Міністерство освіти і науки України
Національний університет «Одеська політехніка»

Інститут комп’ютерних систем
Кафедра інформаційних систем

Гуржій Максим Артурович
студент групи АІ-221

КУРСОВА РОБОТА

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

Спеціальність:
122 Комп’ютерні науки

Освітня програма: Комп’ютерні науки

Керівник:
Глава Мария Геннадьївна
Доцент кафедри. ступінь к.т.н.

Одеса – 2023

ЗМІСТ

Інститут комп’ютерних систем
Кафедра інформаційних систем

ЗАВДАННЯ
НА КУРСОВУ РОБОТУ

Студенту: Гуржію Максиму Артуровичу
Група: АІ-221

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

2. Термін здачі студентом закінченої роботи 01.12.2023

3. Початкові дані до проекту (роботи)_Варіант 10: Завдання: Розробити базу даних маршрутів громадського транспорту з можливістю зберігання топологічної схеми маршрутів 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. Функціонал: Виведення даних про найкоротший/найдовший маршрут. Виведення маршрутів, які мають у своїй частині однакові сегменти вулиць.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які належить розробити): Розробити базу даних маршрутів громадського транспорту з можливістю зберігання топологічної схеми маршрутів

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень): Інформаційна модель БД, скриншоти заповнених таблиць.

Завдання видано 21.09.2023 _____
(підпис викладача)

Завдання прийнято до виконання 21.09.2023 _____
(підпис студента)

АНОТАЦІЯ

Була розроблена база даних маршрутів громадського транспорту. При цьому було визначено структуру сутностей, їх атрибути, типи даних і зв'язки між таблицями. Написаний програмний код для реалізації цієї бази даних на мові PLPGSQL, який включає в себе створення послідовностей, таблиць і наборів даних. Також були розроблені запити, представлення, тригери та користувацькі функції для виконання функціоналу бази даних та отримання вихідних даних для існуючих сутностей. Здійснено аналіз та створення облікових записів для потенційних користувачів бази даних, і надані відповідні права доступу.

ABSTRACT

A database of mass transport routes has been deployed. In this case, the contents of the elements, their attributes, data types and relationships between tables are indicated. Code has been written to implement these PLPGSQL-based databases, which includes creating sequences, tables and data sets. It would also be possible to use detailed descriptions provided, triggers and function usage to use functions based on data and other remote data for different days. Daily analysis and creation of shared records for powerful underlying data, and provided additional rights available.

ВСТУП

У цій роботі розробляється база даних для оптимізації управління громадським транспортом у місті. Мета цієї бази даних полягає в забезпеченні ефективного ведення інформації про маршрути громадського транспорту та їх топологічну структуру.

Створення такої бази даних спрощує ведення інформації про маршрути та забезпечує зручний доступ до необхідної інформації. Зростання актуальності створення таких баз даних відбувається у зв'язку з постійним розвитком міських інфраструктур та підвищенням вимог до ефективності громадського транспорту.

Завдання бази даних для громадського транспорту виступає не тільки інструментом для зберігання та обробки даних, але й сприяє оптимізації управління маршрутами та полегшує взаємодію з користувачами громадського транспорту.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

Задача курсової роботи - Розробити базу даних для оптимізації управління маршрутами громадського транспорту з врахуванням топологічної структури міста.

1.1 Опис предметної області: Маршрути громадського транспорту

Об’єктом дослідження є сфера організації та управління маршрутами громадського транспорту у місті. Ця предметна область є ключовою для забезпечення ефективного та комфортного транспортного обслуговування мешканців та відвідувачів міста.

У даній предметній області стоять перед конкретними завданнями та викликами, такими як оптимізація маршрутів, визначення ефективного розподілу ресурсів та виявлення дублюючих транспортних маршрутів. Розробка бази даних для системи управління маршрутами громадського транспорту має на меті ефективно вирішити ці завдання та полегшити організацію та контроль над транспортним рухом у місті.

Така система повинна включати інформацію про маршрути (тип, номер, назва), сегменти маршрутів (маршрут, код сегменту маршруту, сегмент вулиці), вулиці міста (код, назва) та сегменти вулиць (вулиця, код сегменту, опис місцеположення та граничні номери будинків).

1.2 Опис користувачів системи та їх історії (UserStory)

User story:

Адміністратор БД:

1.Як адміністратор бази даних, я хочу мати можливість додавати нові маршрути з обов'язковим заповненням типу, номеру та назви маршруту.

2.Як адміністратор бази даних, я хочу мати можливість додавати нові вулиці міста з обов'язковим заповненням коду та назви.

3.Як адміністратор бази даних, я хочу мати можливість додавати сегменти маршрутів та сегменти вулиць для побудови топологічної схеми маршрутів.

4.Як адміністратор бази даних, я хочу мати можливість редагувати інформацію про існуючі маршрути та вулиці.

5.Як адміністратор бази даних, я хочу мати можливість видаляти маршрути та вулиці, які більше не використовуються.

Користувач Пошуку Маршруту та Планування Подорожі

1.Як користувач, я хочу мати можливість знайти маршрути, які проходять через задану вулицю, щоб знайти зручні способи переміщення.

2.Як користувач, я хочу мати можливість знайти маршрути, які проходять від точки А до точки Б, щоб планувати свої подорожі.

3.Як користувач, я хочу мати можливість вивести дані про найкоротший та найдовший маршрут між заданими точками, щоб планувати свої подорожі більш ефективно.

4.Як користувач, я хочу мати можливість знаходити маршрути різних видів транспорту, які мають у своїй частині однакові сегменти вулиць, щоб уникати дублювання подорожей.

5.Як користувач, я хочу мати можливість отримувати інформацію про час руху громадського транспорту на вибраних маршрутах та вулицях для планування часу своєї подорожі.

1.3 Детальний опис функціоналу бази даних, що пропонується

Завдання:

1. Реалізувати код для створення таблиць та послідовностей, визначити зв’язки між ними.
2. Додати приклади наборів даних для всіх атрибутів, забезпечуючи реалістичні тестові дані.
3. Створити запити до реалізованих сутностей, використовуючи різні функції, оператори та підзапити.
4. Розробити представлення, які виводять вихідні дані, демонструючи корисність та функціональність бази даних.
5. Додати тригери, що перевіряють правильність вводу, зміни та видалення даних.
6. Розробити користувацький функціонал для зручного введення та редагування списків клієнтів, замовлень та видань.

Функціонал бази даних для маршрутів громадського транспорту:
Введення та редагування інформації:

1.Додавання та редагування маршрутів:
Можливість додавати нові маршрути з вказанням типу, номеру та назви.
Редагування інформації про існуючі маршрути.

2.Додавання та редагування сегментів маршрутів:
Можливість додавати та редагувати сегменти маршрутів, вказуючи маршрут, код сегменту маршруту та сегмент вулиці.

3.Додавання та редагування вулиць міста:
Додавання нових вулиць з вказанням коду та назви.
Редагування інформації про існуючі вулиці.

4. Додавання та редагування сегментів вулиць:

Додавання та редагування сегментів вулиць з вказанням вулиці, коду сегменту, опису місцеположення та граничних номерів будинків.

5.Побудова маршруту з точки А в точку Б:
Реалізація можливості планування маршрутів враховуючи точки відправлення та призначення.

6.Виявлення дублюючих маршрутів:
Система надає можливість виявлення дубльованих маршрутів різними видами транспорту.

7.Виведення маршрутів з однаковими сегментами вулиць:

Розробка функції, яка виводить маршрути, які мають у своїй частині однакові сегменти вулиць.

2 ПРОЕКТУВАННЯ БАЗИ ДАНИХ Маршрутів громадського транспорту

Маршрути :

Дане відношення знаходиться в першій нормальній формі, так як значення кожного атрибуту не розділяється на декілька значень.

Дане відношення знаходиться у другій нормальній формі, так як кожен неключовий атрибут функціонально повно залежить від первинного ключа — ID маршруту.

Дане відношення знаходиться у третій нормальній формі, так як кожен неключовий атрибут залежить тільки від первинного ключа ID маршруту та не виникає інформаційної надмірності та аномалій.

Дане відношення знаходиться в нормальній формі Бойса-Кодда, так як в ньому відсутні функціональні залежності атрибутів складеного ключа від неключових атрибутів. Ця умова виконується за замовчуванням, так як в даному відношенні ключ не являється складеним.

Маршрутні дублікати :

Дане відношення знаходиться в першій нормальній формі, так як значення кожного атрибуту не розділяється на декілька значень.

Дане відношення знаходиться у другій нормальній формі, так як кожен неключовий атрибут функціонально повно залежить від первинного ключа — ID маршрутного дублікату.

Дане відношення знаходиться у третій нормальній формі, так як кожен неключовий атрибут залежить тільки від первинного ключа ID маршрутного дублікату та не виникає інформаційної надмірності та аномалій.

Дане відношення знаходиться в нормальній формі Бойса-Кодда, так як в ньому відсутні функціональні залежності атрибутів складеного ключа від

неключових атрибутів. Ця умова виконується за замовчуванням, так як в даному відношенні ключ не являється складеним.

Сегменти маршрутів :

Дане відношення знаходиться в першій нормальній формі, так як значення кожного атрибуту не розділяється на декілька значень.

Дане відношення знаходиться у другій нормальній формі, так як кожен неключовий атрибут функціонально повно залежить від первинного ключа — ID сегменту маршруту .

Дане відношення знаходиться у третій нормальній формі, так як кожен неключовий атрибут залежить тільки від первинного ключа ID сегменту маршруту та не виникає інформаційної надмірності та аномалій.

Дане відношення знаходиться в нормальній формі Бойса-Кодда, так як в ньому відсутні функціональні залежності атрибутів складеного ключа від неключових атрибутів. Ця умова виконується за замовчуванням, так як в даному відношенні ключ не являється складеним.

Сегменти вулиць :

Дане відношення знаходиться в першій нормальній формі, так як значення кожного атрибуту не розділяється на декілька значень.

Дане відношення знаходиться у другій нормальній формі, так як кожен неключовий атрибут функціонально повно залежить від первинного ключа — ID Сегменту вулиці .

Дане відношення знаходиться у третій нормальній формі, так як кожен неключовий атрибут залежить тільки від первинного ключа ID Сегменту вулиці та не виникає інформаційної надмірності та аномалій.

Дане відношення знаходиться в нормальній формі Бойса-Кодда, так як в ньому відсутні функціональні залежності атрибутів складеного ключа від неключових атрибутів. Ця умова виконується за замовчуванням, так як в даному відношенні ключ не являється складеним.

Вулиці міста :

Дане відношення знаходиться в першій нормальній формі, так як значення кожного атрибуту не розділяється на декілька значень.

Дане відношення знаходиться у другій нормальній формі, так як кожен неключовий атрибут функціонально повно залежить від первинного ключа — ID Вулиці міста .

Дане відношення знаходиться у третій нормальній формі, так як кожен неключовий атрибут залежить тільки від первинного ключа ID Вулиці міста та не виникає інформаційної надмірності та аномалій.

Дане відношення знаходиться в нормальній формі Бойса-Кодда, так як в ньому відсутні функціональні залежності атрибутів складеного ключа від неключових атрибутів. Ця умова виконується за замовчуванням, так як в даному відношенні ключ не являється складеним.

Автомитизація:

1.Розрахунок загальної вартості поїздки: Можна створити збережену процедуру, яка обчислюватиме загальну вартість поїздки на основі даних з таблиць "trips", "tripsegment" і "segments".

2.Створення дублікатів поїздок: Розгляньте можливість створення процедури, яка автоматично створюватиме дублікати поїздок у таблиці "tripduplicate" для певних типів поїздок з таблиці "trips".

3.Зміна послідовності сегментів на поїздці: Можна створити процедуру для зміни порядку сегментів на поїздці, аналогічну тій, яку ви вже описували.

4.Видалення застарілих записів: Розгляньте можливість створення регулярного завдання (наприклад, розгортку) для видалення застарілих записів в таблицях, щоб зберігати базу даних в актуальному стані.

5.Розрахунок статистики по поїздках: Можна створити процедури для розрахунку різних статистичних показників по вашим поїздкам, таких як середня вартість поїздки, найпопулярніший тип поїздки тощо.

6.Створення звітів: Якщо ви потребуєте регулярно створювати звіти на основі даних в базі даних, можна розглянути автоматизацію цього процесу за допомогою збережених процедур та запитів.

7.Оновлення даних в режимі реального часу: Якщо вам потрібно забезпечити оновлення даних в режимі реального часу, ви можете розглянути можливість використання тригерів або інших механізмів для автоматичного оновлення інформації в таблицях при зміні відповідних даних.

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
Маршрути	Містить інформацію про маршрути
Сегменти маршрутів	Містить інформацію про сегменти маршрутів
Вулиці міста	Містить інформацію про вулиці міста
Сегменти вулиць	Містить інформацію про сегменти вулиць
Маршрутні дублікати	Містить інформацію про маршрутні дублікати

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

Таблиця 2.2

Сутність	Властивість	Тип даних	Ключ
Маршрути	ID маршруту	Цілочисельний	Первинний
	Тип маршруту	Символьний	

	Номер маршруту (наприклад, №5, А12 і т.д.).	Цілочисельний	
	Назва маршруту	Символьний	
Сегменти маршрутів	ID сегмента маршруту.	Цілочисельний	Первинний
	ID Маршруту	Цілочисельний	Зовнішний
	ID Сегменту вулиці	Цілочисельний	Зовнішний
Вулиці міста	ID вулиці	Цілочисельний	Первинний
	Назва	Символьний	
Сегменти вулиць	ID сегмента вулиці	Цілочисельний	Первинний
	ID Вулиці	Цілочисельний	Зовнішний
	Граничні номери будинків(початковий та кінцевий номери будинків на сегменті вулиці)	Цілочисельний	
Маршрутні дублікати	ID Маршрутного дублікату	Цілочисельний	Первинний
	ID Маршруту 1	Цілочисельний	Зовнішний
	ID Маршруту 2	Цілочисельний	Зовнішний
Послідовність сегментів на маршруті	ID Послідовності	Цілочисельний	Зовнішний
	ID сегменту маршруту	Цілочисельний	Зовнішний
	ID маршруту	Цілочисельний	Зовнішний

	Порядковий номер	Цілочисельний	Зовнішний
Сегменти	ID сегмента вулиці	Цілочисельний	Зовнішний
	IDсегменту маршруту	Цілочисельний	Зовнішний

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

Таблиця 2.3 – Типи зв’язків між сутностями

Сутність	Тип зв’язку	Сутність
Маршрути	N : 1	Маршрутні дублікати
Маршрути	1 : N	**Послідовність сегментів на маршруті**
Сегменти маршрутів	1 : N	**Послідовність сегментів на маршруті**
Сегменти маршрутів	1 : N	Сегменти
Сегменти вулиць	1 : N	Сегменти
Вулиці міста	1 : N	Сегменти вулиць

На основі виділених сутностей, їх атрибутів та зв’язків між сутностями створено інформаційну модель в нотации "вороняча лапка", яка представлена на рис. 2.1.

Рисунок 2.1 – Інформаційна модель в нотации "вороняча лапка"

3 ВИБІР ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Для реалізації бази даних Навантаження кафедри ВНЗ було обрано систему управління базами даних (СУБД) PostgreSQL. PostgreSQL є потужною, відкритою, об'єктно-реляційною СУБД, яка володіє низкою переваг, що робить її ідеальним вибором для нашого проекту.

Обгрунтування вибору PostgreSQL:

– Відкритий код:

Велика спільнота користувачів надає підтримку та активно оновлює систему.

– Переносимість:

PostgreSQL підтримує різні операційні системи (Windows, Linux, MacOS X), що гарантує гнучкість у виборі хостингу і серверного обладнання.

– Розширюваність:

Можливість використовувати різноманітні розширення дозволяє адаптувати систему під конкретні потреби проекту.

– Безпека:

PostgreSQL забезпечує розширені можливості забезпечення безпеки даних. Підтримка різних методів шифрування і аутентифікації забезпечує високий рівень захисту інформації.

Рекомендовані мінімальні характеристики для клієнтських робочих станцій: Операційна система: Windows 7/10, Linux, MacOS X.

Процесор: 1 ядро, 1.6 GHz або

краще. Оперативна пам'ять: 2 ГБ

або більше.

Місце на жорсткому диску: Мінімум 2 ГБ вільного місця.

Використання PostgreSQL для реалізації бази даних гарантує високу ефективність роботи, надійність та безпеку інформації.

4 СТВОРЕННЯ БАЗИ ДАНИХ

На підставі отриманого проекту засобами мови SQL створюються базові елементи бази даних: домени, таблиці і т.ін. і додаткові елементи бази даних: представлення, тригери і процедури (або призначені для користувача функції).

4.1 Створення таблиць

В першу чергу було створено послідовності для майбутніх таблиць.

```
CREATE SEQUENCE seq_street;
CREATE SEQUENCE seq_streetSegment;
CREATE SEQUENCE seq_Segments;
CREATE SEQUENCE seq_tripSegment;
CREATE SEQUENCE seq_sequenceOfSegmentsOnTrip;
CREATE SEQUENCE seq_trips;
CREATE SEQUENCE seq_tripDuplicates;
```

Далі було створено самі таблиці з набором їх властивостей та, за наявності, зовнішніми ключами.

```
CREATE TABLE trips (
Trip_id INT PRIMARY KEY DEFAULT NEXTVAL('seq_trips'),
TypeOfTrip VARCHAR,
NumberOfTrip INT ,
NameOfTrip VARCHAR
);

CREATE TABLE TripDublicates (
TripDublicate_id INT PRIMARY KEY DEFAULT
NEXTVAL('seq_tripDuplicates'),
```

```
Trip_id INT NOT NULL REFERENCES trips(trip_id)
);

CREATE TABLE street (
street_id INT PRIMARY KEY DEFAULT NEXTVAL('seq_street'),
NameOfStreet VARCHAR
);

CREATE TABLE streetSegment (
streetSegment_id INT PRIMARY KEY DEFAULT
NEXTVAL('seq_streetSegment'),
street_id INT NOT NULL REFERENCES street(street_id),
BorderHousesNumber VARCHAR
);

CREATE TABLE TripSegment (
TripSegment_id INT PRIMARY KEY DEFAULT
NEXTVAL('seq_tripSegment'),
Trip_id INT NOT NULL REFERENCES trips(trip_id),
streetSegment_id INT NOT NULL REFERENCES
streetSegment(streetSegment_id)
);

CREATE TABLE Segments (
streetSegment_id INT NOT NULL REFERENCES
streetSegment(streetSegment_id),
TripSegment_id INT NOT NULL REFERENCES TripSegment(TripSegment_id)
);

CREATE TABLE sequenceOfSegmentsOnTrip (
```

```
sequenceOfSegmentsOnTrip_id INT PRIMARY KEY DEFAULT
NEXTVAL('seq_sequenceOfSegmentsOnTrip'),
TripSegment_id INT NOT NULL REFERENCES
TripSegment(TripSegment_id),
Trip_id INT NOT NULL REFERENCES trips(trip_id),
SequenceNumber INT
);
```

Приведен код, за допомогою якого создадемо таблиці, обов’язково слідкуємо за первинними та зовнішними ключами щоб не вникало помилок.

Опис коду:

Таблиця trips:

Trip_id: Первинний ключ, автоматично генерується послідовністю seq_trips.

TypeOfTrip: Тип маршруту.

NumberOfTrip: Номер маршруту.

NameOfTrip: Назва маршруту.

Таблиця TripDuplicates:

TripDublicate_id: Первинний ключ, автоматично генерується послідовністю seq_tripDuplicates.

Trip_id: Зовнішній ключ, посилається на trip_id таблиці trips.

Таблиця street:

street_id: Первинний ключ, автоматично генерується послідовністю seq_street.

NameOfStreet: Назва вулиці.

Таблиця streetSegment:

streetSegment_id: Первинний ключ, автоматично генерується послідовністю seq_streetSegment.

street_id: Зовнішній ключ, посилається на street_id таблиці street.

borderHousesNumber: Інформація про граничні номери будинків на вулиці.

Таблиця TripSegment:

TripSegment_id: Первинний ключ, автоматично генерується послідовністю seq_tripSegment.

Trip_id: Зовнішній ключ, посилається на trip_id таблиці trips.

streetSegment_id: Зовнішній ключ, посилається на streetSegment_id таблиці streetSegment.

Таблиця Segments:

streetSegment_id: Зовнішній ключ, посилається на streetSegment_id таблиці streetSegment.

TripSegment_id: Зовнішній ключ, посилається на TripSegment_id таблиці TripSegment.

Таблиця sequenceOfSegmentsOnTrip:

sequenceOfSegmentsOnTrip_id: Первинний ключ, автоматично генерується послідовністю seq_sequenceOfSegmentsOnTrip.

TripSegment_id: Зовнішній ключ, посилається на TripSegment_id таблиці TripSegment.

Trip_id: Зовнішній ключ, посилається на trip_id таблиці trips.

SequenceNumber: Порядковий номер сегмента на маршруті.

4.2 Створення представлень

Створити VIEW для спрощення отримання інформації про поїздки та пов'язані з ними сегменти вулиць.

```
CREATE VIEW trip_info AS
SELECT
   t.trip_id,
   t.typeoftrip,
```

```
    t.numberoftrip,
    t.nameoftrip,
    ts.streetsegment_id
FROM
    trips t
JOIN
    tripsegment ts ON t.trip_id = ts.trip_id;
```

	trip_id integer	typeoftrip character varying	numberoftrip integer	nameoftrip character varying	streetsegment_id integer
1	3	Mountain Hike	303	Alpine Adventure	6
2	2	Beach Vacation	222	Sunny Shore Retreat	4

Рис 4.1

Зробимо VIEW для отримання інформації про сегменти вулиць та відповідні назви вулиць.

```
CREATE VIEW street_info AS
SELECT
    ss.street_id,
    ss.borderHousesNumber,
    s.nameofstreet
FROM
    streetsegment ss
JOIN
    street s ON ss.street_id = s.street_id;
```

	street_id integer	borderhousesnumber character varying	nameofstreet character varying
1	3	500-600	Oak Avenue
2	1	250-300	Pine Street

Рис 4.2

VIEW для отримання інформації про дублікати поїздок та відповідні поїздки з таблиці trips.

```
CREATE VIEW trip_duplicate_info AS
SELECT
    td.TripDublicate_id,
    td.Trip_id AS duplicate_trip_id,
    t.TypeOfTrip,
    t.NumberOfTrip,
    t.NameOfTrip
FROM
    TripDublicates td
JOIN
    trips t ON td.Trip_id = t.Trip_id;
```

	tripdublicate_id integer	duplicate_trip_id integer	typeoftrip character varying	numberoftrip integer	nameoftrip character varying
1	4	2	Beach Vacation	222	Sunny Shore Retreat
2	5	3	Mountain Hike	303	Alpine Adventure

Рис 4.3

4.3 Створення тригерів

Напишемо тригер який буде автоматично вставляти в таблицу streetSegment новий запис з таблиці street та буде автоматично вказувати номер граничних будинків 0-100.

```
CREATE OR REPLACE FUNCTION insert_street_segment()
RETURNS TRIGGER AS $$
BEGIN
    INSERT INTO streetSegment (street_id, BorderHousesNumber)
```

```
    VALUES (NEW.street_id, '0-100');

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER after_insert_street
AFTER INSERT
ON street
FOR EACH ROW
EXECUTE FUNCTION insert_street_segment();
```

2.Напишемо який викликається перед вставкою нового запису, ми перевіряємо, чи NameOfTrip порожній.

```
CREATE OR REPLACE FUNCTION set_name_of_trip()
RETURNS TRIGGER AS $$
BEGIN
    IF NEW.NameOfTrip IS NULL THEN
        NEW.NameOfTrip = 'New Trip';
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER before_insert_trip
BEFORE INSERT
ON trips
FOR EACH ROW
```

EXECUTE FUNCTION set_name_of_trip();

3.Якщо значення для NameOfStreet не вказане, воно буде встановлене Unknown street.

```
CREATE OR REPLACE FUNCTION set_default_street_name()
RETURNS TRIGGER AS $$
BEGIN
   IF NEW.NameOfStreet IS NULL THEN
      NEW.NameOfStreet := 'Unknown street';
   END IF;

   RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER before_insert_street
BEFORE INSERT ON street
FOR EACH ROW
EXECUTE FUNCTION set_default_street_name();
```

4.Функція перевіряє, чи існує вказаний trip_id у таблиці trips, і видає виняток, якщо запис відсутній.

```
CREATE OR REPLACE FUNCTION check_trip_duplicates()
RETURNS TRIGGER AS $$
BEGIN
   IF NOT EXISTS (SELECT 1 FROM trips WHERE trip_id = NEW.trip_id) THEN
```

```
        RAISE EXCEPTION 'Невірний trip_id. Відсутній запис у таблиці trips.';
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER before_insert_update_trip_duplicates
BEFORE INSERT OR UPDATE
ON TripDublicates
FOR EACH ROW
EXECUTE FUNCTION check_trip_duplicates();
```

5.Тригер встановлює значення TypeOfTrip "Default Type" якщо вказанне значення Null.

```
CREATE OR REPLACE FUNCTION set_default_trip_type()
RETURNS TRIGGER AS $$
BEGIN
    IF NEW.TypeOfTrip IS NULL THEN
        NEW.TypeOfTrip := 'Default Type';
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER before_insert_trips
BEFORE INSERT ON trips
```

```
FOR EACH ROW
EXECUTE FUNCTION set_default_trip_type();
```

4.5 Створення функцій

Написати функцію що повертає таблицю з трьома стовпцями: trip_type, trip_number, і trip_name, які відповідають типу поїздки, номеру і назві поїздки відповідно.

```
CREATE OR REPLACE FUNCTION get_trip_info(p_trip_id INT)
RETURNS TABLE (
   trip_type VARCHAR,
   trip_number INT,
   trip_name VARCHAR
) AS $$
BEGIN
   RETURN QUERY (
      SELECT
         TypeOfTrip,
         NumberOfTrip,
         NameOfTrip
      FROM trips
      WHERE Trip_id = p_trip_id
   );
END;
$$ LANGUAGE plpgsql;
```

перевірка SELECT * FROM get_trip_info(1);

Функція для вибірки інформації про сегмент вулиці за його ідентифікатором

```
CREATE OR REPLACE FUNCTION get_street_segment_info(p_segment_id INT)
RETURNS TABLE (
    segment_id INT,
    street_id INT,
    border_houses_number VARCHAR
) AS $$
BEGIN
    RETURN QUERY (
        SELECT
            s.streetSegment_id,
            s.street_id,
            s.BorderHousesNumber
        FROM streetSegment s
        WHERE s.streetSegment_id = p_segment_id
    );
END;
$$ LANGUAGE plpgsql;
```

перевірка

Функція для отримання інформації про послідовність сегментів

```
CREATE OR REPLACE FUNCTION get_sequence_info(sequence_id INT)
RETURNS TABLE (
    trip_id INT,
    segment_id INT,
    sequence_number INT
) AS $$
BEGIN
    RETURN QUERY (
        SELECT
```

```
        st.Trip_id,
        sot.TripSegment_id,
        sot.SequenceNumber
    FROM sequenceOfSegmentsOnTrip sot
    JOIN TripSegment st ON sot.TripSegment_id = st.TripSegment_id
    WHERE sot.sequenceOfSegmentsOnTrip_id = sequence_id
  );
END;
$$ LANGUAGE plpgsql;
```

5 МАНІПУЛЮВАННЯ ДАНИМИ

У PostgreSQL для взаємодії з даними в базі даних використовуються різні типи SQL-запитів: INSERT для введення нових даних, UPDATE для модифікації існуючих, DELETE для видалення та SELECT для вибірки необхідної інформації

5.1Оператори відновлення

Введення данних у таблицю

```
INSERT INTO trips (typeoftrip, numberoftrip, nameoftrip)
VALUES
('City Tour', 101, 'Downtown Exploration'),
('Beach Vacation', 202, 'Sunny Shore Retreat'),
('Mountain Hike', 303, 'Alpine Adventure');
```

```
INSERT INTO tripdublicate (trip_id)
VALUES
(1),
(2),
(3);
```

```
INSERT INTO tripsegment (trip_id, streetsegment_id)
VALUES
(1, 4),
(1, 4),
(2, 5);
```

```
INSERT INTO sequenceofsegmentsontrip (tripsegment_id, trip_id, sequencenumber)
VALUES
(19, 1, 1),
(20, 1, 2),
(21, 2, 1);
```

```
INSERT INTO street (nameofstreet)
VALUES
('Main Street'),
('Elm Street'),
('Oak Avenue');

INSERT INTO streetsegment (street_id, borderHousesNumber)
VALUES
(1, '100-200'),
(2, '300-400'),
(3, '500-600');

INSERT INTO segments (streetsegment_id, tripsegment_id)
VALUES
(4, 19),
(5, 20),
(6, 21);
```

Видаляемо елементи з таблиці:

```
DELETE FROM sequenceofsegmentsontrip
WHERE tripsegment_id = 20;
```

Видаляє усі записи з таблиці "sequenceofsegmentsontrip", де значення стовпця "tripsegment_id" дорівнює 20.

```
DELETE FROM segments
WHERE tripsegment_id = 20;
```

Видаляє усі записи з таблиці "segments", де значення стовпця "tripsegment_id" дорівнює 20.

```
DELETE FROM sequenceofsegmentsontrip
```

WHERE tripsegment_id = 19;

Видаляє усі записи з таблиці "sequenceofsegmentsontrip", де значення стовпця "tripsegment_id" дорівнює 19.

DELETE FROM segments

WHERE tripsegment_id = 19;

Видаляє усі записи з таблиці "segments", де значення стовпця "tripsegment_id" дорівнює 19.

DELETE FROM tripsegment

WHERE trip_id = 1;

Видаляє усі записи з таблиці "tripsegment", де значення стовпця "trip_id" дорівнює 1.

DELETE FROM tripduplicates

WHERE trip_id = 1;

Видаляє усі записи з таблиці "tripduplicates", де значення стовпця "trip_id" дорівнює 1.

DELETE FROM trips

WHERE typeOfTrip = 'City Tour';

Видаляє усі записи з таблиці "trips", де значення стовпця "typeOfTrip" дорівнює 'City Tour'.

DELETE FROM segments

WHERE tripsegment_id = 21;

Видаляє усі записи з таблиці "segments", де значення стовпця "tripsegment_id" дорівнює 21. Це видаляє всі сегменти, пов'язані з поїздкою (tripsegment) з ідентифікатором 21.

```
DELETE FROM sequenceofsegmentsontrip
WHERE tripsegment_id = 21;
```

Видаляє усі записи з таблиці "sequenceofsegmentsontrip", де значення стовпця "tripsegment_id" дорівнює 21. Це видаляє всі дані про послідовність сегментів на поїздці, які відносяться до поїздки з ідентифікатором 21.

```
DELETE FROM tripsegment
WHERE streetsegment_id = 5;
```

Видаляє усі записи з таблиці "tripsegment", де значення стовпця "streetsegment_id" дорівнює 5. Це видаляє поїздки (tripsegment), які включають в себе сегменти з ідентифікатором 5.

```
DELETE FROM streetsegment
WHERE street_id = 2;
```

Видаляє усі записи з таблиці "streetsegment", де значення стовпця "street_id" дорівнює 2. Це видаляє всі сегменти дороги (streetsegment), які відносяться до вулиці з ідентифікатором 2.

```
DELETE FROM street
WHERE nameOfStreet = 'Elm Street';
```

Видаляє усі записи з таблиці "street", де значення стовпця "nameOfStreet" дорівнює 'Elm Street'. Це видаляє запис про вулицю з назвою 'Elm Street'.

Змінюємо талиці за допомогою команди UPDATE

```
UPDATE trips
SET numberoftrip = 222
WHERE trip_id = 2;
```

В даному запиті, я змінив значення numberoftrip для поїздки з ідентифікатором 2. Перед оновленням значення numberoftrip було 202, а після оновлення воно стало 222.

```
UPDATE street
SET nameofstreet = 'Pine Street'
WHERE nameofstreet = 'Main Street';
```

В цьому запиті, я змінив назву вулиці з "Main Street" на "Pine Street" в таблиці "street".

```
UPDATE streetsegment
SET borderHousesNumber = '250-300'
WHERE street_id = 1;
```

Я оновив значення borderHousesNumber для вулиці з ідентифікатором 1. Перед оновленням це значення було '100-200', а після оновлення воно стало '250-300'.

```
UPDATE sequenceofsegmentsontrip
SET sequencenumber = 3
WHERE tripsegment_id = 19;
```

В цьому запиті я змінив значення sequencenumber для послідовності з ідентифікатором 19. Перед оновленням це значення було 1, а після оновлення воно стало 3.

Скриншоти виконаних таблиць:

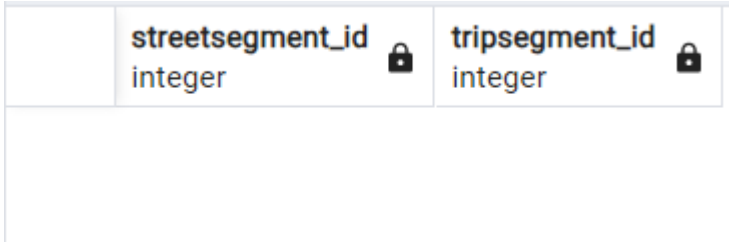

Рис 5.1

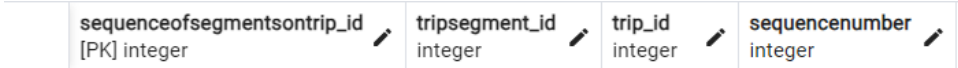

Рис 5.2

	street_id [PK] integer	nameofstreet character varying
1	1	Pine Street
2	3	Oak Avenue

Рис 5.3

	streetsegment_id [PK] integer	street_id integer	borderhousesnumber character varying
1	4	1	250-300
2	6	3	500-600

Рис 5.4

	tripdublicateid [PK] integer	trip_id integer
1	2	2
2	3	3

Рис 5.4

	trip_id [PK] integer	typeoftrip character varying	numberoftrip integer	nameoftrip character varying
1	2	Beach Vacation	222	Sunny Shore Retreat
2	3	Mountain Hike	303	Alpine Adventure

Рис 5.5

	tripsegment_id [PK] integer	trip_id integer	streetsegment_id integer

Рис 5.6

5.2 Оператор вибірки

Оператор вибірки в PostgreSQL, що відомий як SELECT, представляє собою ключовий інструмент для отримання даних з бази даних. Використовуючи його, можна формулювати запити, які вибирають конкретні стовпці з таблиць,

застосовують фільтри для обмеження результатів та використовують різноманітні функції для агрегування даних.

Один із способів використання оператора SELECT - це робота з груповими функціями, що дозволяють групувати рядки на основі значень певного стовпця за допомогою GROUP BY і застосовувати агрегуючі функції, такі як COUNT, SUM, AVG, до цих груп.

Повинно бути відзначено також можливість використання оператора вибірки з умовним оператором WHERE, який дозволяє фільтрувати результати на основі визначених умов для включення рядка в результат.

Додатково, агрегуючі функції, які включають суму (SUM), середнє значення (AVG), кількість (COUNT) тощо, можна використовувати над певними стовпцями. Оператор вибірки також може користуватися спеціальними операторами, такими як LIKE для роботи з рядками та IN для порівняння значень зі списком, що розширює можливості фільтрації даних та вибору необхідної інформації.

Завершальним елементом є використання підзапитів в операторі вибірки, які дозволяють вкладати запити всередину інших запитів, надаючи можливість формулювати більш складні та точні запити та використовувати результати інших запитів для вибору конкретних даних. Усі ці аспекти оператора вибірки в PostgreSQL працюють взаємодіючи, забезпечуючи можливість точного вибору, фільтрації, агрегації та отримання необхідної інформації з бази даних, враховуючи потреби користувача.

```
SELECT COUNT(*) AS total_trips
FROM trips;
```

	total_trips bigint
1	2

Рис 5.6

Виберемо поїздки під назвою Beach Vacation

```
SELECT *
FROM trips
WHERE typeoftrip IN ('Beach Vacation');
```

	trip_id [PK] integer	typeoftrip character varying	numberoftrip integer	nameoftrip character varying
1	2	Beach Vacation	222	Sunny Shore Retreat

Рис 5.7

Основний запит включає підзапит для розрахунку кількості сегментів вулиць для кожної поїздки в таблиці tripsegment. Затим він вибирає інформацію про поїздки з таблиці trips, додаючи кількість пов'язаних сегментів вулиць для кожної з них.

```
SELECT trip_id,
    typeoftrip,
    numberoftrip,
    (SELECT COUNT(*) FROM tripsegment WHERE tripsegment.trip_id = trips.trip_id) AS segment_count
FROM trips;
```

	trip_id [PK] integer	typeoftrip character varying	numberoftrip integer	segment_count bigint
1	3	Mountain Hike	303	1
2	2	Beach Vacation	222	1

Рис 5.8

6 СТВОРЕННЯ КОРИСТУВАЧІВ І ПРИЗНАЧЕННЯ ПРАВ ДОСТУПУ

Створення користувачів у PostgreSQL включає створення облікових записів, які мають доступ до бази даних та набору привілеїв, які визначають, як ці користувачі можуть взаємодіяти з даними та об'єктами бази даних.

При створенні користувача в базі даних можна надати йому конкретні права на виконання певних операцій з об'єктами бази даних (такими як SELECT, INSERT, UPDATE, DELETE та інші) або ж привілеї для виконання певних функцій у системі.

```
--Створювання користувача - Адміна системи та надання йому всіх прав

CREATE USER admin_user WITH PASSWORD '123';
ALTER USER admin_user WITH SUPERUSER;


-- Створювання користувача - Користувач системи та даннадя йому права на перегляд інформації
CREATE USER regular_user WITH PASSWORD '123';
GRANT SELECT ON TABLE segments TO regular_user;
GRANT SELECT ON TABLE sequenceofsegmentsontrip TO regular_user;
GRANT SELECT ON TABLE street TO regular_user;
GRANT SELECT ON TABLE streetsegment TO regular_user;
GRANT SELECT ON TABLE tripdublicates TO regular_user;
GRANT SELECT ON TABLE trips TO regular_user;
GRANT SELECT ON TABLE tripsegment TO regular_user;


Перевірка SET ROLE regular_user;
SELECT * FROM street
```

	street_id [PK] integer	nameofstreet character varying
1	3	Oak Avenue
2	1	Pine Street
3	4	Main Street
4	5	Second Main Street
5	6	Unknown street

Рис 6.1

Перевірка SET ROLE admin_user;

UPDATE trips SET numberoftrip = 5

where trip_id = 2

SELECT * FROM public.trips

	trip_id [PK] integer	typeoftrip character varying	numberoftrip integer	nameoftrip character varying
1	2	Beach Vacation	5	Sunny Shore Retreat
2	3	Mountain Hike	304	Alpine Adventure
3	8	Поїздка	1	New Trip

Рис 6.2

Висновок: У результаті виконання курсової роботи була створена система бази даних для маршрутів громадського транспорту з можливістю зберігання топологічної схеми маршрутів. Оцінюючи результати роботи системи, можна виділити декілька ключових аспектів.

Позитивні аспекти та досягнення:

Структура бази даних: Була ретельно спроектована структура бази даних, що відповідає всім сучасним стандартам та вимогам до інформаційних систем, специфічним для маршрутів громадського транспорту. Реалізація обраної функціональності: Були розроблені та успішно впроваджені таблиці, послідовності, тригери, представлення та збережені процедури, забезпечуючи цілісність, безпеку та консистентність даних. Ефективні механізми запитів: Система надає ефективні механізми для вибірки, вставки, оновлення та видалення даних, що дозволяє здійснювати швидку та зручну роботу з інформацією. Автоматизовані процеси: Введені автоматизовані процеси архівації даних сприяють збереженню історії взаємодій та зменшують ризики втрати інформації

Негативні аспекти та обмеження:

Складність деяких запитів: Виявлено деякі обмеження, пов'язані зі складністю реалізації деяких типів запитів, що може вимагати подальшого оптимізування та налагодження.

Область використання результатів роботи:

Маршрути громадського транспорту: Розроблена система може знайти застосування у системы громадського транспорту, або додатку для потрапляння з пункту А в пункт Б.

Висновки стосовно розробленої системи маршрутів громадського транспорту

вказують на її ефективність у знаходження типів маршрутів та маршрутних дублікатів. Має точну інформацію щодо вулиць, сегметнів маршрутів, сегментів вулиць та послідовності сегментів на маршруті.

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.op.edu.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.op.edu.ua.

4. Глава М.Г. Методичні вказівки до виконання лабораторних робіт з дисципліни „Організація баз даних та знань “.– Режим доступу: http://library.op.edu.ua.

5. Глава М.Г. Методичні вказівки до самостійної роботи з дисципліни „Організація баз даних та знань “. – Режим доступу: http://library.op.edu.ua.