trs1.stop_order < trs2.stop_order;

SELECT *

FROM tram_A_to_B;

```



| | route_id | route_name | stop_1 | a_name | stop_2 | b_name |
|----|----------|------------|--------|-------------------------|--------|-------------------------|
| 1 | 1 8K | | 108 | Pozniaky Station | 109 | Petra Hryhorenka Ave |
| 2 | 1 8K | | 109 | Petra Hryhorenka Ave | 90 | ATS |
| 3 | 1 8K | | 108 | Pozniaky Station | 90 | ATS |
| 4 | 1 8K | | 90 | ATS | 110 | Anny Akhmatovoi St |
| 5 | 1 8K | | 109 | Petra Hryhorenka Ave | 110 | Anny Akhmatovoi St |
| 6 | 1 8K | | 108 | Pozniaky Station | 110 | Anny Akhmatovoi St |
| 7 | 1 8K | | 110 | Anny Akhmatovoi St | 111 | Mykhaila Drahomanova St |
| 8 | 1 8K | | 90 | ATS | 111 | Mykhaila Drahomanova St |
| 9 | 1 8K | | 109 | Petra Hryhorenka Ave | 111 | Mykhaila Drahomanova St |
| 10 | 1 8K | | 108 | Pozniaky Station | 111 | Mykhaila Drahomanova St |
| 11 | 1 8K | | 111 | Mykhaila Drahomanova St | 112 | Kharkivskyi Masyv |
| 12 | 1 8K | | 110 | Anny Akhmatovoi St | 112 | Kharkivskyi Masyv |
| 13 | 1 8K | | 90 | ATS | 112 | Kharkivskyi Masyv |
| 14 | 1 8K | | 109 | Petra Hryhorenka Ave | 112 | Kharkivskyi Masyv |
| 15 | 1 8K | | 108 | Pozniaky Station | 112 | Kharkivskyi Masyv |
| 16 | 1 8K | | 112 | Kharkivskyi Masyv | 113 | Lva Revutskoho St |
| 17 | 1 8K | | 111 | Mykhaila Drahomanova St | 113 | Lva Revutskoho St |

Рисунок 6.19 – Результат виконання запиту

17. Запит для виведення списку автобусних і трамвайних маршрутів, в яких можна потрапити з зуп. А в зуп. Б

```

SELECT 'Автобус' transport, route_name, A_name, B_name

FROM bus_A_to_B

WHERE stop_1 = 16

AND stop_2 = 17

UNION

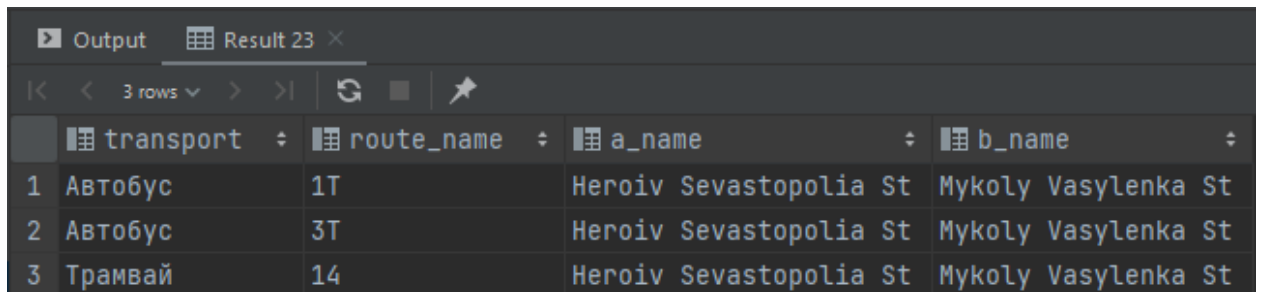
SELECT 'Трамвай', route_name, A_name, B_name

```

FROM tram_A_to_B

WHERE stop_1 = 16

AND stop_2 = 17;



The screenshot shows a database query result with 3 rows. The columns are transport, route_name, a_name, and b_name. The data is as follows:

| | transport | route_name | a_name | b_name |
|---|-----------|------------|------------------------|---------------------|
| 1 | Автобус | 1T | Heroiv Sevastopolia St | Mykoly Vasylenka St |
| 2 | Автобус | 3T | Heroiv Sevastopolia St | Mykoly Vasylenka St |
| 3 | Трамвай | 14 | Heroiv Sevastopolia St | Mykoly Vasylenka St |

Рисунок 6.20 – Результат виконання запиту

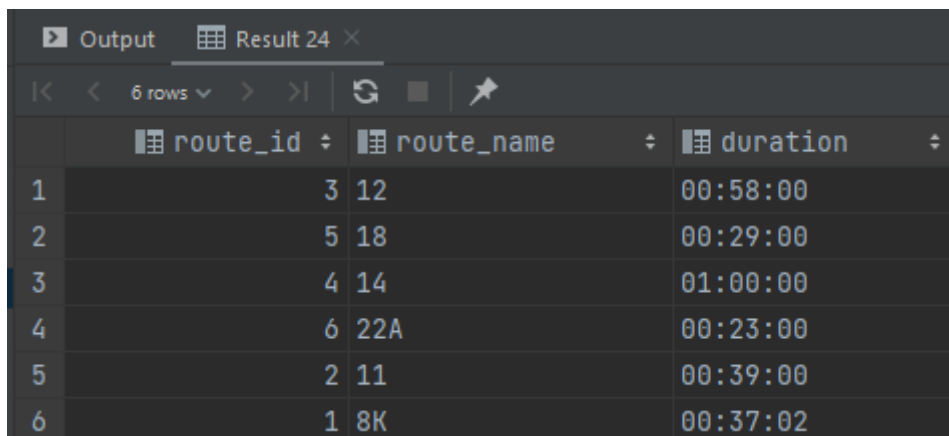
18. Запит для виведення трамвайних маршрутів та їх часової тривалості

SELECT trr.route_id, trr.route_name, CAST(SUM(trs.arrival_time) AS time)
duration

FROM tram_routes trr

JOIN tram_routes_stops trs ON trr.route_id = trs.route_id

GROUP BY trr.route_id, trr.route_name;



The screenshot shows a database query result with 6 rows. The columns are route_id, route_name, and duration. The data is as follows:

| | route_id | route_name | duration |
|---|----------|------------|----------|
| 1 | 3 | 12 | 00:58:00 |
| 2 | 5 | 18 | 00:29:00 |
| 3 | 4 | 14 | 01:00:00 |
| 4 | 6 | 22A | 00:23:00 |
| 5 | 2 | 11 | 00:39:00 |
| 6 | 1 | 8K | 00:37:02 |

Рисунок 6.21 – Результат виконання запиту

19. Представлення для виведення наземних зупинок та кількості автобусних маршрутів, що через них проходять

```

CREATE OR REPLACE VIEW bus_stop_count

AS

SELECT ags.stop_id s_id, ags.stop_name, COUNT(hrs.route_id) bus_routes

FROM above_ground_stops ags

        JOIN bus_routes_stops hrs ON ags.stop_id = hrs.stop_id

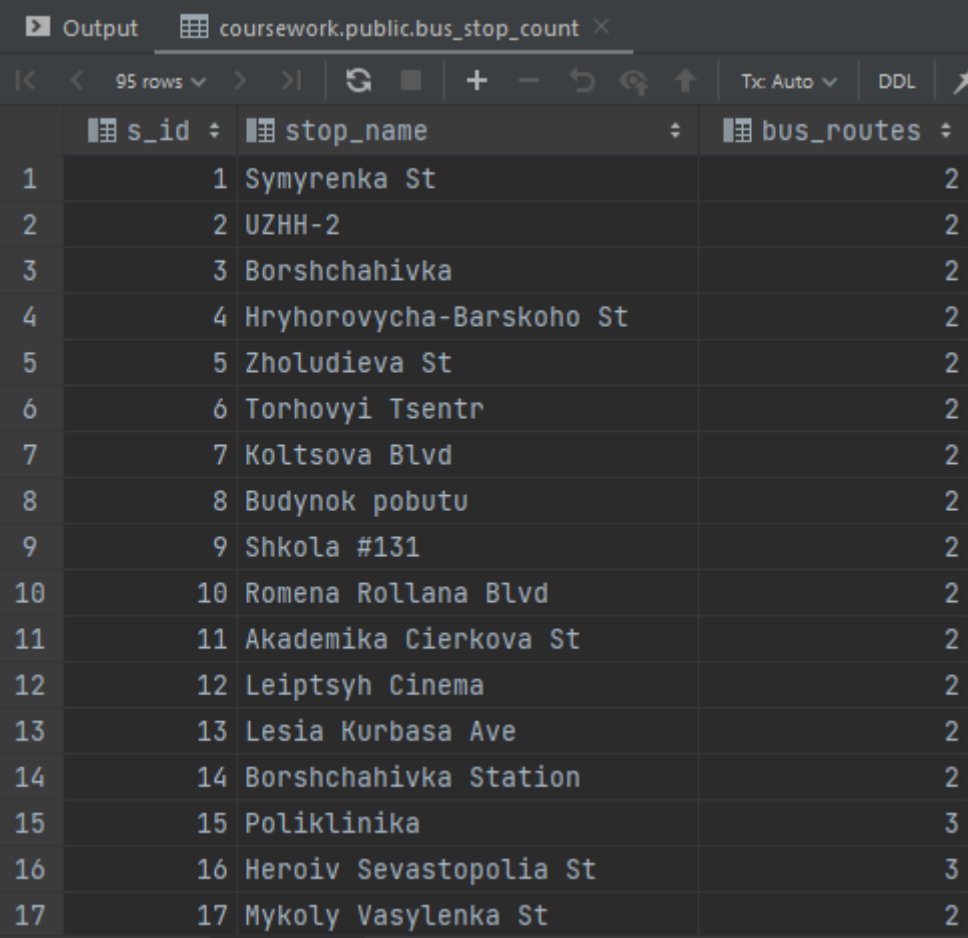
GROUP BY ags.stop_id, ags.stop_name

ORDER BY ags.stop_id;

SELECT *

FROM bus_stop_count;

```



| | s_id | stop_name | bus_routes |
|----|------|---------------------------|------------|
| 1 | 1 | Symyrenka St | 2 |
| 2 | 2 | UZHH-2 | 2 |
| 3 | 3 | Borshchahivka | 2 |
| 4 | 4 | Hryhorovycha-Baraskoho St | 2 |
| 5 | 5 | Zholudieva St | 2 |
| 6 | 6 | Torhovyi Tsentr | 2 |
| 7 | 7 | Koltsova Blvd | 2 |
| 8 | 8 | Budynok pobutu | 2 |
| 9 | 9 | Shkola #131 | 2 |
| 10 | 10 | Romena Rollana Blvd | 2 |
| 11 | 11 | Akademika Cierkova St | 2 |
| 12 | 12 | Leiptsyh Cinema | 2 |
| 13 | 13 | Lesia Kurbasa Ave | 2 |
| 14 | 14 | Borshchahivka Station | 2 |
| 15 | 15 | Poliklinika | 3 |
| 16 | 16 | Heroiv Sevastopolia St | 3 |
| 17 | 17 | Mykoly Vasylenska St | 2 |

Рисунок 6.22 – Результат виконання запиту

20. Представлення для виведення наземних зупинок та кількості трамвайних маршрутів, що через них проходять

```
CREATE OR REPLACE VIEW tram_stop_count
```

```
AS
```

```
SELECT ags.stop_id s_id, ags.stop_name, COUNT(trs.route_id) tram_routes
```

```
FROM above_ground_stops ags
```

```
JOIN tram_routes_stops trs ON ags.stop_id = trs.stop_id
```

```
GROUP BY ags.stop_id, ags.stop_name
```

```
ORDER BY ags.stop_id;
```

```
SELECT *
```

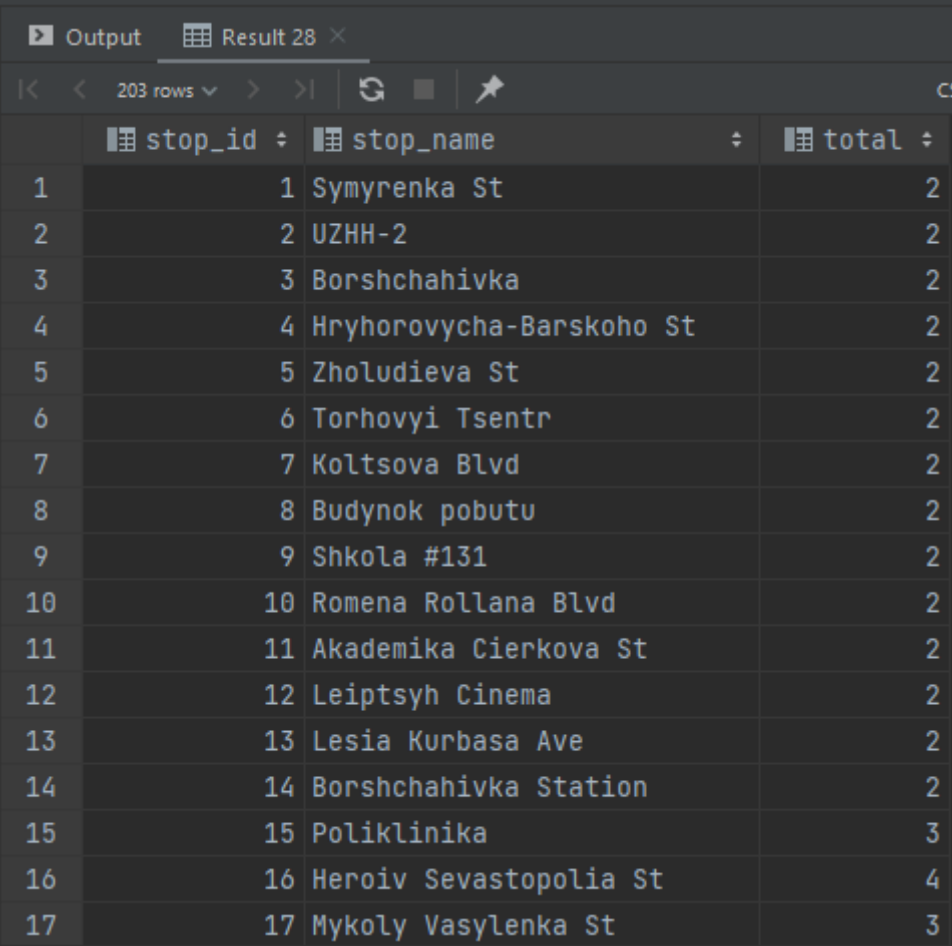
```
FROM tram_stop_count;
```

| Output coursework.public.tram_stop_count | | | |
|--|------|-------------------------|-------------|
| | s_id | stop_name | tram_routes |
| 1 | 16 | Heroiv Sevastopolia St | 1 |
| 2 | 17 | Mykoly Vasylenka St | 1 |
| 3 | 23 | Peremohy Square | 1 |
| 4 | 24 | Starovokzalna St | 1 |
| 5 | 31 | Vatslava Havela Blvd | 1 |
| 6 | 32 | Vidradnyi Ave | 1 |
| 7 | 75 | Pavla Usenka St | 1 |
| 8 | 77 | Mahnitohorska St | 1 |
| 9 | 78 | Chernihivska Station | 1 |
| 10 | 79 | Andriia Malyshka St | 1 |
| 11 | 80 | Oleksandra Boichenka St | 1 |
| 12 | 90 | ATS | 1 |
| 13 | 108 | Pozniaky Station | 1 |
| 14 | 109 | Petra Hryhorenka Ave | 1 |
| 15 | 110 | Anny Akhmatovoi St | 1 |
| 16 | 111 | Mykhaila Drahomanova St | 1 |
| 17 | 112 | Kharkivskyi Masyv | 1 |

Рисунок 6.23 – Результат виконання запиту

21. Запит для виведення наземних зупинок та кількості автобусних і трамвайних маршрутів, що через них проходять

```
SELECT ags.stop_id,  
  
       ags.stop_name,  
  
       (COALESCE((SELECT tram_stop_count.tram_routes FROM  
tram_stop_count WHERE tram_stop_count.s_id = ags.stop_id),  
  
                0) +  
  
        COALESCE((SELECT bus_stop_count.bus_routes FROM bus_stop_count  
WHERE bus_stop_count.s_id = ags.stop_id),  
  
                0)) total  
  
FROM above_ground_stops ags  
  
ORDER BY ags.stop_id;
```



| | stop_id | stop_name | total |
|----|---------|--------------------------|-------|
| 1 | 1 | Symyrenka St | 2 |
| 2 | 2 | UZHH-2 | 2 |
| 3 | 3 | Borshchahivka | 2 |
| 4 | 4 | Hryhorovycha-Barskoho St | 2 |
| 5 | 5 | Zholudieva St | 2 |
| 6 | 6 | Torhovyi Tsentr | 2 |
| 7 | 7 | Koltsova Blvd | 2 |
| 8 | 8 | Budynok pobutu | 2 |
| 9 | 9 | Shkola #131 | 2 |
| 10 | 10 | Romena Rollana Blvd | 2 |
| 11 | 11 | Akademika Cierkova St | 2 |
| 12 | 12 | Leiptsyh Cinema | 2 |
| 13 | 13 | Lesia Kurbasa Ave | 2 |
| 14 | 14 | Borshchahivka Station | 2 |
| 15 | 15 | Poliklinika | 3 |
| 16 | 16 | Heroiv Sevastopolia St | 4 |
| 17 | 17 | Mykoly Vasylenska St | 3 |

Рисунок 6.24 – Результат виконання запиту

в) Створення процедур та функцій

1. Функція для виведення таблиці з графіком прибуття вказаного автобусного маршруту на вказану зупинку

```
CREATE OR REPLACE FUNCTION display_bus_arrivals(IN br_id integer, IN
st_id integer)
```

```
RETURNS TABLE
```

```
(
    arrival time
)
```

```
AS
```


\$\$

DECLARE

st_ord smallint;

intr interval;

arr_time time;

w_end time;

time_of_week smallint;

BEGIN

IF NOT EXISTS(SELECT stop_order FROM bus_routes_stops WHERE
route_id = br_id AND stop_id = st_id) THEN

RAISE NOTICE 'Input data is incorrect!';

END IF;

IF EXTRACT(DOW FROM CURRENT_DATE) BETWEEN 0 AND 4 THEN

SELECT 1 INTO time_of_week;

ELSE

SELECT 2 INTO time_of_week;

END IF;

DROP TABLE IF EXISTS bus_arrivals;

CREATE TEMP TABLE bus_arrivals

(

arrival time

);

```
SELECT stop_order FROM bus_routes_stops WHERE route_id = br_id AND
stop_id = st_id INTO st_ord;
```

```
SELECT time_of_arrival::time FROM bus_nth_stop_first_arrival WHERE r_id
= br_id AND n_stop = st_ord INTO arr_time;
```

```
SELECT bus_routes.work_end FROM bus_routes WHERE route_id = br_id
INTO w_end;
```

```
SELECT bus_schedule.interval::interval FROM bus_schedule WHERE route_id
= br_id AND day_id = time_of_week INTO intr;
```

```
WHILE arr_time < w_end
```

```
LOOP
```

```
INSERT INTO bus_arrivals(arrival) VALUES (arr_time);
```

```
arr_time := arr_time + intr;
```

```
END LOOP;
```

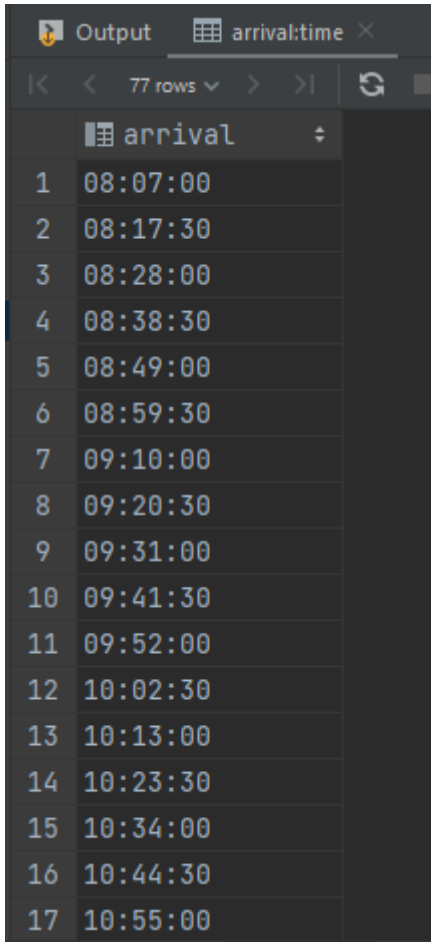
```
RETURN QUERY (SELECT * FROM bus_arrivals);
```

```
END;
```

```
$$ LANGUAGE plpgsql;
```

```
SELECT *
```

```
FROM display_bus_arrivals(1, 15);
```



The screenshot shows a database output window titled 'Output' with a tab for 'arrival:time'. It displays a table with 17 rows of arrival times. The table has a header row with the column name 'arrival' and a list of 17 rows of times starting from 08:07:00 and ending at 10:55:00.

| | arrival |
|----|----------|
| 1 | 08:07:00 |
| 2 | 08:17:30 |
| 3 | 08:28:00 |
| 4 | 08:38:30 |
| 5 | 08:49:00 |
| 6 | 08:59:30 |
| 7 | 09:10:00 |
| 8 | 09:20:30 |
| 9 | 09:31:00 |
| 10 | 09:41:30 |
| 11 | 09:52:00 |
| 12 | 10:02:30 |
| 13 | 10:13:00 |
| 14 | 10:23:30 |
| 15 | 10:34:00 |
| 16 | 10:44:30 |
| 17 | 10:55:00 |

Рисунок 6.25 – Результат виконання функції

2. Функція для виведення таблиці з графіком прибуття на вказану станцію певної лінії метро

CREATE OR REPLACE FUNCTION display_metro_arrivals(IN mr_id integer,
IN st_id integer)

RETURNS TABLE

(
arrival time
)

AS

\$\$

DECLARE

```

st_ord    smallint;

intr      interval;

intr_end  time;

w_end     time;

arr_time  time;

time_of_week smallint;

cur CURSOR FOR SELECT ms.interval_end, ms.interval

                FROM metro_schedule ms

                WHERE route_id = mr_id

                AND day_id = time_of_week;

BEGIN

    IF NOT EXISTS(SELECT stop_order FROM metro_routes_stations WHERE
route_id = mr_id AND stop_id = st_id) THEN

        RAISE NOTICE 'Input data is incorrect!';

    END IF;

    IF EXTRACT(DOW FROM CURRENT_DATE) BETWEEN 0 AND 4 THEN

        SELECT 1 INTO time_of_week;

    ELSE

        SELECT 2 INTO time_of_week;

    END IF;

    DROP TABLE IF EXISTS metro_arrivals;

```

```
CREATE TEMP TABLE metro_arrivals
```

```
(
    arrival time
);
```

```
SELECT stop_order FROM metro_routes_stations WHERE route_id = mr_id
AND stop_id = st_id INTO st_ord;
```

```
SELECT time_of_arrival::time FROM metro_nth_stop_first_arrival WHERE
r_id = mr_id AND n_stop = st_ord INTO arr_time;
```

```
SELECT metro_routes.work_end FROM metro_routes WHERE route_id =
mr_id INTO w_end;
```

```
OPEN cur;
```

```
FETCH cur INTO intr_end, intr;
```

```
WHILE intr_end < w_end
```

```
LOOP
```

```
    WHILE arr_time <= intr_end
```

```
        LOOP
```

```
            INSERT INTO metro_arrivals(arrival) VALUES (arr_time);
```

```
            arr_time := arr_time + intr;
```

```
        END LOOP;
```

```
    FETCH cur INTO intr_end, intr;
```

```
END LOOP;
```

```
CLOSE cur;
```

```

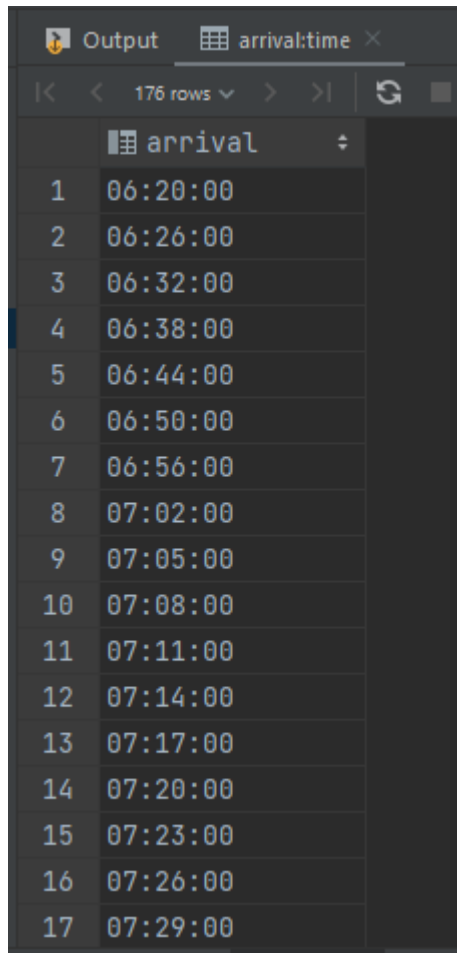
RETURN QUERY (SELECT * FROM metro_arrivals);

END;

$$ LANGUAGE plpgsql;

SELECT * FROM display_metro_arrivals(1, 4);

```



The screenshot shows a database output window titled 'Output' with a tab 'arrival:time'. It displays 176 rows. The first 17 rows are visible, showing a sequence of arrival times from 06:20:00 to 07:29:00. The table has two columns: an index (1-17) and the arrival time.

| | arrival |
|----|----------|
| 1 | 06:20:00 |
| 2 | 06:26:00 |
| 3 | 06:32:00 |
| 4 | 06:38:00 |
| 5 | 06:44:00 |
| 6 | 06:50:00 |
| 7 | 06:56:00 |
| 8 | 07:02:00 |
| 9 | 07:05:00 |
| 10 | 07:08:00 |
| 11 | 07:11:00 |
| 12 | 07:14:00 |
| 13 | 07:17:00 |
| 14 | 07:20:00 |
| 15 | 07:23:00 |
| 16 | 07:26:00 |
| 17 | 07:29:00 |

Рисунок 6.26 – Результат виконання запиту

г) Створення тригерів

1. Тригер для оновлення порядку зупинок у автобусному маршруті при вставленні нової зупинки

```

CREATE OR REPLACE FUNCTION func_on_insert() RETURNS trigger
AS

```

\$\$

BEGIN

UPDATE bus_routes_stops

SET stop_order = stop_order + 1

WHERE route_id = NEW.route_id

AND stop_order >= NEW.stop_order;

RAISE NOTICE 'Inserted new stop into bus route: id = %. Please, update the arrival_time value of the next stop.', NEW.route_id;

RETURN NEW;

END;

\$\$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER shift_order_on_insert

BEFORE INSERT

ON bus_routes_stops

FOR EACH ROW

EXECUTE FUNCTION func_on_insert();

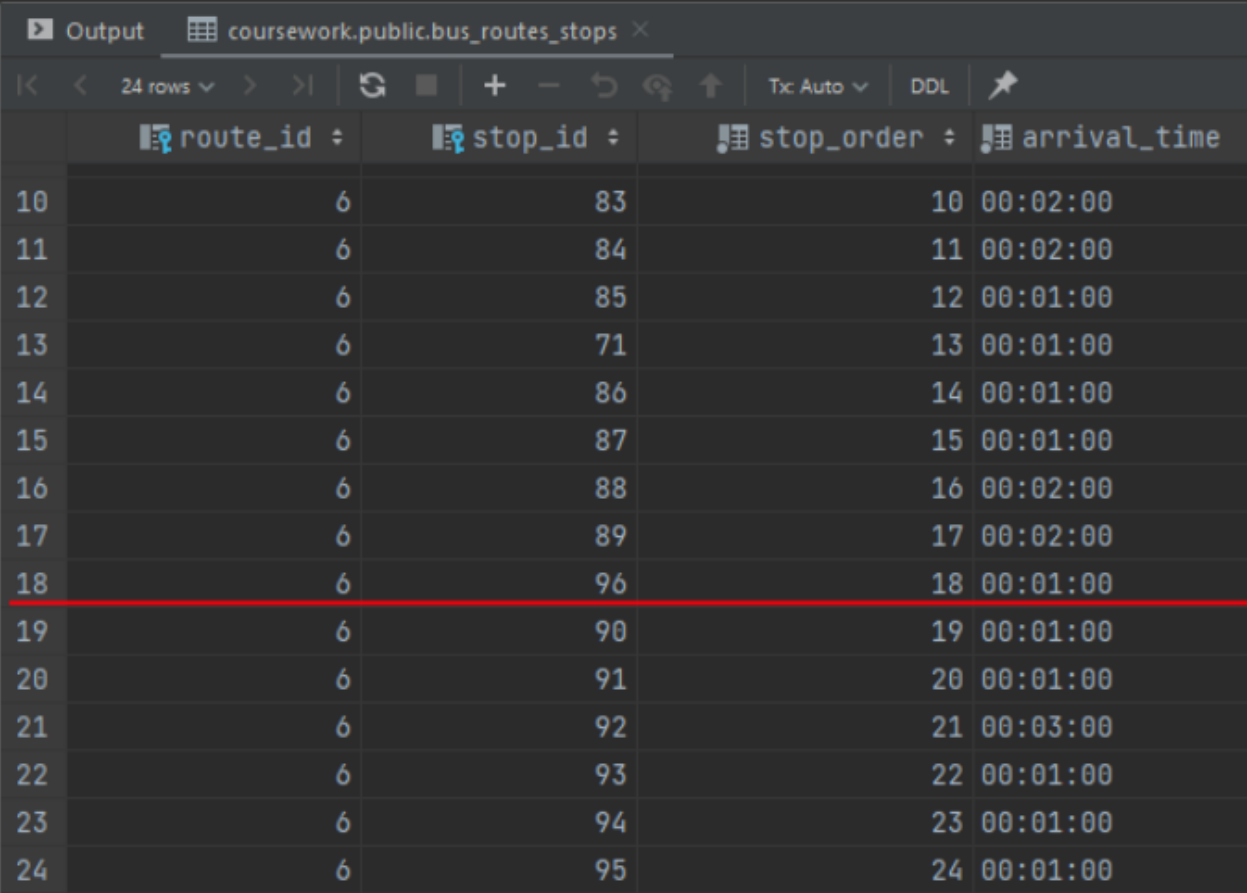
За допомогою команди INSERT вставимо до маршруту з ID=6 зупинку з ID=96 на 18у позицію у маршрут:

INSERT INTO bus_routes_stops

VALUES (6, 96, 18, '00:01:00');

```
coursework.public> INSERT INTO bus_routes_stops
VALUES (6, 96, 18, '00:01:00')
Inserted new stop into bus route: id = 6. Please, update the arrival_time value of the next stop.
[2023-01-09 21:57:20] 1 row affected in 20 ms
```

Рисунок 6.27 – Результат виконання роботи триггеру



| | route_id | stop_id | stop_order | arrival_time |
|----|----------|---------|------------|--------------|
| 10 | 6 | 83 | 10 | 00:02:00 |
| 11 | 6 | 84 | 11 | 00:02:00 |
| 12 | 6 | 85 | 12 | 00:01:00 |
| 13 | 6 | 71 | 13 | 00:01:00 |
| 14 | 6 | 86 | 14 | 00:01:00 |
| 15 | 6 | 87 | 15 | 00:01:00 |
| 16 | 6 | 88 | 16 | 00:02:00 |
| 17 | 6 | 89 | 17 | 00:02:00 |
| 18 | 6 | 96 | 18 | 00:01:00 |
| 19 | 6 | 90 | 19 | 00:01:00 |
| 20 | 6 | 91 | 20 | 00:01:00 |
| 21 | 6 | 92 | 21 | 00:03:00 |
| 22 | 6 | 93 | 22 | 00:01:00 |
| 23 | 6 | 94 | 23 | 00:01:00 |
| 24 | 6 | 95 | 24 | 00:01:00 |

Рисунок 6.28 – Результат виконання роботи триггеру

2. Тригер для оновлення порядку зупинок у автобусному маршруті при видаленні зупинки з нього

REATE OR REPLACE FUNCTION func_on_delete() RETURNS trigger

AS

\$\$

BEGIN

UPDATE bus_routes_stops

SET stop_order = stop_order - 1

WHERE route_id = OLD.route_id

AND stop_order > OLD.stop_order;


```
RAISE NOTICE 'Deleted a stop id: = % from bus route: id = %. Please, update
the arrival_time value of the next stop.', OLD.stop_id, OLD.route_id;
```

```
RETURN OLD;
```

```
END;
```

```
$$ LANGUAGE plpgsql;
```

```
CREATE OR REPLACE TRIGGER shift_order_on_delete
```

```
BEFORE DELETE
```

```
ON bus_routes_stops
```

```
FOR EACH ROW
```

```
EXECUTE FUNCTION func_on_delete();
```

Тепер за допомогою команди DELETE видалимо раніше уведену зупинку у 6й маршрут і поглянемо на результат:

```
DELETE FROM bus_routes_stops
```

```
WHERE route_id = 6 AND stop_order = 18;
```

```
coursework.public> DELETE FROM bus_routes_stops
                     WHERE route_id = 6 AND stop_order = 18
Deleted a stop id: = 96 from bus route: id = 6. Please, update the arrival_time value
[2023-01-09 22:04:18] 1 row affected in 6 ms
```

Рисунок 6.29 – Результат виконання запиту

| | route_id | stop_id | stop_order | arrival_time |
|----|----------|---------|------------|--------------|
| 9 | 6 | 82 | 9 | 00:01:00 |
| 10 | 6 | 83 | 10 | 00:02:00 |
| 11 | 6 | 84 | 11 | 00:02:00 |
| 12 | 6 | 85 | 12 | 00:01:00 |
| 13 | 6 | 71 | 13 | 00:01:00 |
| 14 | 6 | 86 | 14 | 00:01:00 |
| 15 | 6 | 87 | 15 | 00:01:00 |
| 16 | 6 | 88 | 16 | 00:02:00 |
| 17 | 6 | 89 | 17 | 00:02:00 |
| 18 | 6 | 90 | 18 | 00:01:00 |
| 19 | 6 | 91 | 19 | 00:01:00 |
| 20 | 6 | 92 | 20 | 00:03:00 |
| 21 | 6 | 93 | 21 | 00:01:00 |
| 22 | 6 | 94 | 22 | 00:01:00 |
| 23 | 6 | 95 | 23 | 00:01:00 |

Рисунок 6.30 – Результат виконання запиту

д) Оптимізація запитів

Для оптимізації запитів, скористаємося індексами:

Спочатку командою нижче перевіримо вартість і швидкість виконання запиту з таблиці metro_schedule

```
EXPLAIN ANALYSE SELECT * FROM metro_schedule WHERE day_id = 1
AND route_id = 1 AND interval_start BETWEEN '07:00:00' AND '19:00:00';
```

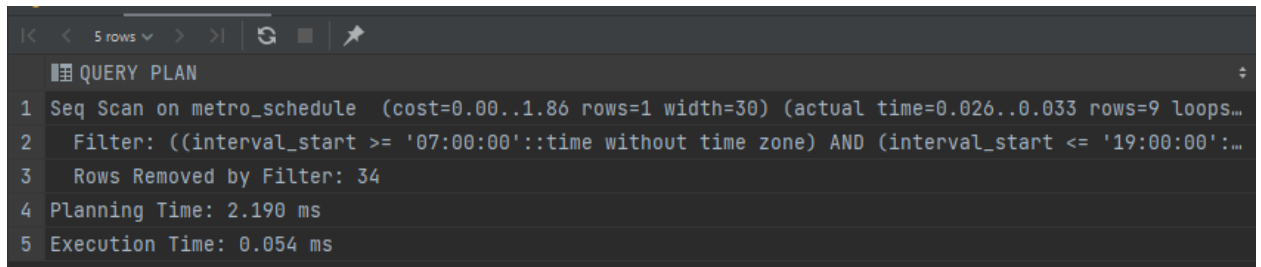
| Step | Operation | Cost | Rows | Width | Actual Time | Actual Rows | Loops |
|------|---|-------------|------|-------|--------------|-------------|-------|
| 1 | Seq Scan on metro_schedule | 0.00..38.00 | 1 | 30 | 0.054..0.066 | 9 | 1 |
| 2 | Filter: ((interval_start >= '07:00:00'::time without time zone) AND (interval_start <= '19:00:00'::tim... | | | | | | |
| 3 | Rows Removed by Filter: | | 34 | | | | |
| 4 | Planning Time: | 0.149 ms | | | | | |
| 5 | Execution Time: | 0.089 ms | | | | | |

Рисунок 6.31 – Статистика виконання запиту до застосування індексу

Тепер застосуємо індекс для колонки row_id:

```
CREATE INDEX ms_index ON metro_schedule(row_id);
```

І знову проглянемо статистику того ж запиту:



The screenshot shows a database query plan interface. At the top, there are navigation icons and a dropdown menu set to '5 rows'. Below this, the title 'QUERY PLAN' is displayed. The plan itself consists of five numbered steps:

| Step | Operation | Cost | Rows | Width | Actual Time | Actual Rows | Loops |
|------|---|------------|------|-------|--------------|-------------|----------|
| 1 | Seq Scan on metro_schedule | 0.00..1.86 | 1 | 30 | 0.026..0.033 | 9 | loops... |
| 2 | Filter: ((interval_start >= '07:00:00'::time without time zone) AND (interval_start <= '19:00:00'::time without time zone)) | | | | | | |
| 3 | Rows Removed by Filter: | | 34 | | | | |
| 4 | Planning Time: | 2.190 ms | | | | | |
| 5 | Execution Time: | 0.054 ms | | | | | |

Рисунок 6.32 – Статистика виконання запиту після застосування індексу

Можемо побачити, що вартість виконання запиту зменшилась як мінімум у 20 разів, а час виконання – приблизно у 2 рази. Такі показники дадуть значущий приріст у роботі з такими об'ємами даних.

ВИСНОВОК

В процесі виконання курсової роботи було розроблено базу даних для системи маршрутів громадського транспорту міста. Було проведено аналіз предметного середовища, спроектовано ER-модель бази даних та визначено потреби майбутньої бази даних. Визначено бізнес-правила, слідування яким необхідне для правильного функціонування БД. Також проаналізували потреби різних користувачів та окреслили можливості, які необхідні кожному типу користувачів у БД. За цією моделлю реалізовано базу даних під управлінням СКБД PostgreSQL: створено БД, таблиці, встановлено відношення між таблицями, створено запити, що дають змістовну інформацію з бази даних, а також реалізовано весь необхідний у роботі функціонал. Проведено дослідження з правильної роботи запитів, функцій, тригерів. Також проведено окреме дослідження з можливостей оптимізації запитів в реляційних базах даних. Результати даної оптимізації більш ніж задовільні.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. <https://www.eway.in.ua/ua/cities/kyiv/routes>
2. <https://www.postgresql.org/docs/15/user-manag.html>
3. <https://www.postgresql.org/docs/15/functions.html>
4. <https://www.postgresql.org/docs/current/plpgsql-trigger.html>
5. <https://www.postgresql.org/docs/15/indexes.html>