МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВОЛИНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ЛЕСІ УКРАЇНКИ

Географічний факультет
Кафедра геодезії, землевпорядкування та кадастру

ГЕОПРОСТОРОВІ БАЗИ ДАНИХ
ЛАБОРАТОРНИЙ ПРАКТИКУМ

рівень вищої освіти	другий (магістерський)
галузь знань	19 Архітектура та будівництво
спеціальність	193 Геодезія та землеустрій
освітня професійна програма	Геодезія та землеустрій

Луцьк – 2022

528.4:[378.22:001.817(072)
М 54

Рекомендовано до друку методичною радою Волинського національного університету імені Лесі Українки (протокол № 10 від 21 червня 2022 р.)

Рецензенти:
Барський Ю.М., доктор економічних наук, професор, декан географічного факультету,
Мельник Ю.А., кандидат технічних наук, доцент, доцент кафедри будівництва та цивільної інженерії Луцького НТУ

М54
Геопросторові бази даних. Лабораторний практикум: навч.-метод. вид. для студентів геогр. ф-ту / О.В. Мельник. – Луцьк: РВВ "Вежа-Друк", 2022. – 70 с.

Методичні вказівки призначені для використання студентами при вивченні освітньої компоненти "Геопросторові бази даних" за освітньо-професійною програмою Геодезія та землеустрій спеціальності 193 Геодезія та землеустрій галузі знань 19 Архітектура та будівництво денної та заочної форм навчання.

У даній методичній розробці сформульовано загальні положення, необхідні для виконання лабораторних робіт з дисципліни "Геопросторові бази даних", а також розглянуті сучасні технології обробки геопросторової інформації у СКБД PostgreSQL та розширення PostGIS, моделі, що лежать у їх основі, сучасні напрями застосування баз геопросторових даних і перспективи розвитку.

528.4:[378.22:001.817(072)
М 54

© Мельник О.В. 2022

Вступ

1.1. Загальна інформація до практичних робіт

Ці розділи відповідають ряду умовностей, щоб полегшити стеження за матеріалом практичних робіт. У цьому розділі наведено короткий огляд того, що буде траплятись у практичних роботах та яким чином це буде відображено в тексті.

1.1.1. Напрямки

Дії які необхідно здійснити, будуть відзначені **жирним** шрифтом.

Наприклад:

Щоб продовжити, натисніть кнопку **Next**.

1.1.2. Код

Приклади запитів SQL відображатимуться в полі

```
SELECT postgis_full_version();
```

Ці приклади можна ввести у вікно запиту або інтерфейс командного рядка.

1.1.3. Примітки

Примітки використовуються для надання корисної, але не критичної інформації для загального розуміння теми і виділятиметься знаком зірочки *:

*Якщо Ви не з'їли яблуко сьогодні, лікар може бути в дорозі.

1.1.4. Функції

Там, де назВи функцій визначено в тексті, вони будуть відтворюватися **жирним** шрифтом.

Наприклад:

ST_Touches(geometry A, geometry B) повертає TRUE, якщо перетинає будь-яку з меж геометрії

1.1.5. Файли, таблиці та імена стовпців

Імена файлів, шляхи, імена таблиць і стовпців буде показано шрифтом фіксованої ширини.

Наприклад:

Виберіть стовпець `name` в таблиці `nyc_streets`.

1.1.6. Меню та елементи інтерфейсу

Меню/підменю та елементи інтерфейсу, такі як поля або прапорці та інші чекбокси на екрані, відображаються *курсивом*.

Наприклад:

У меню *File* виберіть пункт *New*. Установіть прапорець *Confirm*.

1.1.7. Робочий процес

Практичні роботи розроблені таким чином, щоб бути прогресивними. Кожна практична робота почнеться з припущення, що Ви завершили і зрозуміли попередній розділ серії і буде спиратися на ці знання. Практичні роботи будуть прогресувати через кілька ідей і супроводжуватись робочими прикладами, де це можливо. Наприкінці практичних робіт, де це доречно, ми включили кілька вправ, які дозволять вам випробувати ідеї, які ми представили. У деяких випадках розділ буде включати в себе "Варто спробувати". Ці

завдання містять більш складні проблеми, ніж вправи, і покликані кинути виклик слухачам курсу з передовими знаннями.

Практична робота №1. Інсталяція PostgreSQL та розширення PostGIS

1.1. Підготовчий етап.

PostgreSQL – це потужна об'єктно-реляційна система баз даних з відкритим вихідним кодом з більш ніж 30-річною активною розробкою, яка заслужила репутацію надійності, надійності та продуктивності.

PostGIS — просторове розширення бази даних для об'єктно-реляційної бази даних PostgreSQL. Він додає підтримку географічних об'єктів, що в свою чергу дозволяє виконувати просторові запити в мові SQL. Наприклад:

```
SELECT superhero.name
FROM city, superhero
WHERE ST_Contains(city.geom, superhero.geom)
AND city.name = 'Gotham';
```

На додаток до базової обізнаності про місцезнаходження, PostGIS пропонує багато функцій, рідко зустрічаються в інших конкуруючих просторових базах даних, таких як Oracle Locator / Spatial і SQL Server.

PostgreSQL можна вільно завантажити за адресою https://www.enterprisedb.com/downloads/postgres-postgresql-downloads, або за посиланням з курсу в системі дистанційного навчання Moodle.

1.2. Встановлення PostgreSQL та розширення PostGIS

Процес встановлення досить простий і включає декілька нескладних етапів.

1. На привітальному вікні після запуску інсталяції натисніть **Next**

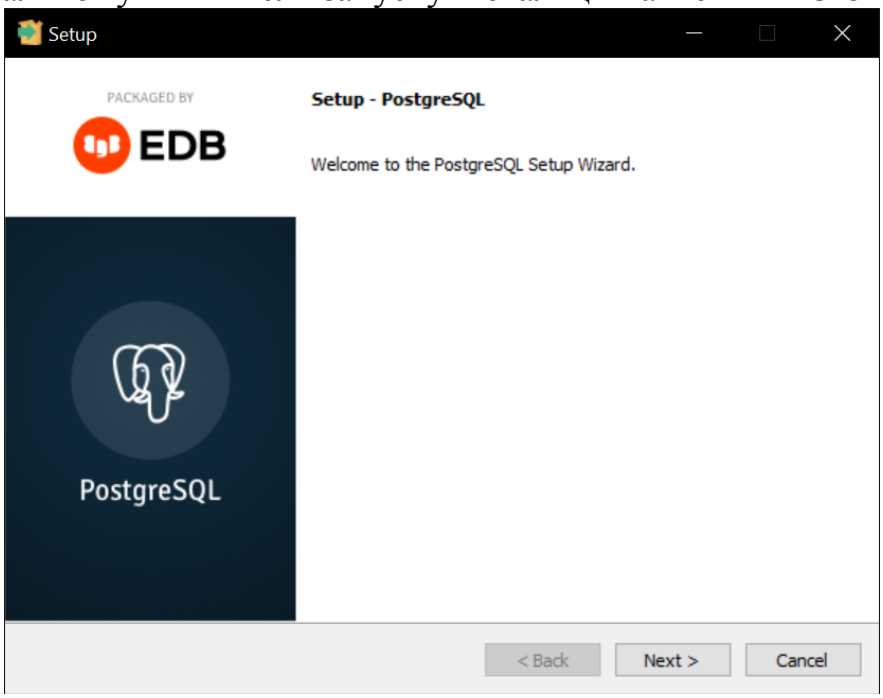

Рис. 1. Початок встановлення СКБД PostgreSQL

2. В наступному вікні оберіть теку куди буде інстальовано дистрибутив PostgreSQL та натисніть **Next**.
   *Наполегливо рекомендується не змінювати шлях за замовчуванням.

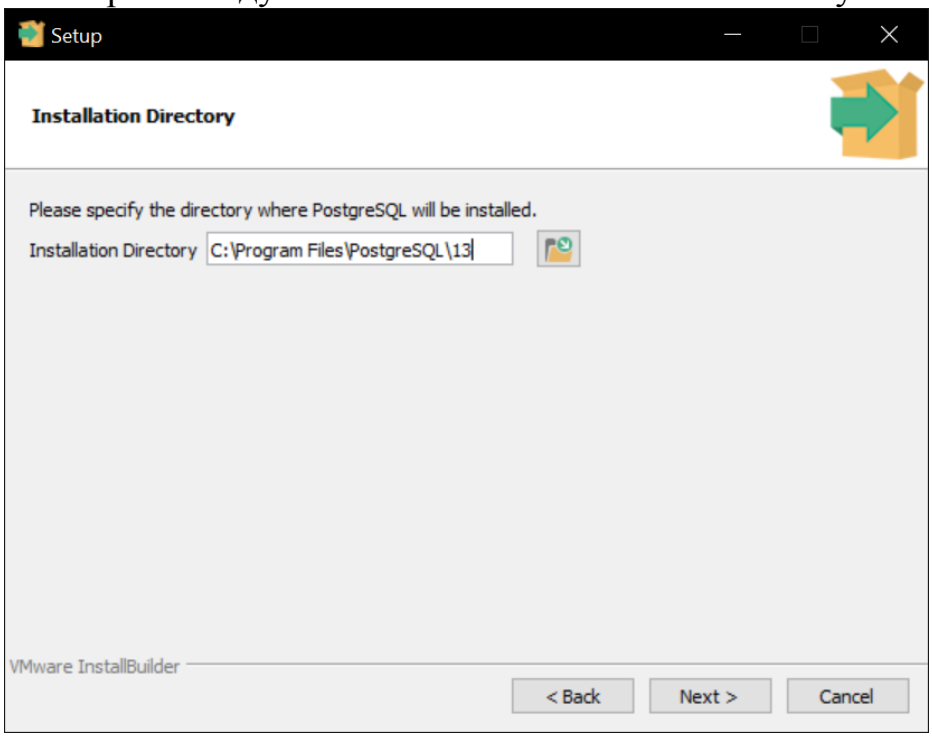

Рис. 2. Вибір директорії для встановлення СКБД PostgreSQL

3. Наступним з'явиться вікно вибору компонентів для інсталяції. На вибір пропонуються:

- Безпосередньо сервер баз даних PostgreSQL;
- Утиліта для графічного адміністрування баз геоданих pgAdmin4;
- Редактор інсталяцій Stack Builder;
- Утиліти для командної стрічки.

  Обираємо ВСІ пункти та натискаємо **Next**.

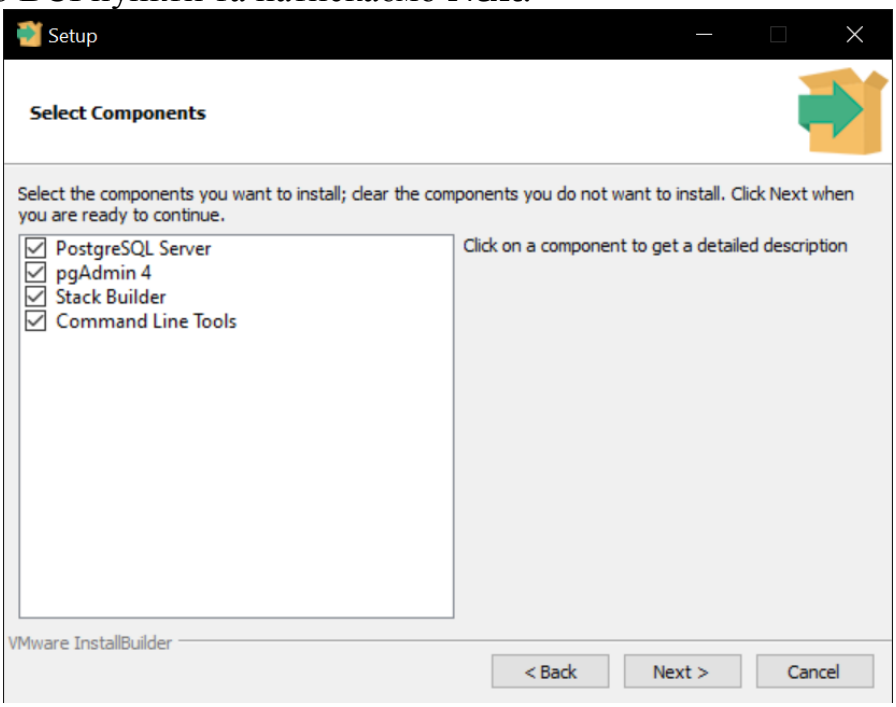

Рис. 3. Вибір компонентів СКБД PostgreSQL

4. В наступному вікні необхідно задати директорію де будуть зберігатись безпосередньо дані які міститимуться в базах даних. Оберіть зручне для Вас розташування з огляду на вільний простір і швидкість накопичувача та натисніть **Next.**

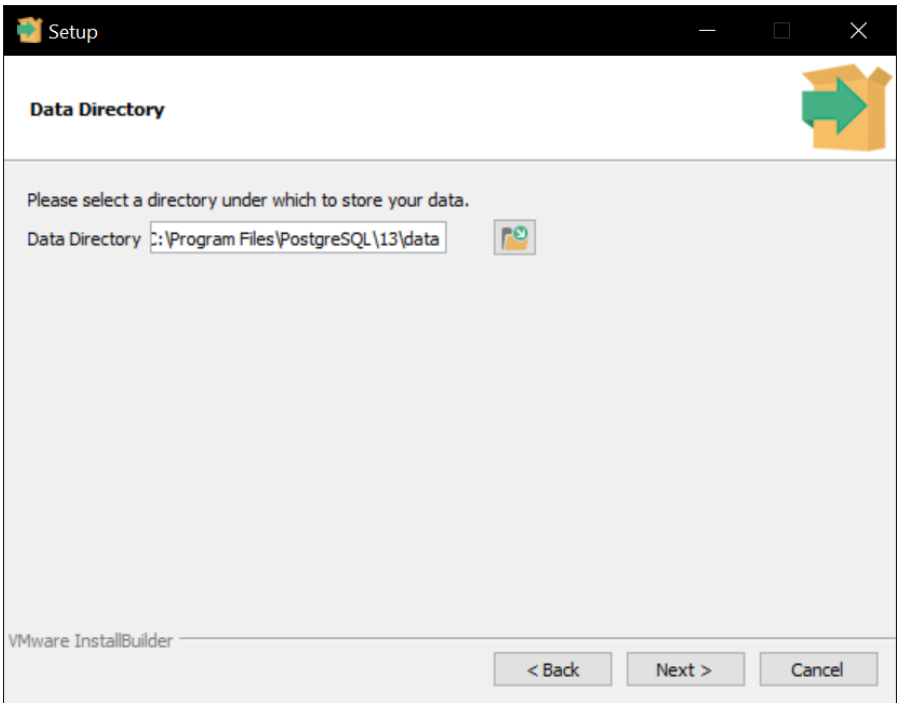

Рис. 4. Вибір директорії для збереження даних СКБД PostgreSQL

5. На наступному кроці необхідно створити та обов'язково запам'ятати пароль адміністратора до СКБД та натискаємо **Next.**

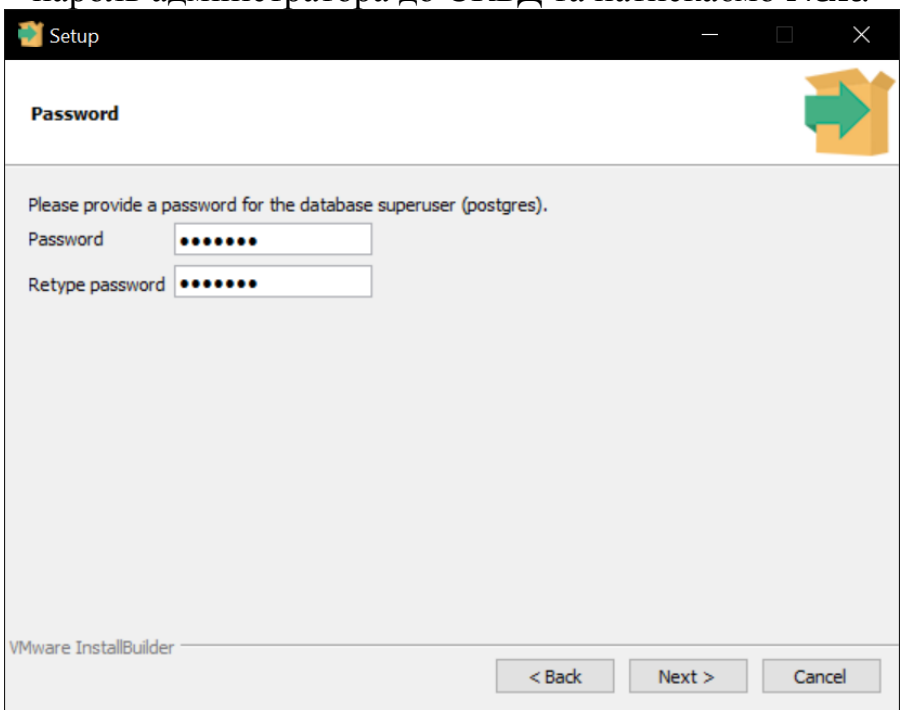

Рис. 5. Створення паролю адміністратора СКБД PostgreSQL

6. Далі – обираємо мережевий порт який "слухатиме" Ваш комп'ютер для доступу до системи керування базами даних (СКБД). Рекомендується

залишити порт за замовчуванням 5432 і запам'ятати його після чого натискаємо **Next**.

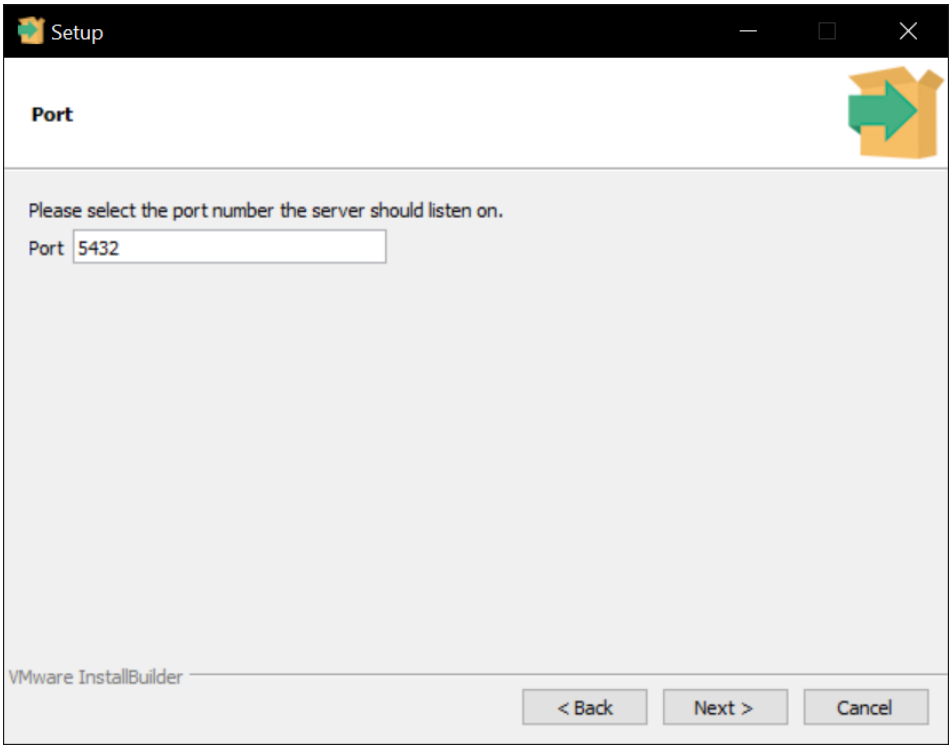

Рис. 6. Вибір мережевого порту для "прослуховування" СКБД PostgreSQL

7. У наступному вікні вибору локалізації залишаємо без змін та натискаємо **Next**.

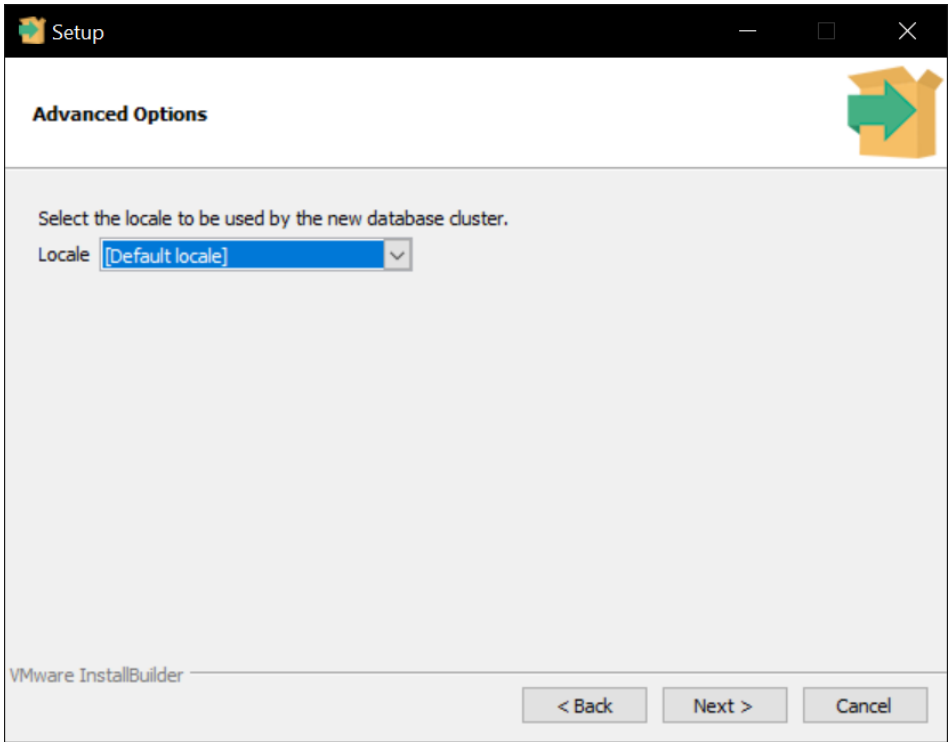

Рис. 7. Вибір локалізації СКБД PostgreSQL

8. В останньому вікні перевіряємо внесені в попередніх кроках налаштування та натискаємо **Next** та очікуємо завершення інсталяції.

9. По завершенню інсталяції PostgreSQL відкриваємо застосунок Application Stack Builder та в випадаючому меню обираємо свою інсталяцію та натискаємо **Next.**

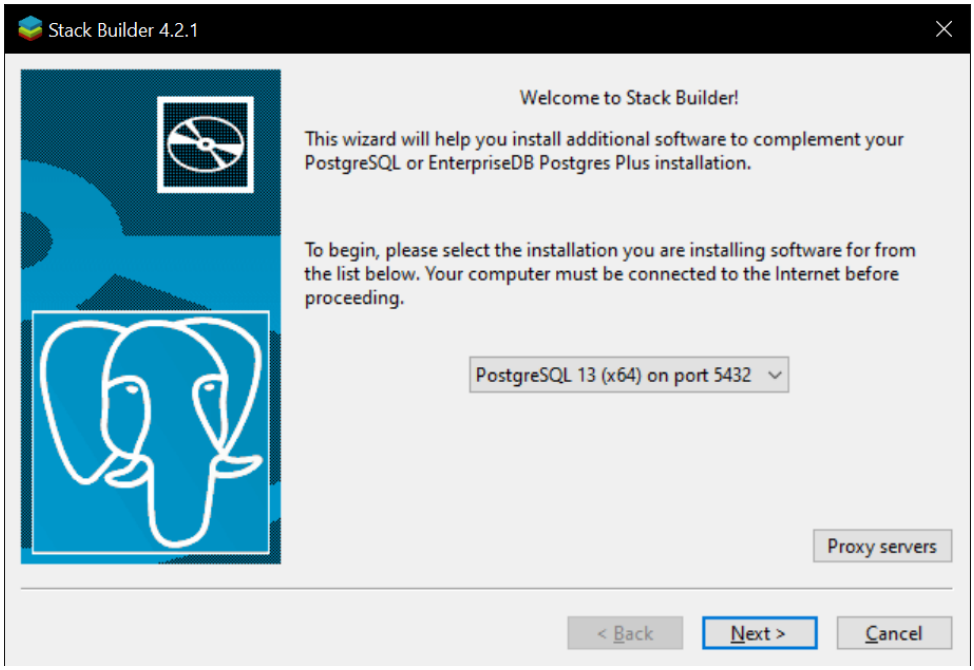

Рис. 8. Початкове вікно застосунку Stack Builder

10. Далі в субменю *Spatial Extensions* ставимо прапорець навпроти *PostGIS 3.1 Bundle fot PostgreSQL 13 (64bit) v.3.1.1*, натискаємо **Next** та обираємо директорію для завантаження в наступному вікні, після чого знову натискаємо **Next.**

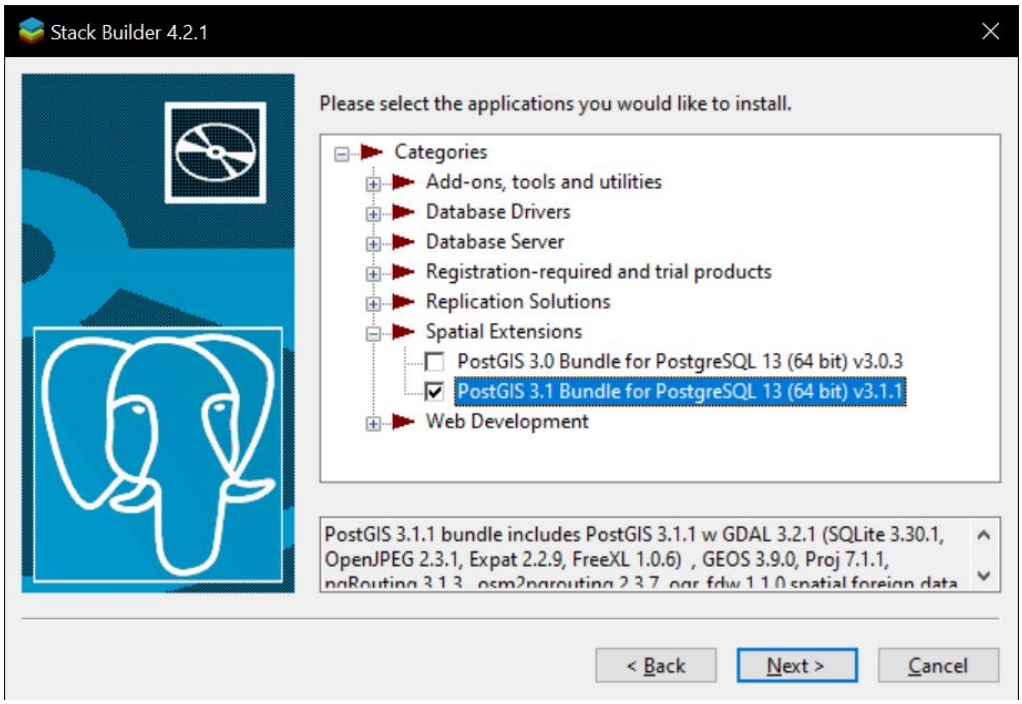

Рис. 9. Вибір елементів для встановлення у застосунку Stack Builder

11. Очікуємо завершення завантаження та підтверджуємо інсталяцію натисканням **Next.**

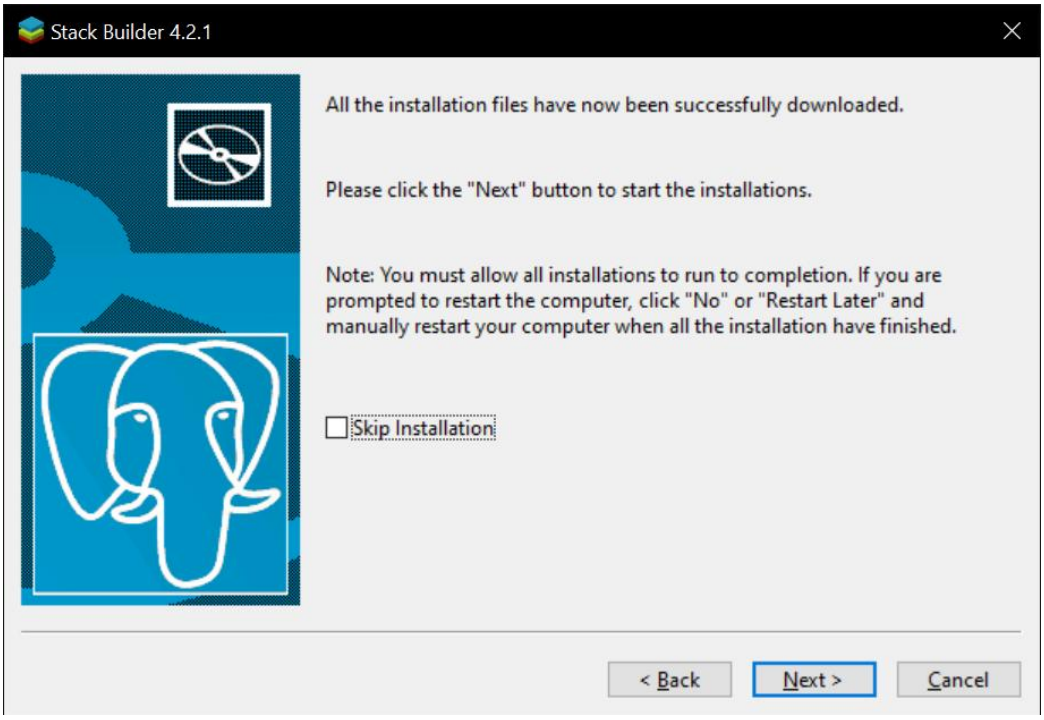

Рис. 10. Вікно підтвердження завантаження та встановлення обраних елементів

12. У вікні встановлення PostGIS продовжуємо інсталяцію залишивши всі пункти "за замовчуванням".

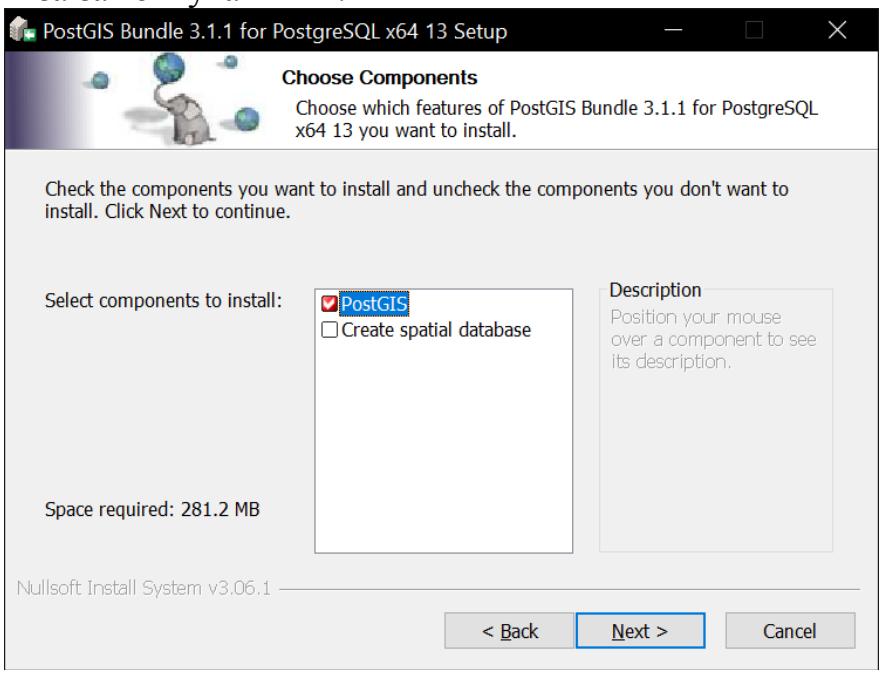

Рис. 11. Початкове вікно встановлення розширення PostGIS

У виринаючому вікні погоджуємося на реєстрацію змінних в системі натискаючи **Yes,**

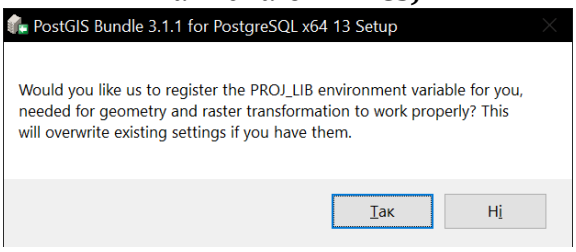

Рис. 12. Вікно підтвердження реєстрації компонентів PostGIS

після чого завершуємо інсталяцію натисканням **Finish**

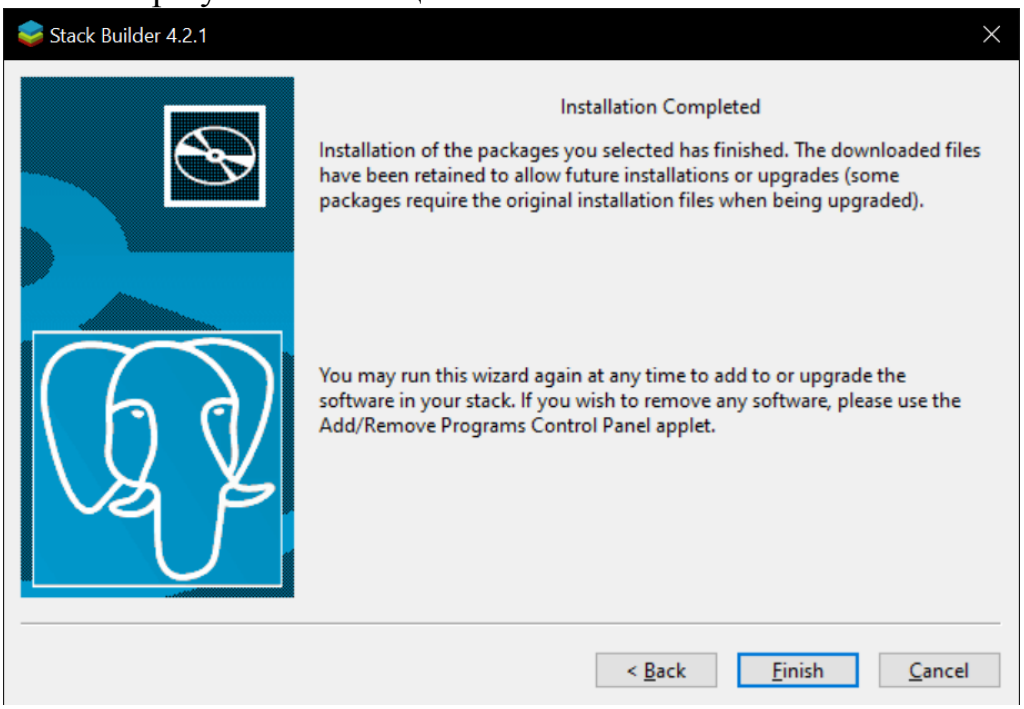

Рис. 13. Завершення встановлення PostGIS

Практична робота №2. Створення просторової бази даних та знайомство з інтерфейсом pgAdmin 4¶

2.1. PgAdmin 4

PostgreSQL має ряд адміністративних фронт-ендів (застосунків). Основним з них є psql – інструмент командного рядку для введення SQL запитів. Іншим популярним інтерфейсом PostgreSQL графічний інструмент з відкритим вихідним кодом pgAdmin. Всі запити, виконані в pgAdmin, також можуть бути виконані в командному рядку з psql.

1. Знайдіть ярлик запуску pgAdmin та запустіть його.
2. При першому завантаженні pgAdmin необхідно ввести пароль адміністратора, який Ви створили при інсталяції СКБД. За замовчуванням, у Вас вже є зконфігурований доступ до локального сервера **PostGIS (localhost:5432).** Двічі клацніть запис і введіть у відповідь на запит пароля про підключення до бази даних.Базу даних PostGIS інстальовано з необмеженим доступом для локальних користувачів (користувачі підключаться з того самого комп'ютера, що й база даних). Це означає, що він прийме *будь-який пароль,* який Ви надаєте. Якщо потрібно підключитися з віддаленого комп'ютера, пароль для користувача `postgres` встановлено під час інсталяції.

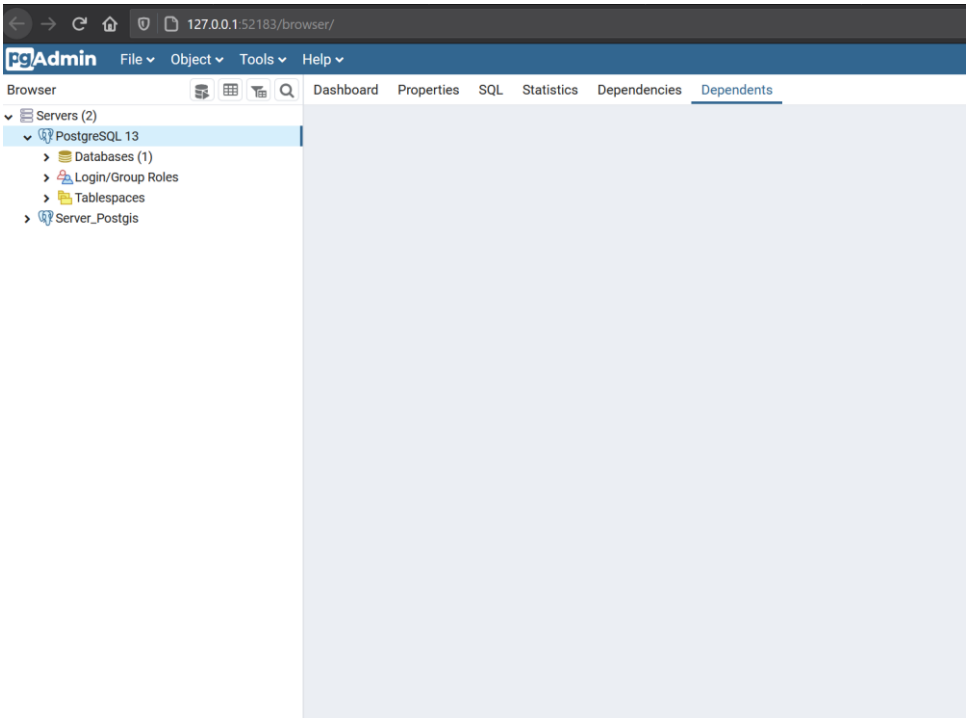

Рис. 14. Початкове вікно pgAdmin 4 в інтернет-браузері

2.2. Створення бази даних

1. Відкрийте елемент дерева баз даних і перегляніть доступні бази даних. База даних `postgres` є базою даних користувачів для PostgreSQL за замовчуванням і не надто цікава для нас.
2. Натисніть правою кнопкою миші на пункт *Databases* і оберіть *New Database*

3. У вікні, що з'явилось введіть ім'я бази nyc даних в поле **Name** та оберіть власника (**Owner**) – postgres.

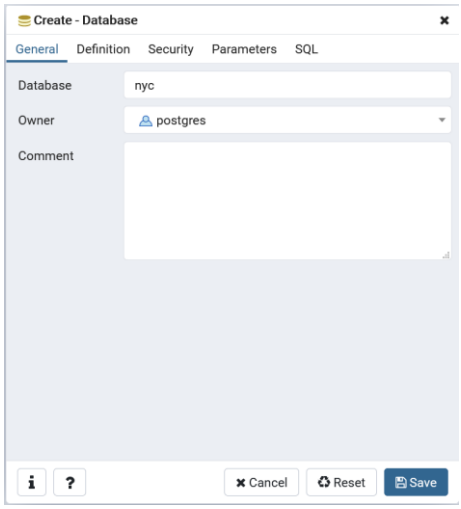

Рис. 15. Створення БД у pgAdmin 4

4. Виберіть нову базу даних nyc і відкрити його для відображення дерева об'єктів. Ви побачите схему public.

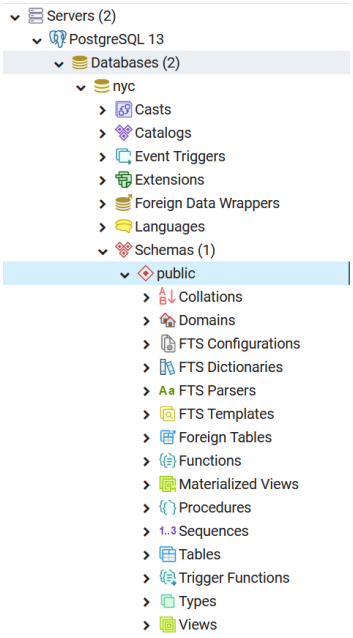

Рис. 16. Структура новоствореної БД у pgAdmin 4

5. Відкрийте редактора запитів з меню *Tools* > *Query Tool.* Введіть наступний запит в поле редактора запитів та натисніть кнопку **Execute/Refresh** або на клавіатурі **F5.**

```
CREATE EXTENSION postgis;
```

Даний запит додасть розширення PostGIS до новоствореної бази даних nyc

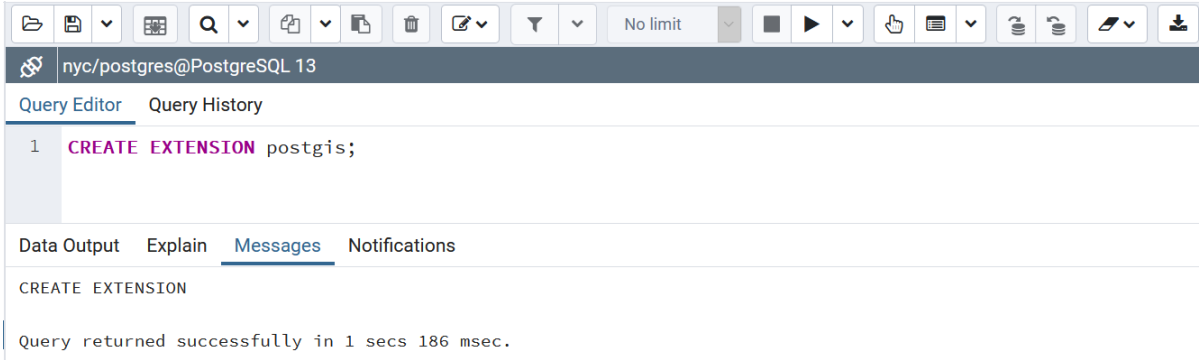

Рис. 17. Вікно редактора запитів до БД у pgAdmin 4

6. Тепер переконайтеся, що PostGIS встановлено за допомогою функції PostGIS:

```
SELECT postgis_full_version();
```

Якщо все зроблено вірно і працює в полі *Data Output* буде відображено повідомлення накшталт:
POSTGIS="3.1.1 3.1.1" [EXTENSION] PGSQL="130" GEOS="3.9.0-CAPI-1.14.1" PROJ="7.1.1" LIBXML="2.9.9" LIBJSON="0.12" LIBPROTOBUF="1.2.1" WAGYU="0.5.0 (Internal)".
*В залежності від версії застосунків дане повідомлення може змінюватись.

PostGIS_Full_Version: Повідомляє про повну версію PostGIS та інформацію про конфігурацію збірки.

Практична робота №3. Завантаження просторових даних у БД та їх візуалізація засобами ГІС QGIS

За підтримки найрізноманітніших бібліотек і додатків, PostGIS надає безліч варіантів завантаження даних. Ця робота буде зосереджена на основах – завантаженні векторних файлів у форматі *.shp за допомогою інструменту завантаження PostGIS.

3.1. Що таке шейпфайли (shapefiles)?

«Shapefile» зазвичай посилається на набір файлів із розширеннями .shp, .shx, .dbfта іншими розширеннями назагальне ім'я префікса (наприклад, nyc_census_blocks). Фактичний файл shapefile стосується саме файлів із розширенням .shp. Проте сам файл .shp неповний для розповсюдження без необхідних допоміжних файлів.

Обов'язкові файли:

.shp— формат фігури; сама особливість геометрії

.shx— формат індексу фігури; позиційний індекс геометрії об'єкта

.dbf—атрибути; атрибути стовпця для кожної фігури в dBase III

До необов'язкових файлів належать:

.prj— проекційний формат; система координат та інформація про проекцію, простий текстовий файл, що описує проекцію з використанням відомого текстового формату

PgShapeLoader робить дані фігури корисними в PostGIS, перетворивши їх з двійкових даних на ряд команд SQL, які потім запускаються в базі даних для завантаження даних.

"SRID" означає "Spatial Reference IDentifier". Він визначає всі параметри географічної системи координат і проекції наших даних. SRID зручний тим, що пакує всю інформацію про проекцію карти (яка може бути досить складною) в єдине число.

Ви можете побачити визначення нашої проекції карти в даних до практичних робіт, шукаючи його або в онлайн-базі даних за адресою http://spatialreference.org/ref/epsg/26918/ або безпосередньо всередині PostGIS із запитом до таблиці `spatial_ref_sys`. Для цього виконайте наступний запит до БД:

```
SELECT srtext FROM spatial_ref_sys WHERE srid = 26918;
```

В обох випадках Ви бачите текстове представлення системи координат **26918** (нижче приведено для наочності):

```
PROJCS["NAD83 / UTM zone 18N",
  GEOGCS["NAD83",
    DATUM["North_American_Datum_1983",
      SPHEROID["GRS
1980",6378137,298.257222101,AUTHORITY["EPSG","7019"]],
      AUTHORITY["EPSG","6269"]],
    PRIMEM["Greenwich",0,AUTHORITY["EPSG","8901"]],

UNIT["degree",0.01745329251994328,AUTHORITY["EPSG","9122"]],
    AUTHORITY["EPSG","4269"]],
  UNIT["metre",1,AUTHORITY["EPSG","9001"]],
  PROJECTION["Transverse_Mercator"],
  PARAMETER["latitude_of_origin",0],
  PARAMETER["central_meridian",-75],
  PARAMETER["scale_factor",0.9996],
  PARAMETER["false_easting",500000],
  PARAMETER["false_northing",0],
  AUTHORITY["EPSG","26918"],
  AXIS["Easting",EAST],
  AXIS["Northing",NORTH]]
```

Якщо Ви відкриєте файл `nyc_neighborhoods.prj` з каталогу даних, то побачите те ж саме визначення проекції.

Поширеною проблемою для людей, які починають роботу з PostGIS, є з'ясування того, який номер SRID використовувати для своїх даних.

Проста відповідь полягає у використанні комп'ютера. Підключіть вміст файлу .prj до http://prj2epsg.org. Це дасть вам число (або список чисел), які найбільше відповідають визначенню проекції. На цьому сайті міститься інформація про найбільш поширені системи координат.

Наша проекція - "Універсальна поперечна проекція Меркатора (UTM) Зона 18 Північ" або EPSG:26918.

3.2. Завантеження векторних даних у БД

1. Перш за все, необхідно завантажити інструмент для імпорту *.shp файлів PostGIS Shapefile Export/Import Manager. Зазвичай, дана утиліта знаходиться за шляхом C:\Program Files\PostgreSQL\13\bin\postgisgui\shp2pgsql-gui.exe

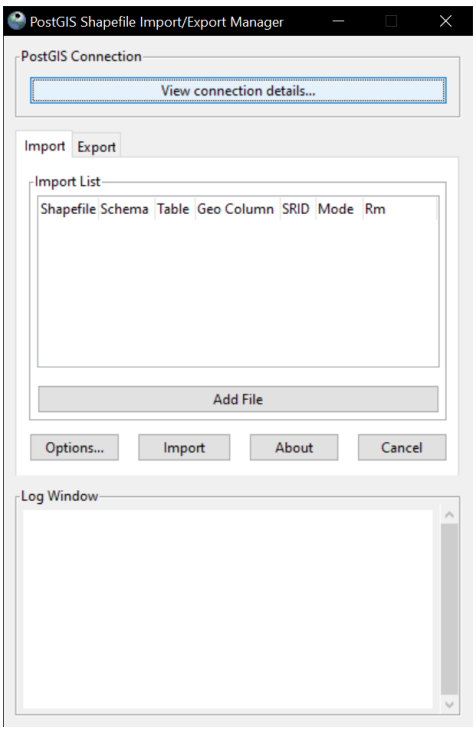

Рис. 18. Менеджер імпорту-експорту векторних даних

2. Заповніть деталі з'єднання для розділу PostGIS Connection і натисніть кнопку OK. Завантажувач протестуватиме з'єднання та повідомить про це у вікні журналу.

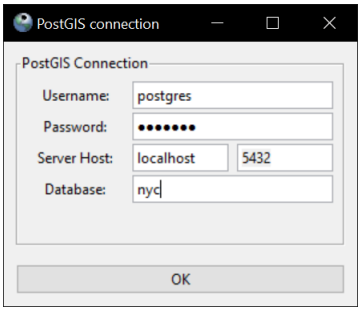

Рис. 19. Менеджер з'єднань

3. Потім відкрийте браузер додавання файлів *Add File* і перейдіть до каталогу даних data. Архів із вихідними файлами можна завантажити з курсу в системі Moodle або безпосередньо за посиланням https://volnu-my.sharepoint.com/:u:/g/personal/hockins_vnu_edu_ua/Eefr5cSosdxHiHswHO_ivRYBebEgKRA2l6mboSjp6FfynQ?e=2iS9Mr
4. Виберіть файл nyc_census_blocks.shp. Переконайтеся, що у шляху до файлів не міститься спеціальних та кириличних символів.
5. Змініть значення **SRID** для файлу на **26918**. Зауважте, що ім'я схеми, таблиці та стовпця вже заповнено з *.shp файлу, але за бажанням їх можна змінити. Клацніть із полів після закінчення редагування, щоб переконатися, що зміни було введено.

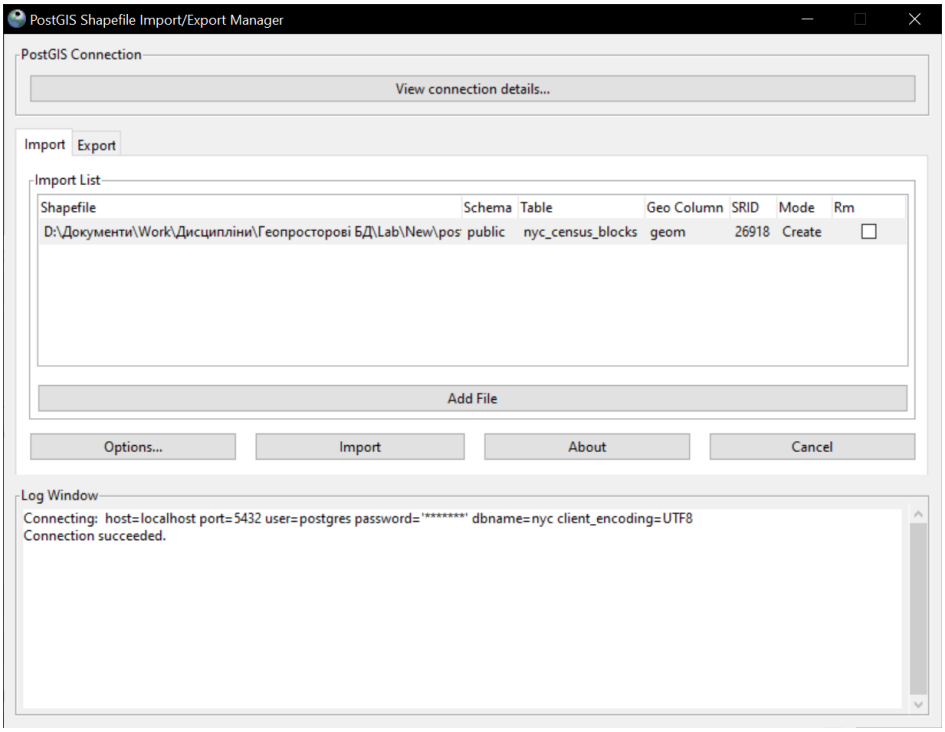

Рис. 20. Менеджер імпорту-експорту векторних даних із завантаженим файлом

6. Натисніть **Options**, щоб переглянути параметри завантаження. Завантажувач буде використовувати швидкий режим "COPY" і створювати просторовий індекс за замовчуванням після завантаження даних.

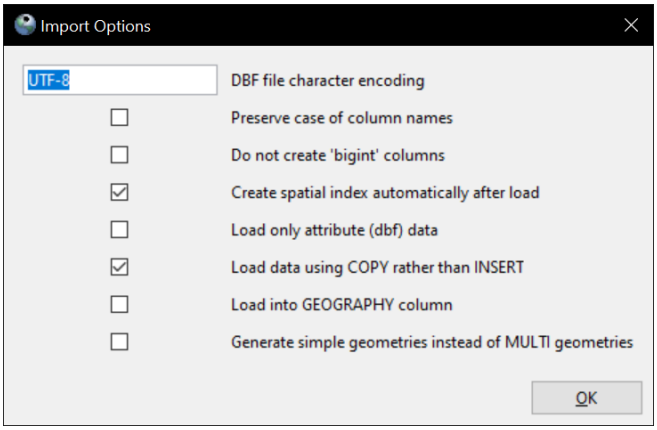

Рис. 21. Параметри імпорту- векторних даних до БД

7. Нарешті, натисніть **Import** і спостерігайте за процесом імпортування. Завантаження може тривати кілька хвилин, але це найбільший файл у нашому тестовому наборі.
8. Повторіть процедуру імпортування решти файлів *.shp у каталозі даних. Ви можете завантажити декілька файлів за один імпорт, додавши декілька файлів перед натисканням кнопки **Import:**
    - nyc_streets.shp
    - nyc_neighborhoods.shp
    - nyc_subway_stations.shp
    - nyc_homicides.shp

Коли всі файли завантажені, натисніть кнопку **Refresh** в pgAdmin, щоб оновити вигляд дерева. Ви повинні побачити ваші чотири таблиці, що відображаються в розділ дерева *Databases* > *nyc* > *Schemas* > *public* > *Tables*.

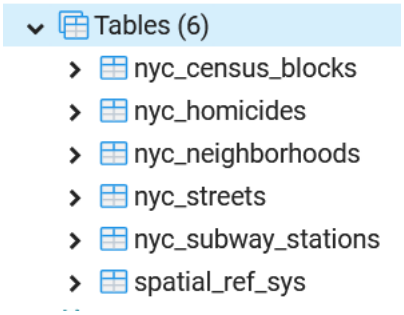

Рис. 22. Результат імпорту векторних даних із п'яти файлів

Для імпорту табличних даних без геоданих

Завдання до самостійної роботи

Підключіть новостворену Вами базу даних у PostGIS в середовище ГІС QGIS та візуалізуйте дані з БД оверлейом на мапу OpenStreetMap або будь яку іншу згідно прикладу, що зображений нижче. (рис. 23). Обов'язковими елементами оформлення є рамка, масштаб, напрямок півночі, легенда та ім'я та група виконавця.

Підготуйте файл *.pdf з візуалізованими даними.

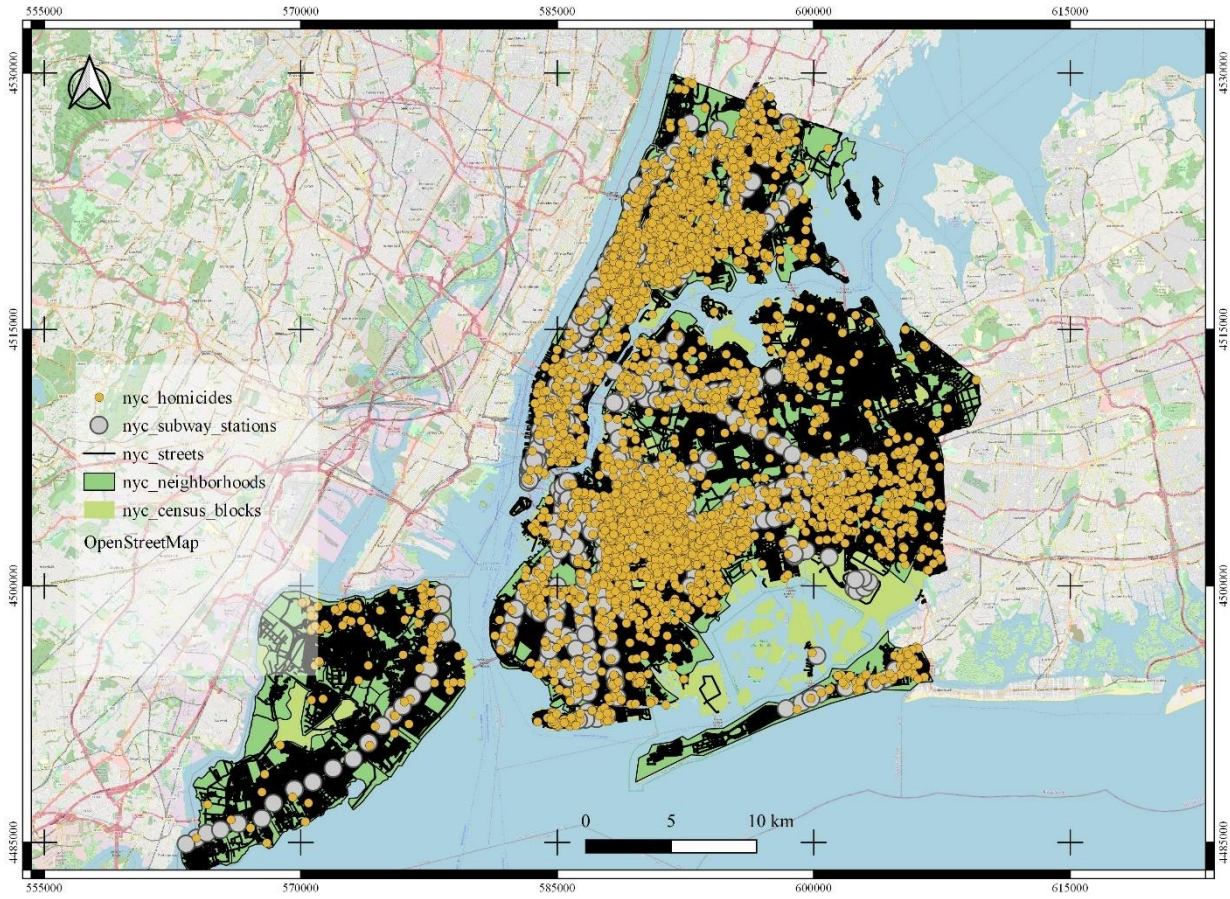

Рис. 21. Зразок оформлення результатів практичної роботи

Практична робота №4. Аналіз даних

Дані для практичних робіт містяться у чотиртох шейп-файлах для міста Нью-Йорк та однією таблицею атрибутів соціодемографічних змінних. Ми завантажили наші файли як таблиці PostGIS і додамо соціодемографічні дані пізніше під час виконання робіт.

Нижче описано кількість записів і атрибутів таблиці для кожного з наших наборів даних. Ці атрибути є фундаментальними для виконання подальших робіт.

Щоб дослідити природу таблиць у pgAdmin, клацніть виділену таблицю правою кнопкою миші та виберіть пункт **Properties**. На вкладці *Columns* Ви знайдете зведення властивостей таблиці, включно зі списком **атрибутів** таблиці.

4.1. Таблиця nyc_census_blocks

Блок перепису - це найменша одиниця, для якої повідомляються дані перепису. Всі дані перепису населення вищого рівня (райони, квартали, райони метро, округи і т.д.) можуть бути побудовані з об'єднань блоків перепису. Ми прикріпили деякі демографічні дані до вихідних даних для практичних робіт.

Кількість записів: 36592

Таблиця 1.

Опис таблиці даних **nyc_census_blocks**

Назва стовпчика	Опис даних
blkid	15-значний код, який однозначно ідентифікує кожен блок перепису. Наприклад: 360050001009000
popn_total	Загальна кількість людей у блоці перепису населення
popn_white	Кількість людей, які самоідентифікують себе як "Білі" в блоці
popn_black	Кількість людей, які самоідентифікують себе як "Темношкірі" у блоці
popn_nativ	Кількість людей, які самоідентифікують себе як "Індіанці" в блоці
popn_asian	Кількість людей, які самоідентифікують себе як "Азіати" в блоці
popn_other	Кількість людей, які самостійно ідентифікують себе з іншими категоріями в блоці
boroname	району Нью-Йорка. Манхеттен, Бронкс, Бруклін, Стейтен-Айленд, Квінз

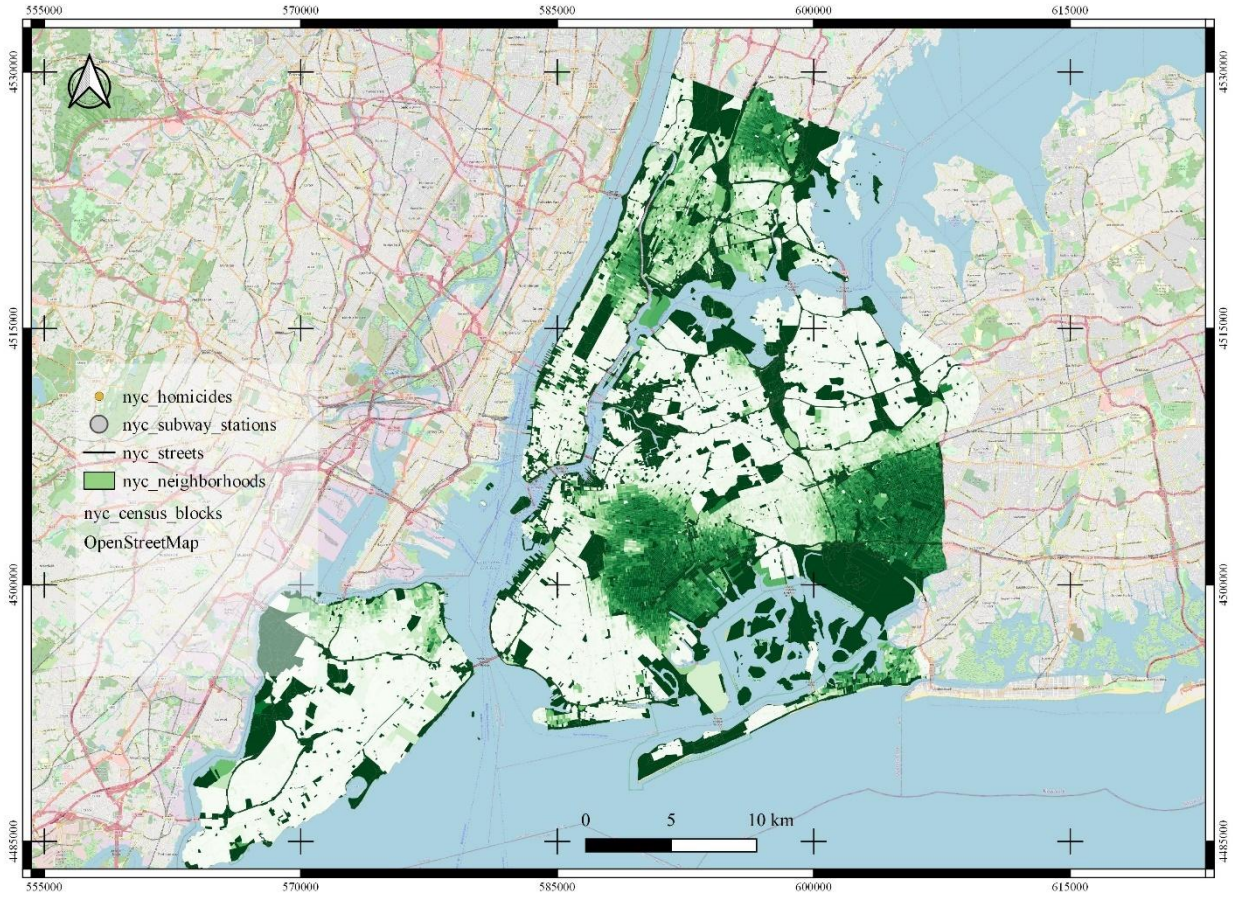

Рис. 22. Популяція темношкірого населення у відсотках від загальної чисельності населення

Завдання: використовуючи стандартні інструменти QGIS візуалізуйте наступну інформацію з БД згідно вимог попередньої роботи

Варіант	Завдання
1	Загальна кількість людей у блоці перепису населення <=5000 чол.
2	Загальна кількість людей у блоці перепису населення не дорівнює 0
3	Загальна кількість людей у блоці перепису населення від 2000 до 5000 чол.
4	Кількість людей, які самоідентифікують себе як "Білі" > 1000 чол.
5	Кількість людей, які самоідентифікують себе як "Темношкірі" від 1000 до 2500
6	Кількість людей, які самоідентифікують себе як "Індіанці" від 1 до 50
7	Кількість людей, які самоідентифікують себе як "Азіати" в блоці від 1 до 500
8	Кількість людей, які самостійно ідентифікують себе з іншими категоріями населення
9	Сумарну кількість людей, які самоідентифікують себе як "Азіати" та "Індіанці"
10	Сумарну кількість людей, які самоідентифікують себе як "Азіати" та "Темношкірі"

4.2. Таблиця nyc_neighborhoods

Нью-Йорк має багату історію назв околиць та суміжних територій. Мікрорайони – це соціальні конструктиви, які не слідують лініям, закладеним урядом. Наприклад, бруклінські райони Керролл-Гарденс, Ред-Хук і Коббл-Хілл колись були згальновідомі як «Південний Бруклін». І тепер, залежно від того, з ким Ви розмовляєте, ті ж чотири блоки в районі, раніше відомому як Ред-Хук, можна віднести до Колумбійських висот, Керролл-Гарденс Вест або Ред Хук.

Кількість записів: 129

Таблиця 2.

Опис таблиці даних **nyc_neighborhoods**

Назва стовпчика	**Опис даних**
name	Назва кварталу
boroname	Назва муніципалітету Нью-Йорка. Манхеттен, Бронкс, Бруклін, Стейтен-Айленд, Квінз.
geom	Межа полігону району

Рис. 23. Райони Нью-Йорку

Завдання: використовуючи стандартні інструменти QGIS візуалізуйте наступну інформацію з БД згідно вимог попередньої роботи а також виведіть відповідні підписи

Варіант	Завдання
1	Всі квартали муніципалітету Манхеттена
2	Всі квартали муніципалітету Бронкса
3	Всі квартали муніципалітету Брукліна
4	Всі квартали муніципалітету Стейтен-Айленд
5	Всі квартали муніципалітету Квінз
6	Всі квартали муніципалітетів що містять слово "Hill"
7	Всі квартали муніципалітетів що містять слово "Park"
8	Всі квартали муніципалітетів що містять слово "Heights"
9	Всі квартали муніципалітетів що містять принаймні одну літеру "w"
10	Всі квартали муніципалітетів що містять принаймні одну літеру "h"

4.3. Таблиця nyc_streets

Центральні лінії вулиць утворюють транспортну мережу міста. Ці вулиці були класифіковані за типами як магістралі, провулки, артеріальними вулицями, автостради і невеликими вулицями. Бажані райони для проживання можуть бути на житлових вулицях, а не поруч з автострадою.

Кількість записів: 19091

Таблиця 3.

Опис таблиці даних **nyc_streets**

Назва стовпчика	Опис даних
name	Назва вулиці
oneway	Вулиця в одну сторону? “yes” = так, “” = ні
type	Тип дороги (головна, другорядна, житлова, автомобільна тощо)
geom	Центральна лінія вулиці

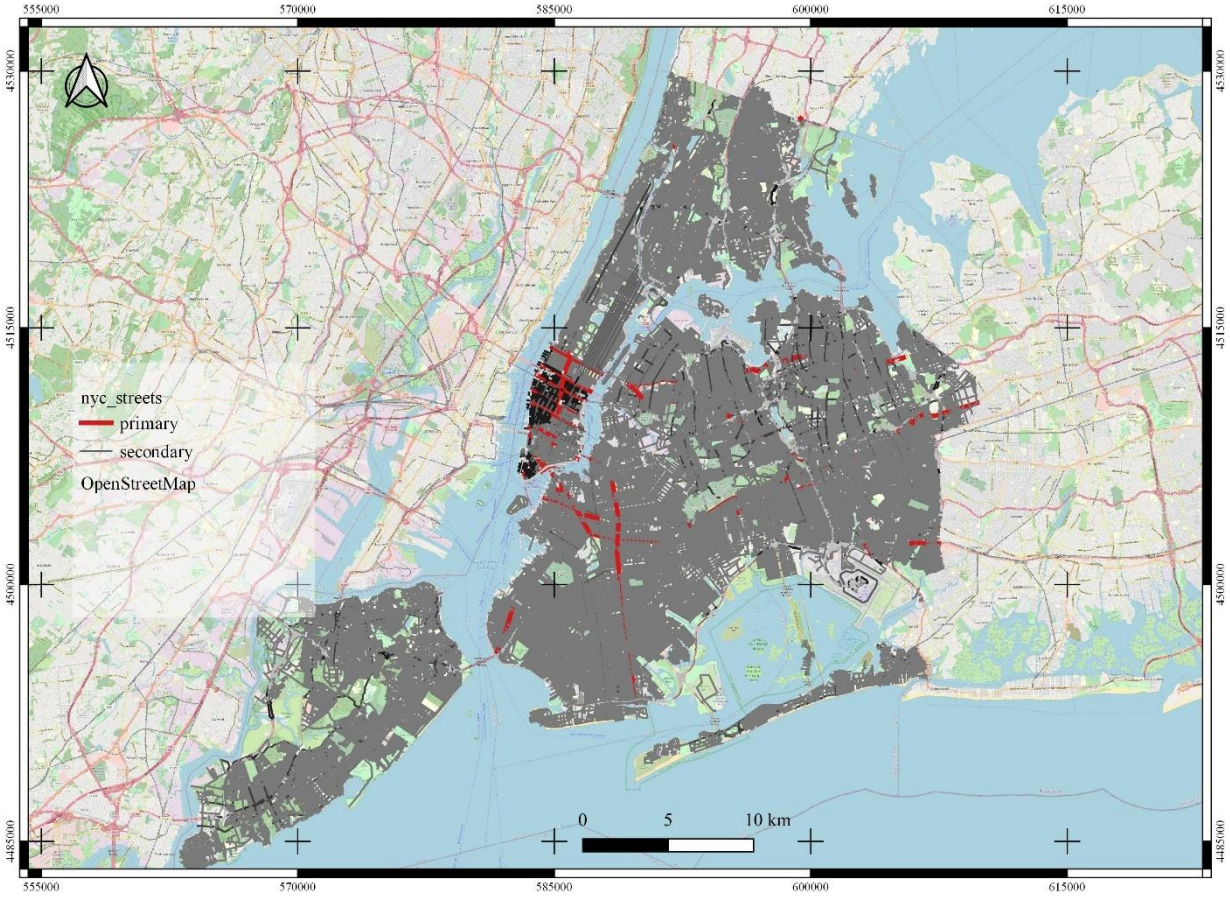

Рис. 22. Вулична мережа Нью-Йорка. Основні дороги в червоному кольорі.

Завдання: використовуючи стандартні інструменти QGIS візуалізуйте наступну інформацію з БД згідно вимог попередньої роботи а також виведіть відповідні підписи

Варіант	Завдання
1	Усі дороги тип яких – пішохідні (footway)
2	Усі дороги тип яких – для автомобілів (motorway)
3	Усі дороги тип яких – головні (primary)
4	Усі дороги тип яких – другорядні (secondary)
5	Усі дороги тип яких – не класифіковані (unclassified)
6	Усі односторонні дороги (oneway = 'yes')

7	Усі двохсторонні дороги (oneway = '')
8	Усі дороги у назві яких є "St" (вулиця)
9	Усі дороги у назві яких є "Ave" (авеню)
10	Усі дороги у назві яких є "Blvd" (бульвар)

4.4. Таблиця nyc_subway_stations

Станції метро пов'язують верхній світ, де люди живуть, з невидимою мережею метро під ним. Як портали до системи громадського транспорту, локації станцій допомагають визначити, наскільки легко різним людям зайти в систему метро.

Кількість записів: 491

Таблиця 4.

Опис таблиці даних **nyc_subway_stations**

name	Назва станції
borough	Назва муніципалітету Нью-Йорка. Манхеттен, Бронкс, Бруклін, Стейтен-Айленд, Квінз
routes	Лінії метро, які проходять через цю станцію
transfers	Лінії, на які Ви можете перейти через цю станцію
express	Станції, на яких зупиняються експреси, "express" = так, "" = ні
geom	Розташування станції
color	"Колір" гілки

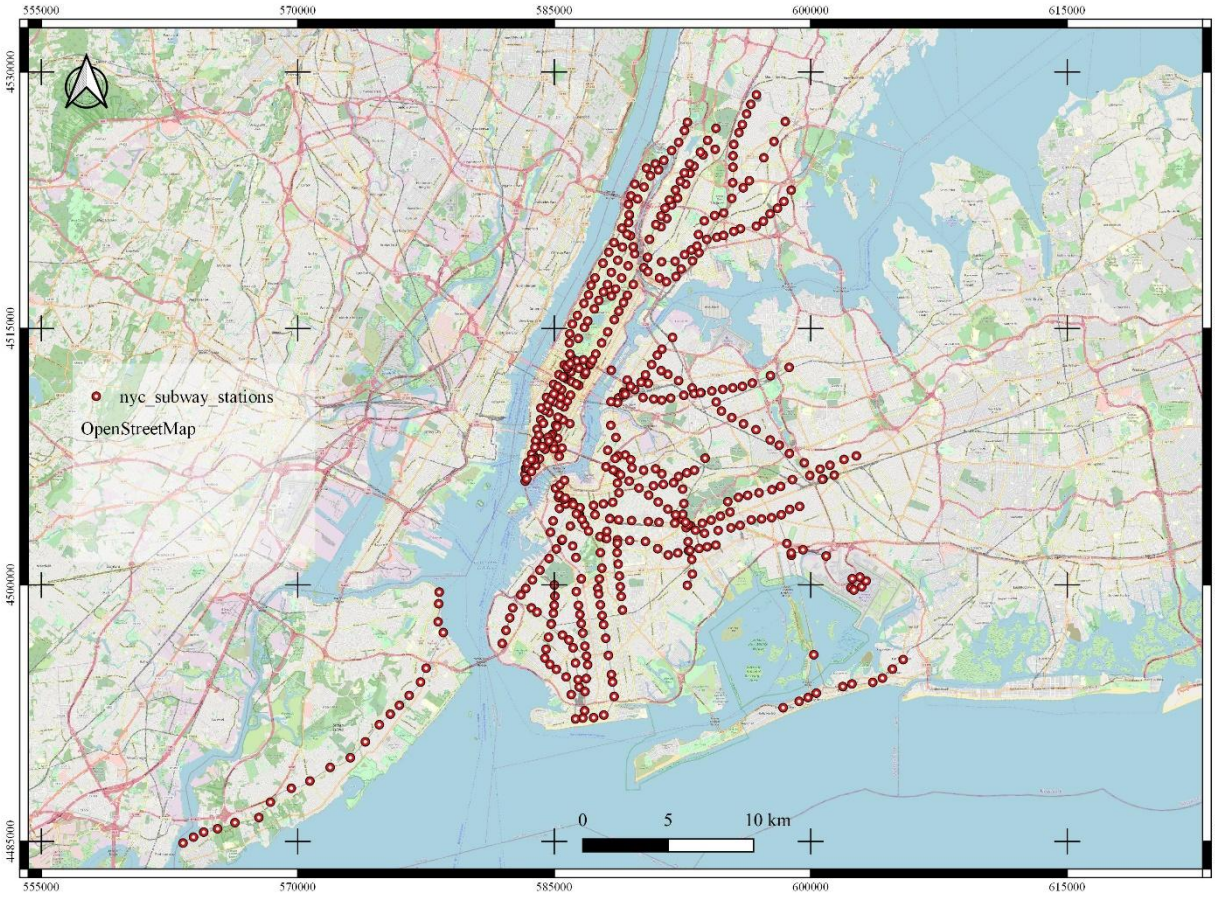

Рис. 23. Розташування станцій метро Нью-Йорка використовуючи стандартні інструменти QGIS візуалізуйте наступну інформацію з БД згідно вимог попередньої роботи а також виведіть відповідні підписи

Варіант	Завдання
1	Станції метро, що належать до будь якої комбінації "синьої" гілки
2	Станції метро, що належать до будь якої комбінації "коричневої" гілки
3	Станції метро, що належать до будь якої комбінації "зеленої" гілки
4	Станції метро, що належать до будь якої комбінації "сірої" гілки
5	Станції метро, що належать до будь якої комбінації "помаранчевої" гілки
6	Станції метро, що належать до будь якої комбінації "фіолетової" гілки
7	Станції метро, що належать до будь якої комбінації "червоної" гілки
8	Станції метро на яких можлива пересадка
9	Станції метро на яких не можлива пересадка
10	Станції, на яких зупиняються експреси

Практична робота №5. Мова запитів SQL

SQL, або " Structured Query Language (Мова структурованих запитів)", є засобом для створення запитів та оновлення даних в реляційних базах даних. Ви вже бачили SQL, коли ми створили нашу першу базу даних. Нагадаємо:

```
SELECT postgis_full_version();
```

Але це було питання про базу даних. Тепер, коли ми завантажили дані в нашу базу даних, давайте використовувати SQL, щоб створити запити до даних. Наприклад, за допомогою запиту SQL дамо відповідь на запитання "Які назВи всіх районів Нью-Йорка?".Для цього виконайте наступні дії:

1. Відкрийте редактор запитів з меню *Tools* > *Query Tool.* Введіть наступний запит в поле редактора запитів та натисніть кнопку **Execute/Refresh** або на клавіатурі **F5.**

```
SELECT name FROM nyc_neighborhoods;
```

Запит буде виконуватися протягом декількох (мілі)секунд і має повернути 129 результатів.

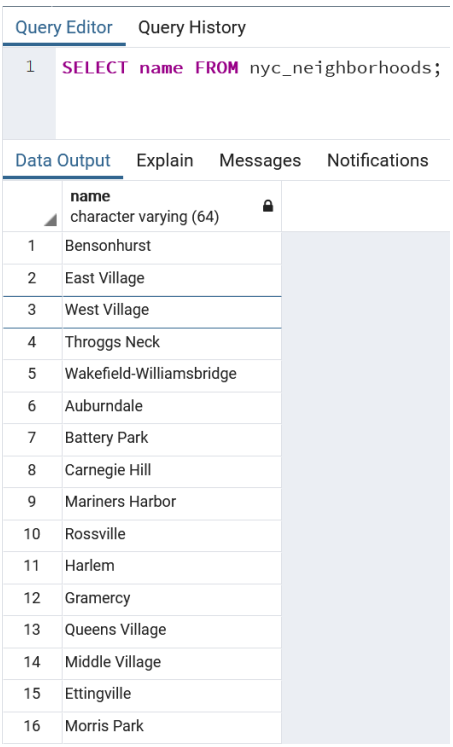

Рис. 24. Фрагмент результатів виконання запиту.

Для розуміння почнемо з чотирьох "дієслів" SQL,

1. *SELECT* – повертає рядки у відповідь на запит
2. *INSERT* – додає нові рядки до таблиці
3. *UPDATE* – змінює наявні рядки в таблиці
4. *DELETE* – видаляє рядки з таблиці

Ми будемо працювати практично виключно з SELECT для того, щоб створювати запити до таблиць з використанням просторових функцій.

5.1. Запити на вибірку

Зазвичай запит на вибірку має вигляд:

```
SELECT some_columns FROM some_data_source WHERE some_condition;
```

Some_columns є іменами стовпців або функціями значень стовпців. Some_data_source є або єдиною таблицею, або складеною таблицею, створеною шляхом об'єднання двох таблиць за ключем або умовою. Головна some_condition – це фільтр, який обмежує кількість рядків, які потрібно повернути.

Повертаємося до нашої таблиці nyc_neighborhoods з використанням фільтру. Таблиця містить всі райони в Нью-Йорку, але ми хочемо тільки ті, що в Брукліні.

```
SELECT name
  FROM nyc_neighborhoods
  WHERE boroname = 'Brooklyn';
```

Запит буде виконуватися ще менше (мілі) секунд і поверне 23 результати.

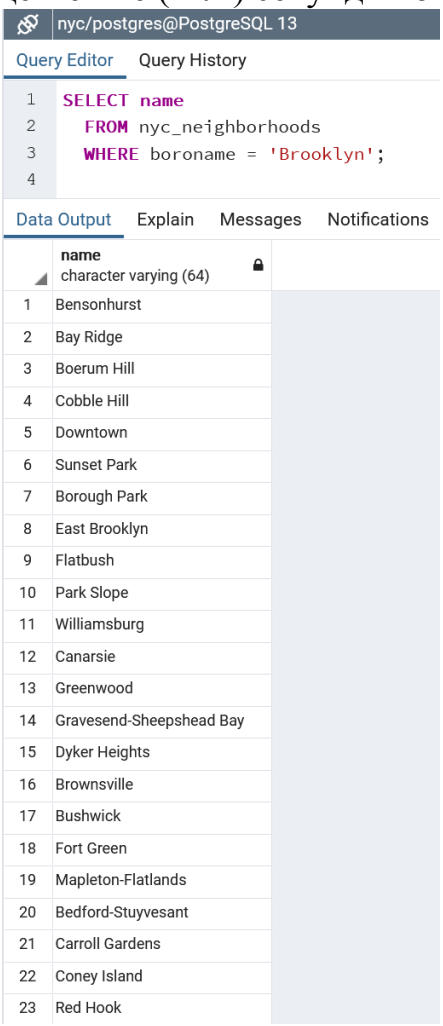

Рис. 25. Результат вибірки районів в Брукліні.

Іноді нам потрібно буде застосувати функцію до результатів нашого запиту. Наприклад, "Яка кількість букв в назвах всіх районів Брукліна?"

PostgreSQL має функцію довжини рядка, **char_length (string)**, якою ми й скористаємось.

```
SELECT char_length(name)
  FROM nyc_neighborhoods
  WHERE boroname = 'Brooklyn';
```

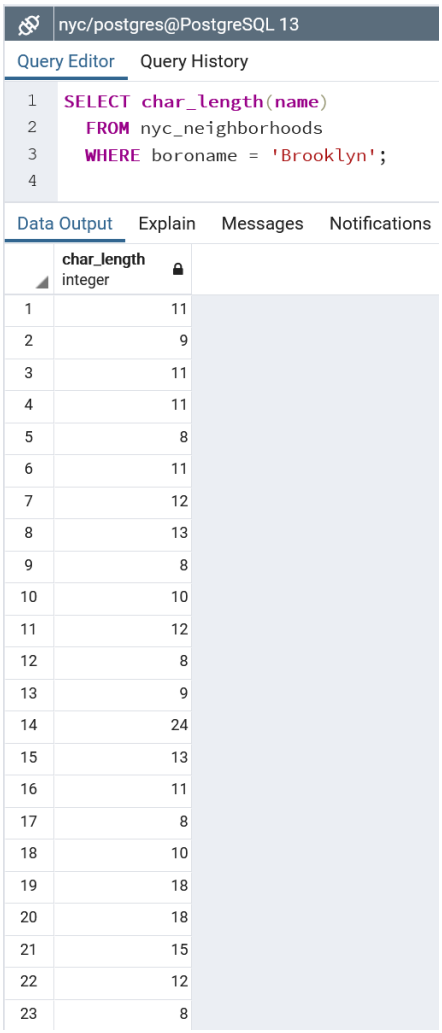

Рис. 26. Результат вибірки кількості символів в назвах.

Часто, існує менша зацікавленість в окремих рядах, ніж в загальній статистиці, яка відноситься до всіх з них. Таким чином, знання про довжини назв районів може бути менш цікавою, ніж знання про середню довжину імен. Функції, які описуються в декількох рядках і повертають один результат, називаються "агрегатні" функції.

PostgreSQL має ряд вбудованих агрегатних функцій, включаючи **avg() –** функцію середніх значень та **stddev() –** функцію стандартних відхилень.

Вирішимо наступне питання: "Яка середня кількість букв і стандартне відхилення кількості букв в назвах всіх мікрорайонів Брукліна?". Для цього створимо новий запит:

```
SELECT avg(char_length(name)), stddev(char_length(name))
  FROM nyc_neighborhoods
  WHERE boroname = 'Brooklyn';
```

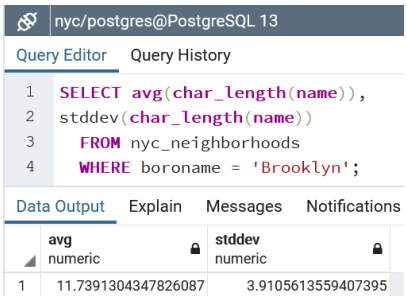

Рис. 27. Статистика середньої довжини назв та середнє відхилення

Агрегатні функції в нашому останньому прикладі були застосовані до кожного рядка в наборі результатів. Що робити, якщо ми хочемо, щоб вибірка здійснювалася над меншими групами в межах загального набору результатів? Для цього додаємо речення GROUP BY. Агрегатні функції часто потребують додавання оператору GROUP BY для групування результату, з одного або кількох стовпців.

Для прикладу знайдемо вирішення наступного питання: "Яка середня кількість букв в назвах всіх районів у боро Нью-Йорка?"

```
SELECT boroname, avg(char_length(name)), stddev(char_length(name))
  FROM nyc_neighborhoods
  GROUP BY boroname;
```

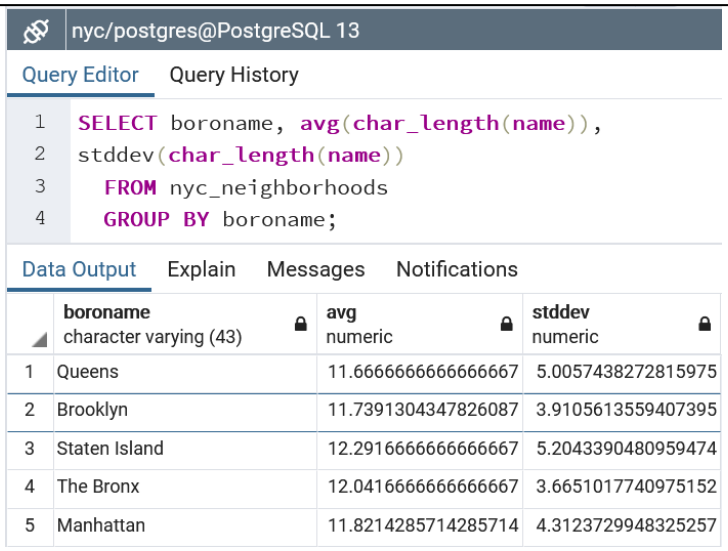

Рис. 28. Середня кількість букв в назвах всіх районів у боро Нью-Йорка

Ми включаємо стовпець boroname в результат виводу, щоб ми могли визначити, яка статистика застосовується до якого боро. У сукупному запиті можна виводити лише стовпці виводу, які є членами речення групування або агрегатних функцій.

Використані функції в практичній роботі:

avg(expression): Агрегатна функція PostgreSQL, яка повертає середнє значення числового стовпця.

char_length(string): Рядкові функції PostgreSQL, що повертають кількість символів у рядку.

stddev(expression): Агрегатна функція PostgreSQL, яка повертає стандартне відхилення вхідних значень.

count(expression): Агрегатна функція PostgreSQL, яка повертає кількість записів у наборі даних.

sum(expression): Агрегатна функція PostgreSQL, яка повертає суму записів у наборі даних.

Завдання: використовуючи таблицю nyc_census_blocks створіть запити та дайте відповідь на наступні питання по кожному з боро Нью-Йорка:

1	Яка загальна чисельність населення боро Манхеттен? Який відсоток населення вважає себе "Білим" населенням у боро Манхеттен?
2	Яка кількість людей, які самоідентифікують себе як "Темношкірі у боро Манхеттен? Яка загальна чисельність населення боро Бронкс?
3	Який відсоток населення вважає себе "Білим" населенням у боро Бронкс? Яка к кількість людей, які самоідентифікують себе як "Темношкірі" у боро Бронкс?
4	Яка загальна чисельність населення боро Бруклін? Який відсоток населення вважає себе "Білим" населенням у боро Бруклін?
5	Яка кількість людей, які самоідентифікують себе як "Індіанці" в блоці у боро Бруклін? Яка загальна чисельність населення боро Стейтен-Айленд?
6	Який відсоток населення вважає себе "Білим" населенням у боро Стейтен-Айленд? Яка кількість людей, які самоідентифікують себе як "Індіанці" у боро Стейтен-Айленд?
7	Яка загальна чисельність населення боро Квінз? Який відсоток населення вважає себе "Білим" населенням у боро Квінз?
8	Яка загальна чисельність населення м. Нью-Йорк, що вважає себе "Білим" населенням? Яка сумарна чисельність населення, що проживає в боро Стейтен-Айленд та Квінз?
9	Який відсотковий склад "Білого" населення для кожного боро? Яка сумарна чисельність населення, що проживає в боро Бруклін та Квінз?

10	Яка чисельність населення, що вважає себе "Білим" населенням у боро Бруклін? Яка сумарна чисельність населення, що проживає в боро Бруклін та Манхеттен?

Приклад виконання:

"Яке населення міста Нью-Йорк?"

Запит:

```
SELECT Sum(popn_total) AS population
  FROM nyc_census_blocks;
```

Відповідь: 8175032

5.2. Таблиця nyc_census_sociodata

В даній таблиці міститься великий набір соціально-економічних даних, зібраних в процесі перепису населення, але на великому рівні. Блоки перепису об'єднуються для формування переписних районів (і кварталів). Тут зібрано деякі соціально-економічні дані на рівні перепису районів, щоб дати відповідь на деякі питання про Нью-Йорк.

Файл nyc_census_sociodata є таблицею з даними. Нам потрібно буде підключити її до географії перепису перед проведенням будь-якого просторового аналізу.

Таблиця 4.

Опис таблиці даних **nyc_census_sociodata**

tractid	11-значний код, який однозначно ідентифікує кожен об'єкт. ("36005000100")
transit_total	Кількість працівників у районі
transit_private	Кількість робітників в районі, які користуються приватними автомобілями / мотоциклами
transit_public	Кількість працівників у районі, які користуються громадським транспортом
transit_walk	Кількість працівників у районі, які ходять пішки
transit_other	Кількість працівників у районі, які використовують інші форми, такі як ходьба / велосипед
transit_none	Кількість працівників у районі, які працюють з дому
transit_time_mins	Загальна кількість хвилин, витрачених на транзит усіма працівниками у районі (хвилини)
family_count	Кількість сімей у районі
family_income_median	Медіана доходу сім'ї у районі в доларах
family_income_mean	Середній дохід сім'ї у районі в доларах

family_income_aggregate	Загальний дохід усіх сімей у районі (доларах)
edu_total	Кількість людей з освітньою історією
edu_no_highschool_dipl	Кількість людей, які не мають диплома про середню освіту
edu_highschool_dipl	Кількість людей з дипломом про середню освіту і без подальшої освіти
edu_college_dipl	Кількість людей з дипломом коледжу і без подальшої освіти
edu_graduate_dipl	Кількість людей які закінчили вищу освіту

Завдання: використовуючи стандартні інструменти QGIS візуалізуйте інформацію з будь якого поля на Ваш вибір.

Варіанти правильних відповідей

1	Яка загальна чисельність населення боро Манхеттен? SELECT Sum(popn_total) AS population FROM nyc_census_blocks WHERE boroname = 'Manhattan'; 1585873 Який відсоток населення вважає себе "Білим" населенням у боро Манхеттен? SELECT 100 * (sum(popn_white)/sum(popn_total)) FROM nyc_census_blocks WHERE boroname = 'Manhattan'; 57.44930394804628
2	Яка кількість людей, які самоідентифікують себе як "Темношкірі у боро Манхеттен? SELECT 100 * (sum(popn_black)/sum(popn_total)) FROM nyc_census_blocks WHERE boroname = 'Manhattan'; 15.555280908370342 Яка загальна чисельність населення боро Бронкс? SELECT Sum(popn_total) AS population FROM nyc_census_blocks WHERE boroname = 'The Bronx'; 1385108
3	Який відсоток населення вважає себе "Білим" населенням у боро Бронкс? SELECT 100 * (sum(popn_white)/sum(popn_total)) FROM nyc_census_blocks WHERE boroname = 'The Bronx'; 27.903744689944755 Яка кількість людей, які самоідентифікують себе як "Темношкірі" у боро Бронкс? SELECT 100 * (sum(popn_black)/sum(popn_total)) FROM nyc_census_blocks WHERE boroname = 'The Bronx'; 36.47369013824193
4	Яка загальна чисельність населення боро Бруклін? SELECT Sum(popn_total) AS population FROM nyc_census_blocks WHERE boroname = 'Brooklyn'; 2504700 Який відсоток населення вважає себе "Білим" населенням у боро Бруклін? SELECT 100 * (sum(popn_white)/sum(popn_total)) FROM nyc_census_blocks WHERE boroname = 'Brooklyn'; 42.80117379326865

5	Яка кількість людей, які самоідентифікують себе як "Індіанці" в блоці у боро Бруклін? SELECT 100 * (sum(popn_nativ)/sum(popn_total)) FROM nyc_census_blocks WHERE boroname = 'Brooklyn'; 0.5399449035812672 Яка загальна чисельність населення боро Стейтен-Айленд?
6	Який відсоток населення вважає себе "Білим" населенням у боро Стейтен-Айленд? SELECT 100 * (sum(popn_white)/sum(popn_total)) FROM nyc_census_blocks WHERE boroname = 'Staten Island'; 72.89420348601541 Яка кількість людей, які самоідентифікують себе як "Індіанці" у боро Стейтен-Айленд? SELECT 100 * (sum(popn_nativ)/sum(popn_total)) FROM nyc_census_blocks WHERE boroname = 'Staten Island'; 0.36161542892496745
7	Яка загальна чисельність населення боро Квінз? SELECT Sum(popn_total) AS population FROM nyc_census_blocks WHERE boroname = 'Queens'; 2230621 Який відсоток населення вважає себе "Білим" населенням у боро Квінз? SELECT 100 * (sum(popn_white)/sum(popn_total)) FROM nyc_census_blocks WHERE boroname = 'Queens'; 39.72207739459102
8	Яка загальна чисельність населення м. Нью-Йорк, що вважає себе "Білим" населенням? SELECT sum(popn_white) FROM nyc_census_blocks; 3597337 Яка кількість записів в таблиці що відносяться до боро Стейтен-Айленд? SELECT count(blkid) FROM nyc_census_blocks WHERE boroname = 'Staten Island'; 5037
9	Який відсотковий склад "Білого" населення для кожного боро? SELECT boroname, 100 * Sum(popn_white)/Sum(popn_total) AS white_pct FROM nyc_census_blocks GROUP BY boroname; boroname | white_pct

	---------------+------------------ Brooklyn | 42.8011737932687 Manhattan | 57.4493039480463 The Bronx | 27.9037446899448 Queens | 39.722077394591 Staten Island | 72.8942034860154 Яка кількість записів в таблиці що відносяться до боро Бруклін? SELECT count(blkid) FROM nyc_census_blocks WHERE boroname = 'Brooklyn'; 9682
10	Яка чисельність населення, що вважає себе "Білим" населенням у боро Бруклін? Яка кількість записів в таблиці що відносяться до боро Манхеттен? SELECT count(blkid) FROM nyc_census_blocks WHERE boroname = 'Manhattan'; 3856

Назва стовпчика	Опис даних
blkid	15-значний код, який однозначно ідентифікує кожен блок перепису. Наприклад: 360050001009000
popn_total	Загальна кількість людей у блоці перепису населення
popn_white	Кількість людей, які самоідентифікують себе як "Білі" в блоці
popn_black	Кількість людей, які самоідентифікують себе як "Темношкірі" у блоці
popn_nativ	Кількість людей, які самоідентифікують себе як "Індіанці" в блоці
popn_asian	Кількість людей, які самоідентифікують себе як "Азіати" в блоці
popn_other	Кількість людей, які самостійно ідентифікують себе з іншими категоріями в блоці
boroname	боро Нью-Йорка. Манхеттен, Бронкс, Бруклін, Стейтен-Айленд, Квінз

Практична робота №6. Робота з геометрією об'єктів у PostGIS

6.1. Вступ

У попередніх практичних роботах ми завантажили до БД найрізноманітніші дані. Перш ніж ми почнемо працювати з нашими даними, спробуємо виконати для ознайомлення деякі прості приклади.

У pgAdmin, ще раз виберіть базу даних **nyc** і відкрийте інструмент запиту SQL. Вставте цей приклад SQL-коду у вікно редактора *Query Tool* (видаліть будь-який текст, який може бути там за замовчуванням), а потім виконайте команду.

```
CREATE TABLE geometries (name varchar, geom geometry);

INSERT INTO geometries VALUES
('Point', 'POINT(0 0)'),
('Linestring', 'LINESTRING(0 0, 1 1, 2 1, 2 2)'),
('Polygon', 'POLYGON((0 0, 1 0, 1 1, 0 1, 0 0))'),
('PolygonWithHole', 'POLYGON((0 0, 10 0, 10 10, 0 10, 0 0),(1 1, 1 2, 2 2, 2 1, 1 1))'),
('Collection', 'GEOMETRYCOLLECTION(POINT(2 0),POLYGON((0 0, 1 0, 1 1, 0 1, 0 0)))');

SELECT name, ST_AsText(geom) FROM geometries;
```

У наведеному вище прикладі створюється (**CREATE**) таблиця **geometries** (геометрія), потім додається (**INSERT**) п'ять геометрій: точка, лінія, багатокутник, багатокутник з отвором і колекція. Нарешті, вставлені рядки вибираються (**SELECT**) і відображаються на панелі виводу. В результаті отримаємо вивід у вікні *Data Output*

Data Output | Explain | Messages | Notifications

	name character varying	st_astext text
1	Point	POINT(0 0)
2	Linestring	LINESTRING(0 0,1 1,2 1,2 2)
3	Polygon	POLYGON((0 0,1 0,1 1,0 1,0 0))
4	PolygonWithHole	POLYGON((0 0,10 0,10 10,0 10,0 0),(1 1,1 2,2 2,2 1,1 1))
5	Collection	GEOMETRYCOLLECTION(POINT(2 0),POLYGON((0 0,1 0,1 1,0 1,0 0)))

Рис. 29. Результат виконання запиту

6.2. Таблиці метаданих

Відповідно до специфікації Simple Features for SQL (SFSQL), PostGIS надає дві таблиці для відстеження та звітування про типи геометрії, доступні в даній базі даних.

- Перша таблиця, spatial_ref_sys, визначає всі просторові референцні системи, відомі базі даних, і буде більш детально описана пізніше.
- Друга таблиця (власне, вид), geometry_columns, містить перелік усіх "особливостей" (визначених як об'єкт з геометричними атрибутами), а також основні деталі цих особливостей.

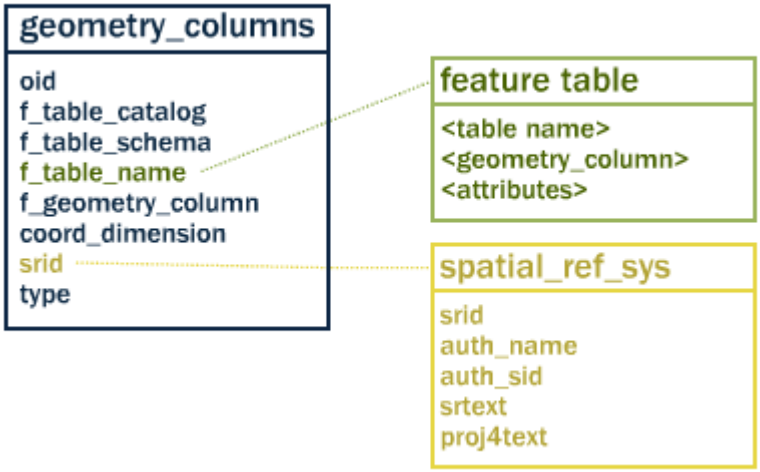

Рис. 30. Відношення в таблицях

Давайте подивимося на таблицю geometry_columns нашій базі даних. Вставте наступний запит у вікно *Query Tool*, як і раніше:

```
SELECT * FROM geometry_columns;
```

Data Output | Explain | Messages | Notifications

	f_table_catalog character varying (256)	f_table_schema name	f_table_name name	f_geometry_column name	coord_dimension integer	srid integer	type character varying (30)
1	nyc	public	nyc_census_blocks	geom	2	26918	MULTIPOLYGON
2	nyc	public	nyc_homicides	geom	2	26918	POINT
3	nyc	public	nyc_neighborhoods	geom	2	26918	MULTIPOLYGON
4	nyc	public	nyc_streets	geom	2	26918	MULTILINESTRING
5	nyc	public	nyc_subway_stations	geom	2	26918	POINT
6	nyc	public	nyc_census_blocks...	geom	2	26918	MULTIPOLYGON
7	nyc	public	geometries	geom	2	0	GEOMETRY

Рис. 31. Результат обробки запиту

Структура таблиці geometry_columns наступна

- f_table_catalog, f_table_schema, та f_table_name визначає повне ім'я таблиці, що містить задану геометрію. Оскільки PostgreSQL не використовує каталоги, f_table_catalog, як правило, порожній.
- f_geometry_column – ім'я стовпця геометрії в таблиці об'єктів. Для таблиць геометрії з декількома стовпцями, буде по одному запису для кожного з таких.

- `coord_dimension` і `crid` визначають розмірність геометрії (2-, 3- або 4-вимірної) та ідентифікатор системи координат, що використовується для геометрії в цій таблиці (зовнішній ключ до таблиці `SPATIAL_REF_SYS`) відповідно.
- Стовпчик `type` визначає тип просторового об'єкта.

Створюючи запит до цієї таблицю, клієнти та бібліотеки ГІС можуть визначити, чого очікувати при отриманні даних, і можуть виконувати будь-яку необхідну проекцію, обробку або рендерінг без необхідності перевірки кожної геометрії.

6.3. Представлення об'єктів реального світу

Специфікація Simple Features for SQL (SFSQL) – оригінальний стандарт для розробки PostGIS, визначає, як відображаються об'єкт реального світу. Обираючи безперервну форму і оцифруючи її при фіксованій роздільній здатності, ми досягаємо прохідного представлення об'єкта. SFSQL визначає лише 2-вимірні представлення. PostGIS розширив даний стандарт до 3-х та 4-х вимірних відображення. Лише зовсім недавно специфікація SQL-Multimedia Part 3 (SQL/MM) офіційно визначила такі представлення.

Таблиця наших прикладів містить суміш різних типів геометрії. Ми можемо збирати загальну інформацію про кожен об'єкт за допомогою функцій, що зчитує метадані геометрії.

- **ST_GeometryType(geometry)** повертає тип геометрії
- **ST_NDims(geometry)** повертає кількість вимірів геометрії
- **ST_SRID(geometry)** повертає номер ідентифікатора просторової референцної системи геометрії

```
SELECT name, ST_GeometryType(geom), ST_NDims(geom), ST_SRID(geom)
 FROM geometries;
```

Data Output | Explain | Messages | Notifications

	name character varying	st_geometrytype text	st_ndims smallint	st_srid integer
1	Point	ST_Point	2	0
2	Linestring	ST_LineString	2	0
3	Polygon	ST_Polygon	2	0
4	PolygonWithHole	ST_Polygon	2	0
5	Collection	ST_GeometryCollection	2	0

Рис. 31. Результат виконання запиту

6.4. Точкові об'єкти (Points)

Просторова **точка** являє собою єдине місце на Землі. Ця точка представлена єдиною координатою (включаючи або 2-, або 3- або 4-виміри). Точки використовуються для представлення об'єктів, коли точні деталі, такі як форма та розмір, не важливі в цільовій шкалі. Наприклад, міста на карті світу можна охарактеризувати як точки, в той час як карта однієї держаВи може представляти міста як полігони.

	
Точка	Мультиточка з 4 частинами

Рис.32 Точка та мультиточка

```
SELECT ST_AsText(geom)
  FROM geometries
  WHERE name = 'Point';
```

	st_astext text
1	POINT(0 0)

Рис. 33. Результат виконання запиту

Деякі з конкретних просторових функцій для роботи з точками:

1. **ST_X(geometry)** повертає координату X
2. **ST_Y(geometry)** повертає координату Y

Отже, ми можемо читати координати точки:

```
SELECT ST_X(geom), ST_Y(geom)
  FROM geometries
  WHERE name = 'Point';
```

Таблиця станцій метро Нью-Йорка (nyc_subway_stations) - це набір даних, представлений як точки. Наступний запит SQL поверне геометрію, пов'язану з однією точкою (у стовпці **ST_AsText**).

```
SELECT name, ST_AsText(geom)
  FROM nyc_subway_stations
  LIMIT 1;
```

Data Output | Explain | Messages | Notifications

	name character varying (31)	st_astext text
1	Cortlandt St	POINT(583521.854408956 4507077.862599085)

Рис. 34. Результат виконання запиту

6.5. Лінійні рядки (Linestrings)

Лінійні рядки - це шлях між місцями. Він приймає форму впорядкованої серії з двох або більше точок. Дороги і річки, як правило, представлені як лінії. Кажуть, що лінійний рядок закривається, якщо він починається і закінчується на тій же точці. Кажуть, що він простий, якщо він не перетинається або не торкається себе (за винятком кінцевих точок, якщо він закритий). Рядок може бути як закритим, так і простим.

	
Простий не закритий лінійний рядок	Простий мультилнійний рядок із 4 кінцевими точками та 2 об'єктами

Рис. 35. Типи лінійних рядків

Вулична мережа для Нью-Йорка (nyc_streets) була завантажена раніше в попередніх роботах. Цей набір даних містить такі відомості, як ім'я та тип. Одна вулиця реального світу може складатися з безлічі ліній, кожна з яких представляє відрізок дороги з різними атрибутами.

Наступний запит SQL поверне геометрію, пов'язану з одним рядком (у стовпці ST_AsText).

```
SELECT ST_AsText(geom)
  FROM geometries
  WHERE name = 'Linestring';
```

Data Output | Explain | Messa

	st_astext text
1	LINESTRING(0 0,1 1,2 1,2 2)

Рис. 36. Результат виконання запиту

Деякі з конкретних просторових функцій для роботи з лінійними рядками:

- **ST_Length(geometry)** повертає довжину лінії
- **ST_StartPoint(geometry)** повертає першу координату як точку
- **ST_EndPoint(geometry)** повертає останню координату як точку

- **ST_NPoints(geometry)** повертає кількість координат у рядок лінії
  Отже, довжина нашого лінійного рядка становить: 3.41421356237309

```
SELECT ST_AsText(geom)
  FROM geometries
  WHERE name = 'Linestring';
```

6.6. Полігони (Polygons)

Полігон - це представлення області. Зовнішня межа полігона представлена кільцем. Це кільце є лінійним рядком (Linestring), який є одночасно закритим і простим, як визначено вище. Отвори всередині багатокутника також представлені кільцями.

Полігони використовуються для представлення об'єктів, розмір і форма яких важливі. Міські межі, парки, контури будівель або водойми зазвичай представлені як полігони, коли масштаб досить великий, щоб побачити їх площу. Дороги і річки іноді можуть бути представлені як полігони.

Полігон представлений зовнішнім кільцем

Мультиполігон, що містить 2 елементи котрі представлені зовнішніми та 3 внутрішніми кільцями

Рис. 37. Типи полігонів

Наступний запит SQL поверне геометрію, пов'язану з одним багатокутником (у стовпці **ST_AsText**).

```
SELECT ST_AsText(geom)
  FROM geometries
  WHERE name LIKE 'Polygon%';
```

*Замість того, щоб використовувати знак = у нашому реченні WHERE, ми використовуємо оператор LIKE для виконання операції зіставлення рядків. **Ви можете використовуватися до символу "*" для відповідності шаблону, але в SQL використовується символ "%'**, а також оператор LIKE, щоб сказати системі робити підстановку.

Data Output | Explain | Messages | Notifications

	st_astext text
1	POLYGON((0 0,1 0,1 1,0 1,0 0))
2	POLYGON((0 0,10 0,10 10,0 10,0 0),(1 1,1 2,2 2,2 1,1 1))

Рис. 38. Результат виконання запиту

Перший багатокутник має тільки одне кільце. Другий має внутрішню «дірку». Більшість графічних систем включають поняття «багатокутник», але ГІС-системи є відносно унікальними, дозволяючи багатокутникам явно мати отвори.

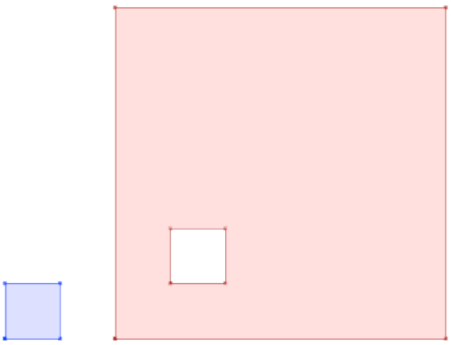

Рис. 39. Типи полігонів.

Деякі специфічні просторові функції для роботи з багатокутниками:

- **ST_Area(geometry)** повертає площу багатокутників
- **ST_NRings(geometry)** повертає кількість кілець (зазвичай 1, більше, якщо є отвори)
- **ST_ExteriorRing(geometry)** повертає зовнішнє кільце у вигляді лінії
- **ST_InteriorRingN(geometry,n)** повертає вказане внутрішнє кільце як рядок
- **ST_Perimeter(geometry)** повертає довжину всіх кілець

Розрахувати площу наших багатокутників можна за допомогою функції площі:

```
SELECT name, ST_Area(geom)
  FROM geometries
  WHERE name LIKE 'Polygon%';
```

	name character varying	st_area double precision
1	Polygon	1
2	PolygonWithHole	99

Рис. 40. Результат виконання запиту

Зверніть увагу, що багатокутник з отвором має площу, яка є площею зовнішньої оболонки (квадрат 10x10 квадрата) мінус площа отвору (квадрат 1x1).

6.7. Колекції

Існує чотири типи колекцій, які групують кілька простих геометрій у множини.

- **MultiPoint**, колекція точок.
- **MultiLineString**, колекція лінійних рядків
- **MultiPolygon**, колекція багатокутників

- **GeometryCollection**, неоднорідна колекція будь-якої геометрії (включаючи інші колекції)

6.8. Колекції

Колекції - це ще одна концепція, яка з'являється в програмному забезпеченні ГІС більше, ніж у загальному графічному програмному забезпеченні. Вони корисні для безпосереднього моделювання об'єктів реального світу як просторових об'єктів.

Наша набор даних містить багатокутник і точку:

```
SELECT name, ST_AsText(geom)
  FROM geometries
  WHERE name = 'Collection';
```

	name character varying	st_astext text
1	Collection	GEOMETRYCOLLECTION(POINT(2 0),POLYGON((0 0,1 0,1 1,0 1,0 0)))

Рис. 41. Результат виконання запиту

Рис. 42. Візуалізація запиту

Деякі з корисних просторових функцій для роботи з колекціями:

- **ST_NumGeometries(geometry)** повертає кількість деталей у колекції
- **ST_GeometryN(geometry,n)** повертає вказану частину
- **ST_Area(geometry)** повертає загальну площу всіх полігональних частин
- **ST_Length(geometry)** повертає загальну довжину всіх лінійних частин

6.9. Введення та виведення геометрії

У базі даних геометрія зберігається на диску у форматі, який використовується тільки програмою PostGIS. Для того щоб зовнішні програми вставляли і отримували корисні геометрії, їх потрібно перетворити в формат, який можуть зрозуміти інші додатки. На щастя, PostGIS підтримує передавання та отримання геометрії у великій кількості форматів:

Well-known text (WKT)

- **ST_GeomFromText(text, srid)** повертає `geometry`
- **ST_AsText(geometry)** повертає `text`
- **ST_AsEWKT(geometry)** повертає `text`

Well-known binary (WKB)

- **ST_GeomFromWKB(bytea)** повертає `geometry`
- **ST_AsBinary(geometry)** повертає `bytea`
- **ST_AsEWKB(geometry)** повертає `bytea`

Geographic Mark-up Language (GML)

- **ST_GeomFromGML(text)** повертає `geometry`
- **ST_AsGML(geometry)** повертає `text`

Keyhole Mark-up Language (KML)

- **ST_GeomFromKML(text)** повертає `geometry`
- **ST_AsKML(geometry)** повертає `text`

GeoJSON

- **ST_AsGeoJSON(geometry)** повертає `text`
  Scalable Vector Graphics (SVG)
- **ST_AsSVG(geometry)** повертає `text`

Найбільш поширеним використанням конструктора є перетворення текстового представлення геометрії у внутрішнє представлення:

Зауважте, що на додаток до текстового параметра з представленням геометрії, ми також маємо числовий параметр, що забезпечує SRID геометрії.

У наведеному нижче SQL-запиті показано приклад представлення WKB (для перетворення бінарного виводу для друку потрібен виклик encode(), потрібний виклик двійкового виводу у форматі ASCII):

```
SELECT encode(
  ST_AsBinary(ST_GeometryFromText('LINESTRING(0 0,1 0)')),
  'hex');
```

	encode text
1	0102000000020000000000000000000000000000000000000000000000000000f03f0000000000000000

Рис. 43. Результат виконання запиту

Для цілей наших лабораторних робіт ми продовжимо використовувати WKT, щоб переконатися, що Ви можете читати та розуміти геометрію, яку ми переглядаємо. Однак більшість фактичних процесів, таких як перегляд даних у ГІС-застосунку, передача даних до веб-служби або віддалена обробка даних, WKB є форматом вибору.

Оскільки WKT і WKB були визначені в специфікації SFSQL, вони не обробляють 3- або 4-мірні геометрії. Для цих випадків PostGIS визначив розширений добре відомий текст (EWKT) і розширені добре відомі двійкові (EWKB) формати. Вони забезпечують ті ж можливості форматування WKT і WKB з додатковою розмірністю.

Ось приклад 3D-лінії в WKT:

```
SELECT ST_AsText(ST_GeometryFromText('LINESTRING(0 0 0,1 0 0,1 1 2)'));
```

	st_astext text
1	LINESTRING Z (0 0 0,1 0 0,1 1 2)

Рис. 44. Результат виконання запиту

Зверніть увагу, що текстове представлення змінюється. Це тому, що рутине введення тексту для PostGIS є ліберальним в тому, що вона потребує. Він можу отримувати:

- шістнадцятково закодований EWKB;
- розширений відомий текст;
- Стандарт ISO відомий текст.

На вихідній стороні **ST_AsText** функція консервативна, і лише може отримувати стандартний ISO текст.

Крім **функції ST_GeometryFromText**, існує безліч інших способів створення геометрії з WKT або подібних відформатованих входів:

```
-- Використання ST_GeomFromText з параметром SRID
SELECT ST_GeomFromText('POINT(2 2)',4326);

-- Використання ST_GeomFromText без параметра SRID
SELECT ST_SetSRID(ST_GeomFromText('POINT(2 2)'),4326);

-- Використання функції ST_Make*
SELECT ST_SetSRID(ST_MakePoint(2, 2), 4326);

-- Використання синтаксису злиття PostgreSQL та ISO WKT
SELECT ST_SetSRID('POINT(2 2)'::geometry, 4326);

-- Використання синтаксису лиття PostgreSQL та розширеного WKT
SELECT 'SRID=4326;POINT(2 2)'::geometry;
```

6.10. Введення з тексту (Text)

Рядки WKT, які ми бачили до сих пір, були типу "текст", і ми перетворюємо їх на введення "геометрії" за допомогою функцій PostGIS, таких як **ST_GeomFromText()**.

PostgreSQL включає короткий синтаксис форми, який дозволяє конвертувати дані з одного типу в інший, синтаксис лиття, *oldata::newtype*. Так, наприклад, цей SQL перетворює подвійний в текстовий рядок.

```
SELECT 0.9::text;
```

Менш тривіально, цей SQL перетворює рядок WKT в геометрію:

```
SELECT 'POINT(0 0)'::geometry;
```

Одна річ, яку слід зазначити про використання лиття для створення геометрії: якщо Ви не вкажете SRID, Ви отримаєте геометрію з невідомою SRID. Ви можете вказати SRID за допомогою «розширеної» відомої текстової форми, яка включає блок SRID спереду:

```
SELECT 'SRID=4326;POINT(0 0)'::geometry;
```

Дуже часто використовують кастинг нотацію при роботі з WKT, а також як стовпці *geometry* та *geography*.

Використані функції в практичній роботі:

- **sum(expression)** повернути суму для набору записів
- **count(expression)** повертає розмір набору записів
- **ST_GeometryType(geometry)** повертає тип геометрії
- **ST_NDims(geometry)** повертає кількість вимірів геометрії
- **ST_SRID(geometry)** повертає номер просторового ідентифікатора посилання геометрії
- **ST_X(point)** повертає координату X

- **ST_Y(point)** повертає координату Y
- **ST_Length(linestring)** повертає довжину лінії
- **ST_StartPoint(geometry)** повертає першу координату як точку
- **ST_EndPoint(geometry)** повертає останню координату як точку
- **ST_NPoints(geometry)** повертає кількість координат у рядок лінії
- **ST_Area(geometry)** повертає площу багатокутників
- **ST_NRings(geometry)** повертає кількість кілець (зазвичай 1, більше, якщо є отвори)
- **ST_ExteriorRing(polygon)** повертає зовнішнє кільце у вигляді лінії
- **ST_InteriorRingN(polygon, integer)** повертає вказане внутрішнє кільце як рядок
- **ST_Perimeter(geometry)** повертає довжину всіх кілець
- **ST_NumGeometries(multi/geomcollection)** повертає кількість деталей у колекції
- **ST_GeometryN(geometry, integer)** повертає вказану частину колекції
- **ST_GeomFromText(text)** повертає geometry
- **ST_AsText(geometry)** повертає WKT text
- **ST_AsEWKT(geometry)** повертає EWKT text
- **ST_GeomFromWKB(bytea)** повертає geometry
- **ST_AsBinary(geometry)** повертає WKB bytea
- **ST_AsEWKB(geometry)** повертає EWKB bytea
- **ST_GeomFromGML(text)** повертає geometry
- **ST_AsGML(geometry)** повертає GML text
- **ST_GeomFromKML(text)** повертає geometry
- **ST_AsKML(geometry)** повертає KML text
- **ST_AsGeoJSON(geometry)** повертає JSON text
- **ST_AsSVG(geometry)** повертає SVG text

Також пам'ятайте таблиці, які у нас є:

- nyc_census_blocks
  - blkid, popn_total, boroname, geom
- nyc_streets
  - name, type, geom
- nyc_subway_stations
  - name, geom
- nyc_neighborhoods
  - name, boroname, geom

Завдання: використовуючи таблиці БД створіть запити, щоб дати відповідь на запитання:

1. Яка площа району (neighborhood) ‘West Village’?
2. Який тип геометрії вулиці "Pelham St"? Довжина?
3. Як репрезентується у форматі GeoJSON станція метро "Broad St"?
4. Яка загальна довжина вулиць (у кілометрах) в Нью-Йорку?
5. Яка площа Manhattan в гектарах?
6. Яка станція метро найбільш західна?

7. Яка довжина ‘Columbus Cir’ (також відома як Columbus Circle)?
8. Яка довжина вулиць в Нью-Йорку, підсумована за типом?

Варіанти правильних відповідей

1	SELECT ST_Area(geom) FROM nyc_neighborhoods WHERE name = 'West Village'; 1044614.5296486
2	SELECT ST_GeometryType(geom), ST_Length(geom) FROM nyc_streets WHERE name = 'Pelham St'; ST_MultiLineString 50.323
3	SELECT ST_AsGeoJSON(geom) FROM nyc_subway_stations WHERE name = 'Broad St'; {"type":"Point", "crs":{"type":"name","properties":{"name":"EPSG:26918"}}, "coordinates":[583571.905921312,4506714.341192182]}
4	SELECT Sum(ST_Length(geom)) / 1000 FROM nyc_streets; 10418.9047172
	SELECT Sum(ST_Area(geom)) / 1000 FROM nyc_neighborhoods WHERE boroname = 'Manhattan'; 13965.3201224118 SELECT Sum(ST_Area(geom)) / 1000 FROM nyc_census_blocks WHERE boroname = 'Manhattan'; 14601.3987215548
5	SELECT ST_X(geom), name FROM nyc_subway_stations ORDER BY ST_X(geom) LIMIT 1; Tottenville
6	SELECT ST_Length(geom) FROM nyc_streets WHERE name = 'Columbus Cir';

308.34199

7

```
SELECT type, Sum(ST_Length(geom)) AS length
FROM nyc_streets
GROUP BY type
ORDER BY length DESC;
```

	type character varying (50)	length double precision
1	residential	8629870.33786606
2	motorway	403622.4781263628
3	tertiary	360394.8790513027
4	motorway_link	294261.419479668
5	secondary	276264.3038979258
6	unclassified	166936.3716044583
7	primary	135034.2330179469
8	footway	71798.48783780965
9	service	28337.635038596007
10	trunk	20353.58198260764
11	cycleway	8863.751448259294
12	pedestrian	4867.050328250257
13	construction	4803.081621035617
14	residential; motorway_link	3661.5750629374543
15	trunk_link	3202.1898124020076
16	primary_link	2492.5745708353616
17	living_street	1894.6390545733245
18	primary; residential; motorw...	1367.7657694133486
19	undefined	380.5386191034604
20	steps	282.74522134212725
21	motorway_link; residential	215.07778911517033

Практична робота №7. Просторові зв'язки

До сих пір ми використовували тільки просторові функції, які вимірюють (**ST_Area**, **ST_Length**), серіалізують (**ST_GeomFromText**) або десеріалізують (**ST_AsGML**) геометрії. Спільним для цих функцій є те, що вони працюють лише над однією геометрією за раз.

Просторові бази даних потужні, тому що вони не тільки зберігають геометрію, вони також мають можливість порівнювати *відношення між геометріями*.

На такі питання, як "Які найближчі велосипедні стійки до парку?" або "Де перетини ліній метро та вулиць?" можна відповісти, лише порівнюючи геометрію, що представляє велосипедні стійки, вулиці та лінії метро.

Стандарт OGC визначає наступний набір методів порівняння геометрії.

7.1. ST_Equals

ST_Equals (геометрія A, геометрія B) перевіряє просторову рівність двох геометрій.

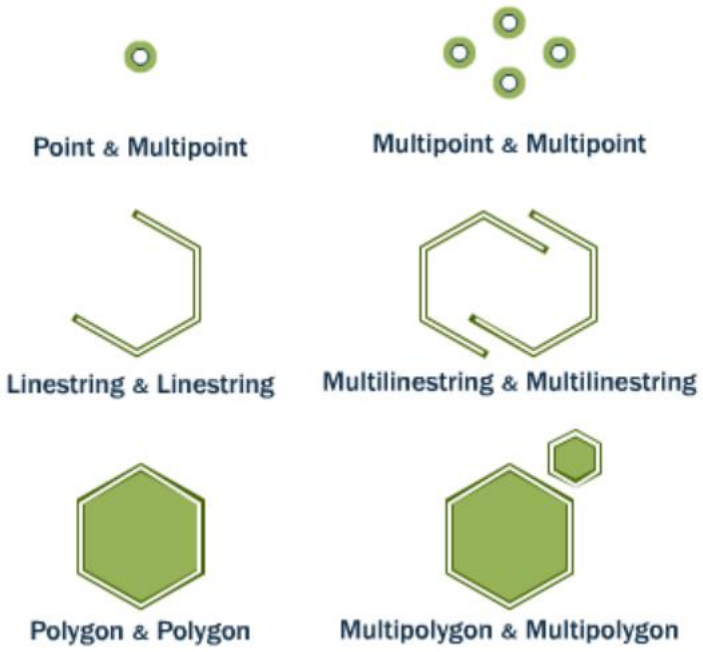

Рис. 45. Відношення між геометріями

ST_Equals повертає значення TRUE, якщо дві геометрії одного типу мають однакові значення координат x,y, тобто якщо друга фігура дорівнює (ідентична) першій фігурі.

По-перше, давайте витягнемо представлення точки з нашої `nyc_subway_stations` таблиці. Ми візьмемо тільки запис для "Broad St".

```
SELECT name, geom
FROM nyc_subway_stations
WHERE name = 'Broad St';
```

	name character varying (31)	geom geometry
1	Broad St	0101000020266900000EEBD4CF27CF2141BC17D69516315141

Рис. 45. Результат виконання запиту

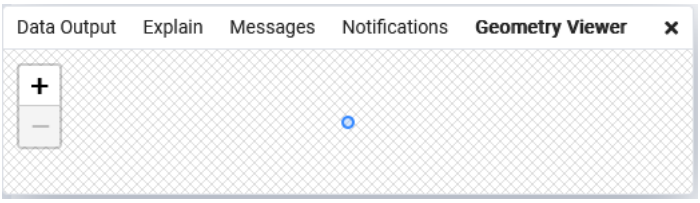

Рис. 46. Візуалізація результатів виконання запиту

Потім підключіть представлення геометрії назад до **ST_Equals:**

```
SELECT name
FROM nyc_subway_stations
WHERE ST_Equals(
  geom,
  '0101000020266900000EEBD4CF27CF2141BC17D69516315141');
```

	name character varying (31)
1	Broad St

Рис. 47. Результат виконання запиту

7.2. ST_Intersects, ST_Disjoint, ST_Crosses та ST_Overlaps

ST_Intersects, **ST_Crosses** і **ST_Overlaps** перевіряють, чи перетинаються інтер'єри геометрії.

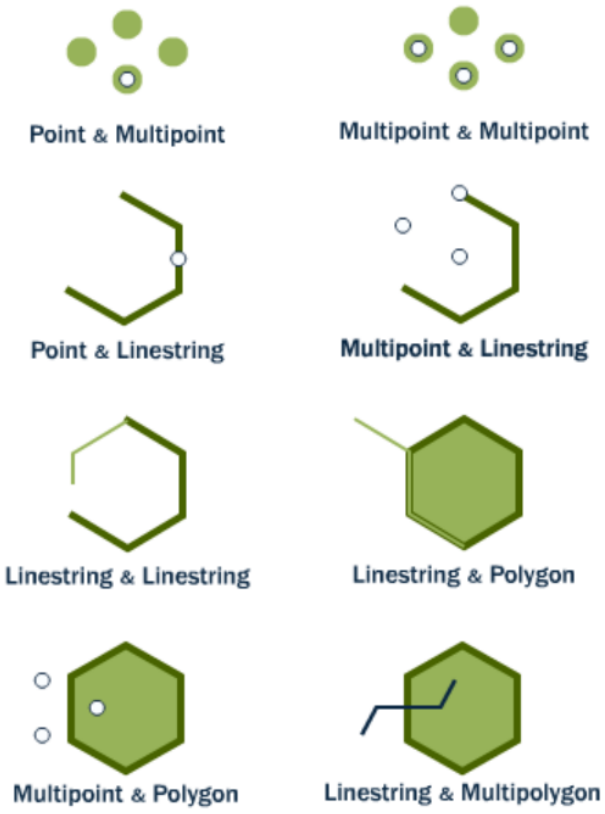

Рис. 48. Види перетинів геометрій

ST_Intersects(geometry A, geometry B) повертає t (TRUE), якщо дві фігури мають спільний простір, тобто якщо їх межі або внутрішній простір перетинаються.

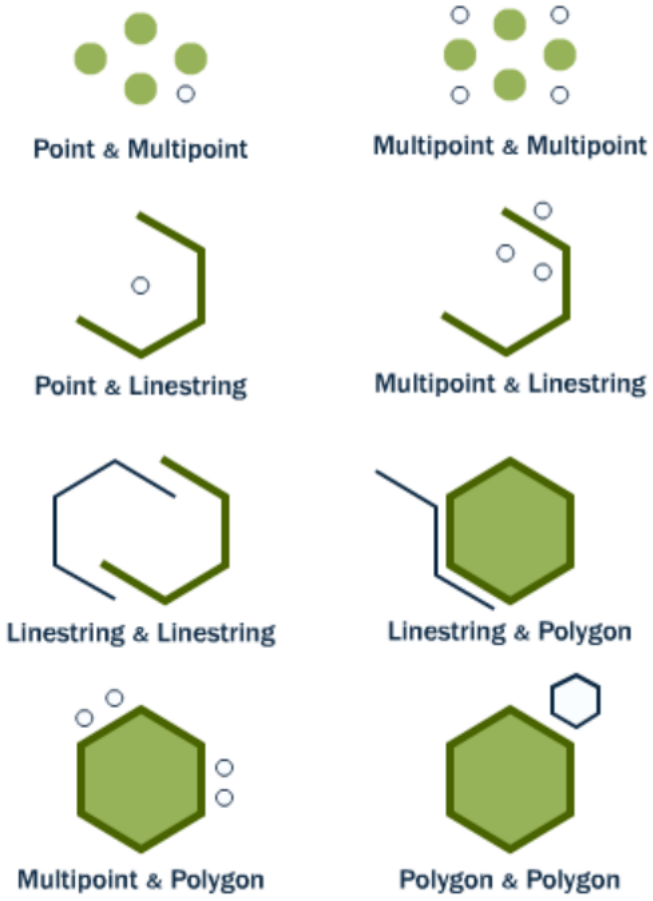

Рис. 48. Види роз'єднання геометрій

Протилежністю ST_Intersects є **ST_Disjoint (геометрія A , геометрія B)**. Якщо дві геометрії роз'єднані, вони не перетинаються, і навпаки. Насправді, часто більш ефективно тестувати "не перетинається", ніж тестувати "роз'єднаність", тому що тести перетинів можуть бути просторово проіндексовані, тоді як тест на роз'єднання не може.

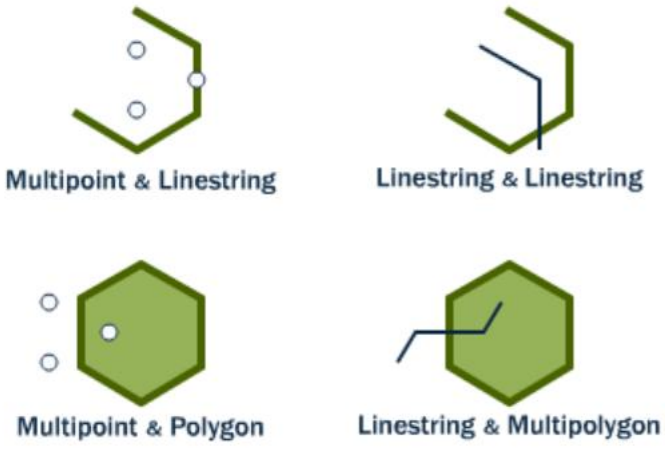

Рис. 49. Види перетину геометрій

Для геометрій multipoint/polygon, multipoint/linestring, linestring/linestring, linestring/polygon, та linestring/multipolygon порівняння, **ST_Crosses(geometry A, geometry B)** повертає t (TRUE) якщо перетин призводить до геометрії, розмірність якої на одну менше максимальної розмірності двох вихідних геометрій, а набір перетинів є внутрішньою для обох вихідних геометрій.

Рис. 49. Види накладання геометрій

ST_Overlaps(geometry A, geometry B) порівнює дві геометрії одного виміру і повертає значення TRUE, якщо їх набір перетинів призводить до геометрії, відмінної від обох, але з одного виміру.

Давайте візьмемо нашу станцію метро Broad Street і визначимо її околиці за допомогою **функції ST_Intersects**:

```
SELECT name, ST_AsText(geom)
FROM nyc_subway_stations
WHERE name = 'Broad St';
```

	name character vary	st_astext text
1	Broad St	POINT(583571.9059213118 4506714.341192182)

Рис. 50. Результат виконання запиту

```
SELECT name, boroname
FROM nyc_neighborhoods
WHERE ST_Intersects(geom, ST_GeomFromText('POINT(583571
4506714)',26918));
```

	name character varying (64)	boroname character varying (43)
1	Financial District	Manhattan

Рис. 51. Результат виконання запиту

7.3. ST_Touches

ST_Touches перевіряє, чи торкаються їх межі дві геометрії, але не перетинаються в їхніх внутрішніх просторах.

ST_Touches(geometry A, geometry B) повертає значення TRUE, якщо перетинається будь-яка з меж геометрії або якщо лише один з інтер'єрів геометрії перетинає межу іншої.

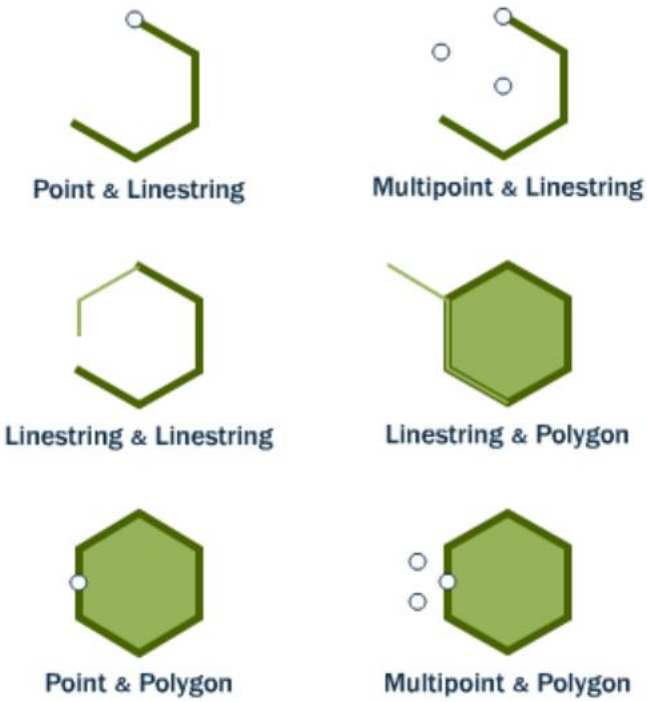

Рис. 52. Види дотикання геометрій

7.4. ST_Within and ST_Contains

ST_Within і **ST_Contains** перевіряють, чи одна геометрія повністю знаходиться в межах іншої.

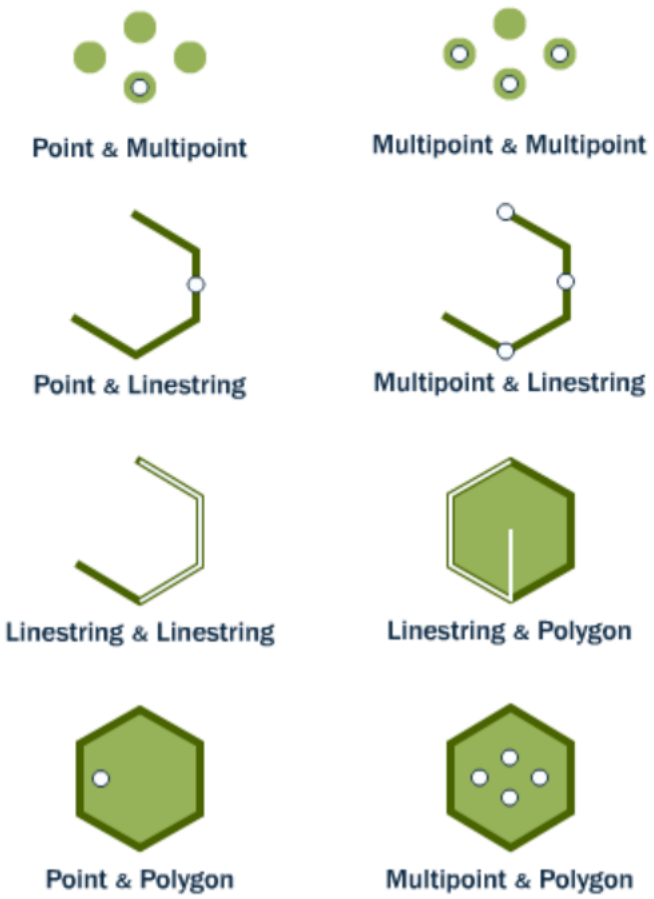

Рис. 53. Види геометрій всередині та містить

ST_Within(geometry A , geometry B) повертає значення TRUE, якщо перша геометрія повністю входить до другої геометрії. ST_Within тести на абсолютно протилежний результат ST_Contains.

ST_Contains(geometry A, geometry B) повертає значення TRUE, якщо друга геометрія повністю міститься в першій геометрії.

7.5. ST_Distance та ST_DWithin

Надзвичайно поширене питання ГІС - "знайти всі речі на відстані X від певного об'єкту".

ST_Distance(geometry A, geometry B) обчислює *найкоротшу* відстань між двома геометріями і повертає її як число з плаваючою точкою. Це корисно для фактичного звітування про відстань між об'єктами.

Для перевірки того, чи знаходяться два об'єкти на відстані один від одного, функція **ST_DWithin** забезпечує прискорений індекс true/false test. Це корисно для таких питань, як "скільки дерев знаходиться в межах 500-метрового буфера дороги?". Вам не потрібно обчислювати фактичний буфер, вам просто потрібно перевірити відношення на відстані.

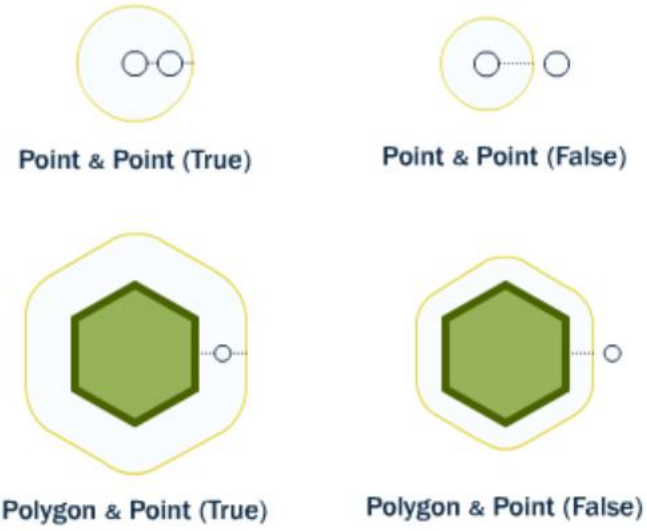

Рис. 54. Перевірка відношення об'єктів на відстані

Використовуючи нашу станцію метро Broad Street знову, ми можемо знайти вулиці поблизу (в межах 10 метрів від) зупинки метро:

```
SELECT name, boroname
FROM nyc_neighborhoods
WHERE ST_Intersects(geom, ST_GeomFromText('POINT(583571 4506714)',26918));
```

	name character varying (200)
1	Broad St
2	Wall St
3	Nassau St

Рис. 55. Результат виконання запиту

І ми можемо перевірити відповідь на карті. Станція Broad St насправді знаходиться на перетині вулиць Wall, Broad та Nassau Streets.

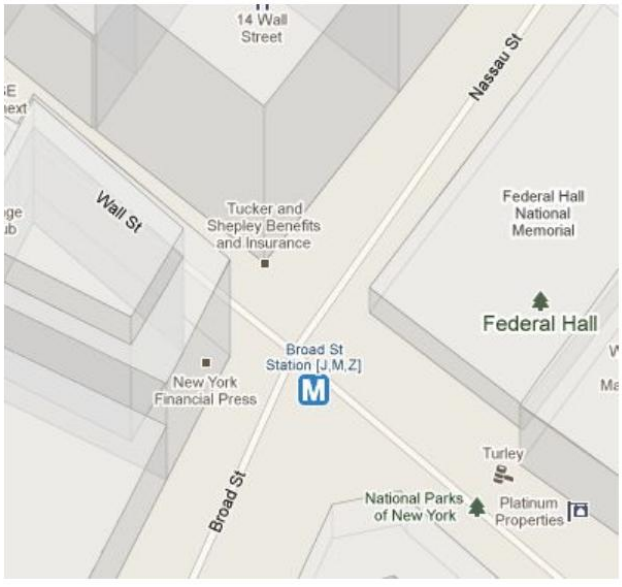

Рис. 56. Результат виконання запиту

Використані функції в практичній роботі:

- **sum(expression)** сума набору записів.
- **count(expression)** розмір набору записів
- **ST_Contains(geometry A, geometry B)** повертає true, якщо геометрія A містить геометрію B
- **ST_Crosses(geometry A, geometry B)** повертає true, якщо геометрія A перетинає геометрію B
- **ST_Disjoint(geometry A , geometry B)** повертає true, якщо геометрія не «просторово перетинається»"
- **ST_Distance(geometry A, geometry B)** повертає мінімальну відстань між геометрією A та геометрією B
- **ST_DWithin(geometry A, geometry B, radius)** повертає true, якщо геометрія A - відстань радіуса або менше від геометрії B
- **ST_Equals(geometry A, geometry B)** повертає true, якщо геометрія A така ж, як геометрія B
- **ST_Intersects(geometry A, geometry B)** повертає true, якщо геометрія A перетинає геометрію B
- **ST_Overlaps(geometry A, geometry B)** повертає true, якщо геометрія A і геометрія B об'єднують простір, але не повністю містяться один одним.
- **ST_Touches(geometry A, geometry B)** повертає true, якщо межа геометрії A торкається геометрії B
- **ST_Within(geometry A, geometry B)** повертає true, якщо геометрія A знаходиться в геометрії B

Також пам'ятайте таблиці, які у нас є:

nyc_census_blocks

blkid, popn_total, boroname, geom

nyc_streets

name, type, geom

nyc_subway_stations

name, geom

nyc_neighborhoods

name, boroname, geom

Завдання: використовуючи таблиці БД створіть запити, щоб дати відповідь на запитання:

1. **Яке значення геометрії для вулиці Atlantic Commons?**
2. **В якому районі та боро знаходиться вулиця Atlantic Commons?**
3. **З якими вулицями перетинається вулиця Atlantic Commons?**
4. **Приблизно скільки людей живе (в межах 50 метрів від) на вулиці Atlantic Commons?**

Варіанти правильних відповідей

1	SELECT ST_AsText(geom) FROM nyc_streets WHERE name = 'Atlantic Commons'; MULTILINESTRING((586781.701577724 4504202.15314339,586863.51964484 4504215.9881701))
2	SELECT name, boroname FROM nyc_neighborhoods WHERE ST_Intersects(geom, ST_GeomFromText('LINESTRING(586782 4504202,586864 4504216)', 26918)); name | boroname ----------+---------- Fort Green | Brooklyn
3	SELECT name FROM nyc_streets WHERE ST_DWithin(geom, ST_GeomFromText('LINESTRING(586782 4504202,586864 4504216)', 26918), 0.1); name ---------- Cumberland St Atlantic Commons
4	SELECT Sum(popn_total) FROM nyc_census_blocks WHERE ST_DWithin(geom, ST_GeomFromText('LINESTRING(586782 4504202,586864 4504216)', 26918), 50); 1438

Практична робота №8. Просторові з'єднання

Просторові з'єднання - це своєрідний клей для просторових баз даних. Вони дозволяють об'єднувати інформацію з різних таблиць за допомогою просторових зв'язків як ключа об'єднання. Багато з того, що ми вважаємо «стандартним ГІС-аналізом», може бути виражено як просторові об'єднання.

У попередній роботі ми досліджували просторові відносини, використовуючи двоетапний процес: спочатку ми витягли точку станції метро для «Broad St»; потім ми використали цей момент, щоб задати додаткові питання, такі як "в якому районі знаходиться станція "Broad St"?

Використовуючи просторове об'єднання, ми можемо відповісти на питання в один крок, отримавши інформацію про станцію метро і околиці, яка її містить:

```
SELECT
  subways.name AS subway_name,
  neighborhoods.name AS neighborhood_name,
  neighborhoods.boroname AS borough
FROM nyc_neighborhoods AS neighborhoods
JOIN nyc_subway_stations AS subways
ON ST_Contains(neighborhoods.geom, subways.geom)
WHERE subways.name = 'Broad St';
```

	subway_name character varying (31)	neighborhood_name character varying (64)	borough character varying (43)
1	Broad St	Financial District	Manhattan

Рис. 57. Результат виконання запиту

Ми могли б приєднатися до кожної станції метро до її району, але в цьому випадку ми хотіли отримати інформацію лише про одну. Будь-яка функція, яка забезпечує істинний /хибний зв'язок між двома таблицями, може бути використана для керування просторовим об'єднанням, але найбільш часто використовуваними з них є: **ST_Intersects**, **ST_Contains** та **ST_DWithin**.

8.1. Приєднатися та підсумувати

Комбінація **JOIN** з **GROUP BY** забезпечує такий аналіз, який зазвичай проводиться в гіс-системі.

Наприклад: **"Яке населення і расовий склад районів Манхеттена?"** Тут у нас є питання, яке поєднує інформацію про населення з перепису з кордонами кварталів, з обмеженням тільки на один район Манхеттена.

```
SELECT
  neighborhoods.name AS neighborhood_name,
  Sum(census.popn_total) AS population,
  100.0 * Sum(census.popn_white) / Sum(census.popn_total) AS white_pct,
  100.0 * Sum(census.popn_black) / Sum(census.popn_total) AS black_pct
FROM nyc_neighborhoods AS neighborhoods
JOIN nyc_census_blocks AS census
ON ST_Intersects(neighborhoods.geom, census.geom)
WHERE neighborhoods.boroname = 'Manhattan'
GROUP BY neighborhoods.name
ORDER BY white_pct DESC;
```

	neighborhood_name character varying (64)	population double precision	white_pct double precision	black_pct double precision
1	Carnegie Hill	18763	90.08154346319886	1.4123541011565315
2	West Village	26718	87.59637697432443	2.1745639643685903
3	North Sutton Area	22460	87.56455921638468	1.553873552983081
4	Upper East Side	203741	85.02216048807064	2.7220834294520984
5	Soho	15436	84.64628142005701	2.235034983156258
6	Greenwich Village	57224	81.97609394659584	2.43778834055641
7	Central Park	46600	79.49356223175965	7.967811158798283
8	Tribeca	20908	79.1180409412665	3.548880811172757
9	Gramercy	104876	75.4586368663946	4.720813150768526
10	Murray Hill	29655	75.01264542235711	2.5122239082785365
11	Chelsea	61340	74.81578089338116	6.372676882947506
12	Upper West Side	214761	74.561489283436	9.188353565125885
13	Midtown	76840	72.60150963040083	5.166579906298803
14	Battery Park	17153	71.8300005829884	3.375502827493733
15	Financial District	34807	69.92846266555578	3.829689430287011
16	Clinton	32201	65.2743703611689	7.943852675382752
17	East Village	82266	63.26550458269516	8.771545960664186
18	Garment District	10539	55.18550147072777	7.08795900939368
19	Morningside Heights	42844	52.719167211278126	19.370273550555503
20	Little Italy	12568	49.029280712921704	1.8220878421387652
21	Yorkville	58450	35.5551753635586	29.72284003421728
22	Inwood	50047	35.21489799588387	16.846164605271046
23	Washington Heights	169013	34.873057102116405	16.827107973942834
24	Lower East Side	96156	33.50908939639752	9.075876700361912
25	East Harlem	60576	26.43621236133122	40.38398045430534
26	Hamilton Heights	67432	23.889251393996915	35.772333610155414

Рис. 58. Результат виконання запиту

Що тут відбувається? Поняття (фактичний порядок оцінки оптимізований під обкладинками бази даних) ось що відбувається:

1. **JOIN** створює віртуальну таблицю, яка містить стовпці як з районів, так і з таблиць перепису.
2. **WHERE** фільтрує нашу віртуальну таблицю лише до рядків на Манхеттені.
3. Решта рядків групуються за назвою районів і подаються через функцію агрегації в **Sum()** значення генеральної сукупності.
4. Після невеликої арифметики та форматування (наприклад, **GROUP BY**, **ORDER BY**) на кінцевих числах наш запит видає відсотки.

Ми також можемо використовувати дистанційні тести як ключ об'єднання, щоб створити узагальнені запити "всі елементи в радіусі". Давайте вивчимо расову географію Нью-Йорка за допомогою дистанційних запитів.

По-перше, давайте отримаємо базовий расовий склад міста.

```
SELECT
  100.0 * Sum(popn_white) / Sum(popn_total) AS white_pct,
  100.0 * Sum(popn_black) / Sum(popn_total) AS black_pct,
  Sum(popn_total) AS popn_total
FROM nyc_census_blocks;
```

	white_pct double precision	black_pct double precision	popn_total double precision
1	44.00395007628105	25.546578900241613	8175032

Рис. 59. Результат виконання запиту

Так, з 8 млн осіб в Нью-Йорку близько 44% зареєстровані як "білі", а 26% записані як "темношкірі".

Зверніть увагу, що вміст поля маршрутів таблиці nyc_subway_stations - це те, що нас цікавить, щоб знайти A-поїзд. Значення там трохи складні.

```
SELECT DISTINCT routes FROM nyc_subway_stations;
```

	routes character varying (20)
1	4,5
2	[null]
3	N,Q,R,W
4	J
5	B,M,Q,R
6	D,F,N,Q
7	J,M
8	E,F
9	1,2,3
10	N

Read-only column

Рис. 60. Результат виконання запиту

Ключове слово DISTINCT виключає повторювані рядки з результату. Без ключового слова DISTINCT наведений вище запит ідентифікує 491 результат замість 73.

Отже, щоб знайти A-поїзд, ми хочемо будь-який ряд на маршрутах, який має "A" в ньому. Ми можемо зробити це кількома способами, але в цій роботі ми будемо використовувати той факт, що **strpos (маршрути, 'A')** поверне ненульове число тільки в тому випадку, якщо "A" знаходиться в полі маршрутів.

```
SELECT DISTINCT routes FROM nyc_subway_stations;
```

	routes character varying (20)
1	A,C
2	A,B,C,D
3	A,C,E,L
4	A,C,F
5	A,B,C
6	A,S
7	A,C,E
8	A,C,G
9	A

Рис. 61. Результат виконання запиту

Підіб'ємо підсумки расового складу в межах 200 метрів від лінії A-поїзда.

```
SELECT
  100.0 * Sum(popn_white) / Sum(popn_total) AS white_pct,
  100.0 * Sum(popn_black) / Sum(popn_total) AS black_pct,
  Sum(popn_total) AS popn_total
FROM nyc_census_blocks AS census
JOIN nyc_subway_stations AS subways
ON ST_DWithin(census.geom, subways.geom, 200)
WHERE strpos(subways.routes,'A') > 0;
```

	white_pct double precision	black_pct double precision	popn_total double precision
1	45.59012559002023	22.09362356709373	189824

Рис. 62. Результат виконання запиту

Таким чином, расова географія уздовж A-поїзда радикально не відрізняється від Нью-Йорка в цілому.

8.2. Розширене приєднання

В останньому розділі ми побачили, що A-поїзд не обслуговував населення, яке сильно відрізнялося від расового складу решти міста. Чи існують поїзди, які мають несередню расову географію?

Щоб відповісти на це питання, ми додамо ще одне приєднання до нашого запиту, щоб ми могли одночасно розрахувати склад багатьох ліній метро одночасно. Для цього нам потрібно буде створити нову таблицю, яка перераховує всі рядки, які ми хочемо підсумувати.

```
CREATE TABLE subway_lines ( route char(1) );
INSERT INTO subway_lines (route) VALUES
  ('A'),('B'),('C'),('D'),('E'),('F'),('G'),
  ('J'),('L'),('M'),('N'),('Q'),('R'),('S'),
  ('Z'),('1'),('2'),('3'),('4'),('5'),('6'),
  ('7');
```

Тепер ми можемо приєднати таблиці ліній метро до нашого початкового запиту.

```
SELECT
  lines.route,
  100.0 * Sum(popn_white) / Sum(popn_total) AS white_pct,
  100.0 * Sum(popn_black) / Sum(popn_total) AS black_pct,
  Sum(popn_total) AS popn_total
FROM nyc_census_blocks AS census
JOIN nyc_subway_stations AS subways
ON ST_DWithin(census.geom, subways.geom, 200)
JOIN subway_lines AS lines
ON strpos(subways.routes, lines.route) > 0
GROUP BY lines.route
ORDER BY black_pct DESC;
```

	route character (1)	white_pct double precision	black_pct double precision	popn_total double precision
1	S	39.839644455121466	46.503108014774334	33301
2	3	42.72731756087282	42.06198693548893	223047
3	5	33.79377760724286	41.38562664729877	218919
4	2	39.26304853922876	38.39114588512005	291661
5	C	46.87871806640494	30.598767440098747	224411
6	4	37.55300060572121	27.428313466439615	174998
7	B	39.95588172248356	26.852519457641385	256583
8	A	45.59012559002023	22.09362356709373	189824
9	J	37.62955269040576	21.637651380013697	132861
10	Q	56.88447982881239	20.63141166844987	127112
11	Z	38.35718630567766	20.15700496952864	87131
12	D	39.4971289442432	19.38569196913136	234931
13	L	57.59003977551354	16.775640676365352	110118
14	G	49.57114923117945	16.13115871181821	135012
15	6	52.32131878266216	15.716646172763603	260240
16	1	59.05773443745385	11.26648400266063	327742
17	F	60.867158591172384	7.501776071199752	229439
18	M	56.53746354680934	6.445612987669063	174196
19	E	66.76268167725763	4.717561951670001	90958
20	R	58.45765714546774	4.017279275529318	196999
21	7	35.732072928975306	3.4804347613792834	102401
22	N	59.689293060517485	3.4744776442567935	147792

Рис. 63. Результат виконання запиту

Як і раніше, об'єднання створюють віртуальну таблицю всіх можливих комбінацій, доступних у межах обмежень **JOIN ON**, і ці рядки потім подаються до резюме **GROUP**. Просторова особливість знаходиться в функції **ST_DWithin**, яка забезпечує включення в розрахунок тільки блоків перепису, близьких до відповідних станцій метро.

Використані функції в практичній роботі:

sum(expression) сума набору записів.
count(expression) розмір набору записів
ST_Area(geometry) повертає площу полігонів
ST_AsText(geometry) повертає текст WKT
ST_Contains(geometry A, geometry B) повертає true, якщо геометрія A містить геометрію B
ST_Distance(geometry A, geometry B) повертає мінімальну відстань між геометрією A та геометрією B
ST_DWithin(geometry A, geometry B, radius) повертає true, якщо геометрія A - це відстань радіуса або менше від геометрії B
ST_GeomFromText(text) повертає геометрію
ST_Intersects(geometry A, geometry B) повертає true, якщо геометрія A перетинає геометрію B
ST_Length(linestring) повертає довжину лінії
ST_Touches(geometry A, geometry B) повертає true, якщо межа геометрії A торкається геометрії B
ST_Within(geometry A, geometry B) returns the true if geometry A is within geometry B

Також пам'ятайте таблиці, які у нас є:

nyc_census_blocks

blkid, popn_total, boroname, geom

nyc_streets

name, type, geom

nyc_subway_stations

name, geom

nyc_neighborhoods

name, boroname, geom

Завдання: використовуючи таблиці БД створіть запити, щоб дати відповідь на запитання:

1. **Яка станція метро в 'Little Italy'? На якому маршруті метро?**
2. **Які всі квартали обслуговує 6-поїзд? (Підказка: Стовпець маршрутів у таблиці nyc_subway_stations має такі значення, як "B,D,6,V" та "C,6")**
3. **Після терактів 11 вересня район «Баттері-Парк» був закритий на кілька днів. Скільки людей довелося евакуювати?**
4. **У якому районі найбільша щільність населення (осіб/км^2)?**

Варіанти правильних відповідей

1	SELECT s.name, s.routes FROM nyc_subway_stations AS s JOIN nyc_neighborhoods AS n ON ST_Contains(n.geom, s.geom) WHERE n.name = 'Little Italy'; name | routes ----------+-------- Spring St | 6
2	SELECT DISTINCT n.name, n.boroname FROM nyc_subway_stations AS s JOIN nyc_neighborhoods AS n ON ST_Contains(n.geom, s.geom) WHERE strpos(s.routes,'6') > 0; name | boroname --------------------+----------- Midtown | Manhattan Hunts Point | The Bronx Gramercy | Manhattan Little Italy | Manhattan Financial District | Manhattan South Bronx | The Bronx Yorkville | Manhattan Murray Hill | Manhattan Mott Haven | The Bronx Upper East Side | Manhattan Chinatown | Manhattan East Harlem | Manhattan Greenwich Village | Manhattan

	Parkchester | The Bronx Soundview | The Bronx
3	SELECT Sum(popn_total) FROM nyc_neighborhoods AS n JOIN nyc_census_blocks AS c ON ST_Intersects(n.geom, c.geom) WHERE n.name = 'Battery Park'; 17153
4	SELECT n.name, Sum(c.popn_total) / (ST_Area(n.geom) / 1000000.0) AS popn_per_sqkm FROM nyc_census_blocks AS c JOIN nyc_neighborhoods AS n ON ST_Intersects(c.geom, n.geom) GROUP BY n.name, n.geom ORDER BY 2 DESC; name | popn_per_sqkm ----------+---------- Upper East Side | 48524.4877489857 Upper West Side | 40152.4896080024

Практична робота №9. Робота з проекціями

Земля не плоска, і немає простого способу покласти її на плоску паперову карту (або екран комп'ютера), тому люди придумали всілякі геніальні рішення, кожен з яких має плюси і мінуси. Деякі проекції зберігають площу, тому всі об'єкти мають відносний розмір один до одного; інші проекції зберігають кути (конформні), такі як проекція Меркатора; деякі проекції намагаються знайти хорошу проміжну суміш з невеликими спотвореннями за кількома параметрами. Спільним для всіх прогнозів є те, що вони перетворюють (сферичний) світ на плоску декартову систему координат, і яка проекція на вибір залежить від того, як Ви будете використовувати дані.

Ми вже стикалися з прогнозами, коли завантажували наші дані для Нью-Йорку. (Нагадаємо, що SRID 26918). Іноді, однак, Вам потрібно трансформувати і перепроектувати дані між просторовими системами відліку. PostGIS включає вбудовану підтримку зміни проекції даних, використовуючи функцію **ST_Transform(geometry, srid).** Для управління просторовими ідентифікаторами посилань на геометрії PostGIS надає **функції ST_SRID(geometry)** та **ST_SetSRID(geometry, srid).**

Ми можемо перевірити SRID наших даних за допомогою **функції ST_SRID**:

```
SELECT ST_SRID(geom) FROM nyc_streets LIMIT 1;
```

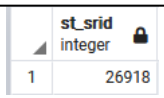

Рис. 64. Результат виконання запиту

А що таке визначення «26918»? Як ми побачили в практичній роботі №2, визначення міститься в таблиці spatial_ref_sys. Насправді існує **два** визначення.

Визначення «відомий текст» (WKT) знаходиться в стовпці srtext, а друге визначення є у форматі «proj.4» у стовпці proj4text.

```
SELECT * FROM spatial_ref_sys WHERE srid = 26918;
```

Data Output | Explain | Notifications

	srid [PK] integer	auth_name character varying (	auth_srid integer	srtext character varying (2048)	proj4text character varying (2048)
1	26918	EPSG	26918	PROJCS["NAD83 / UTM zone 18N",GEOGCS["NAD83",DATUM["North_American_Datum_1983",SPHER...	+proj=utm +zone=18 +datum=NAD83 +units=m +no_defs

Рис. 65. Результат виконання запиту

Алгоритм перепроектування PostGIS спробує знайти найкращу проекцію з таблиці spatial_ref_sys :

1. **auth_name / auth_srid** Якщо proj може знайти відповідну назву та SRID у своєму внутрішньому каталозі, він використовуватиме це для створення визначення проекції.
2. **srtext** буде використовуватись, якщо proj може розібрати і сформувати об'єкт визначення з srtext, він буде використовувати це.
3. **proj4text** у всіх інших випадках, proj спробує обробити proj4text.

Вся ця надмірність означає, що все, що вам потрібно для створення нової проекції в PostGIS, є або дійсним рядком srtext , або рядком proj4text . Усі загальні пари імені/коду вже завантажені в таблицю за замовчуванням.

Якщо у вас є вибір при створенні власної проекції, заповніть стовпець srtext , оскільки цей стовпець також використовується зовнішніми програмами, такими як GeoServer, QGIS, FME та іншими.

9.1. Порівняння даних

Разом узяті координати і SRID визначають місце на земній кулі. Без SRID координата - це всього лише абстрактне поняття. «Декартова» координатна площина визначається як «плоска» система координат, розміщена на поверхні Землі. Оскільки функції PostGIS працюють на такій площині, операції порівняння вимагають, щоб обидві геометрії були представлені в одному SRID.

Якщо Ви будете подавати в геометрії з різними SRID, Ви просто отримаєте помилку.

Будьте обережні, при використанні ST_Transform для перетворення на льоту. Просторові індекси будуються з використанням SRID збережених геометрій. Якщо порівняння проводиться в іншому SRID, просторові індекси (часто) не використовуються. Найкраще вибрати один SRID для всіх таблиць у базі даних. Використовуйте функцію трансформації лише під час читання або запису даних у зовнішні програми.

9.2. Трансформування даних

Якщо повернутися до нашого визначення proj4 для SRID 26918, то можна побачити, що нашою робочою проекцією є UTM (Універсальна поперечна проекція Меркатора) зони 18, з метрами як одиницею виміру.

```
SELECT srtext FROM spatial_ref_sys WHERE srid = 26918;
```

```
PROJCS["NAD83 / UTM zone 18N",
  GEOGCS["NAD83",
    DATUM["North_American_Datum_1983",
      SPHEROID["GRS 1980",6378137,298.257222101,AUTHORITY["EPSG","7019"]],
      TOWGS84[0,0,0,0,0,0,0],
      AUTHORITY["EPSG","6269"]],
    PRIMEM["Greenwich",0,AUTHORITY["EPSG","8901"]],
    UNIT["degree",0.0174532925199433,AUTHORITY["EPSG","9122"]],
    AUTHORITY["EPSG","4269"]],
  PROJECTION["Transverse_Mercator"],
  PARAMETER["latitude_of_origin",0],
  PARAMETER["central_meridian",-75],
  PARAMETER["scale_factor",0.9996],
  PARAMETER["false_easting",500000],
  PARAMETER["false_northing",0],
  UNIT["metre",1,AUTHORITY["EPSG","9001"]],
  AXIS["Easting",EAST],AXIS["Northing",NORTH],
  AUTHORITY["EPSG","26918"]]
```

Рис. 66. Опис системи координат NAD83 SRID 26918

Давайте переведемо деякі дані з нашої робочої проекції в географічні координати – також відомі як "довгота/широта".

Щоб перетворити дані з одного SRID на інший, спочатку потрібно перевірити, чи має ваша геометрія припустимий SRID. Оскільки ми вже підтвердили дійсний SRID, нам далі потрібен SRID проекції в яку буде здійснена трансформація. Найпоширенішим SRID для географічних координат є 4326, що відповідає «довготі/широті на сфероїді WGS84». Ви можете побачити визначення за посиланням https://epsg.io/4326 або по аналогії а запитом з таблиці spatial_ref_sys.

Переведемо координати станції метро «Broad St» в географічні:

```
SELECT ST_AsText(ST_Transform(geom,4326))
FROM nyc_subway_stations WHERE name = 'Broad St';
```

	st_astext text
1	POINT(-74.01067146887341 40.70710481558761)

Рис. 67. Трансформовані координати.

Якщо Ви завантажуєте дані або створюєте нову геометрію без вказівки SRID, значення SRID буде дорівнює 0. Нагадаємо в Geometries, що коли ми створювали нашу таблицю геометрій, ми не вказували SRID. Якщо ми запитуємо нашу базу даних, ми повинні очікувати, що всі nyc_ таблиці матимуть SRID 26918, тоді як таблиця геометрії за замовчуванням відповідає SRID 0.

Щоб переглянути призначення SRID таблиці, надішліть запит на geometry_columns таблицю бази даних.

	name name	srid integer
1	nyc_census_blocks	26918
2	nyc_homicides	26918
3	nyc_neighborhoods	26918
4	nyc_streets	26918
5	nyc_subway_stations	26918
6	nyc_census_blocks_2000	26918
7	geometries	0

Рис. 68. Системи координат таблиць БД.

Однак якщо Ви знаєте, яким передбачається SRID системи координат, Ви можете встановити його постфактум, використовуючи **ST_SetSRID** геометрії. Тоді Ви зможете трансформувати геометрію в інші системи. Наприклад:

```
SELECT ST_AsText(
   ST_Transform(
     ST_SetSRID(geom,26918),
     4326)
   )
  FROM geometries;
```

Використані функції в практичній роботі:

1. **Sum (вираз)** функція для сумування набору записів
2. **ST_Length(linestring)** повертає довжину смуги
3. **ST_SRID(geometry)** повертає SRID геометрії
4. **ST_Transform (geometry, srid)** перетворює геометрію в різні просторові системи відліку
5. **ST_GeomFromText(текст)** повертає геометрію
6. **ST_AsText(геометрія) повертає** текст WKT
7. **ST_AsGML(геометрія) повертає** текст GML

Також пам'ятайте таблиці, які у нас є:

nyc_census_blocks
blkid, popn_total, boroname, geom
nyc_streets
name, type, geom
nyc_subway_stations
name, geom
nyc_neighborhoods
name, boroname, geom

Завдання: використовуючи таблиці БД створіть запити, щоб дати відповідь на запитання:

1. **Яка довжина всіх вулиць в Нью-Йорку, виміряна в UTM 18?**
2. **Яке визначення WKT для SRID 2831?**
3. **Яка довжина всіх вулиць в Нью-Йорку, вимірюється в SRID 2831**
4. **Скільки вулиць перетинають 74-й меридіан?**

Варіанти правильних відповідей

1	`SELECT Sum(ST_Length(geom))`
	`  FROM nyc_streets;`
	
	`10418904.7172`

2	`SELECT srtext FROM spatial_ref_sys`
	`WHERE SRID = 2831;`
	
	`PROJCS["NAD83(HARN) / New York Long Island",`

	GEOGCS["NAD83(HARN)", DATUM["NAD83 (High Accuracy Regional Network)", SPHEROID["GRS 1980", 6378137.0, 298.257222101, AUTHORITY["EPSG","7019"]], TOWGS84[-0.991, 1.9072, 0.5129, 0.0257899075194932, - 0.009650098960270402, -0.011659943232342112, 0.0], AUTHORITY["EPSG","6152"]], PRIMEM["Greenwich", 0.0, AUTHORITY["EPSG","8901"]], UNIT["degree", 0.017453292519943295], AXIS["Geodetic longitude", EAST], AXIS["Geodetic latitude", NORTH], AUTHORITY["EPSG","4152"]], PROJECTION["Lambert Conic Conformal (2SP)", AUTHORITY["EPSG","9802"]], PARAMETER["central_meridian", -74.0], PARAMETER["latitude_of_origin", 40.166666666666664], PARAMETER["standard_parallel_1", 41.03333333333333], PARAMETER["false_easting", 300000.0], PARAMETER["false_northing", 0.0], PARAMETER["scale_factor", 1.0], PARAMETER["standard_parallel_2", 40.666666666666664], UNIT["m", 1.0], AXIS["Easting", EAST], AXIS["Northing", NORTH], AUTHORITY["EPSG","2831"]]
3	SELECT Sum(ST_Length(ST_Transform(geom,2831))) FROM nyc_streets; 10421993.706374
4	SELECT Count(*) FROM nyc_streets WHERE ST_Intersects(ST_Transform(geom, 4326), 'SRID=4326;LINESTRING(-74 20, -74 60)'); 223

Список використаних джерел

Основна

1. Геоінформаційні системи і бази даних : монографія / В. І. Зацерковний, В. Г. Бурачек, О. О. Железняк, А. О. Терещенко. – Ніжин : НДУ ім. М. Гоголя, 2014. – 492 с.
2. Говоров М. Геоінформаційні технології та інфраструктура геопросторових даних: у шести томах. Том 3: Просторові кадастрові інформаційні системи для інфраструктури просторових даних. Навчальний пособник /Говоров М., Лященко А.А., Кейк Д., Зандберген, П. М.А. Молочко, Л. Бевайніс, Л.М. Даценко, Путренко В.В. – К.: Планета-Прінт, 2017. – 520 с.
3. Де Мерс. Географические информационные системы. Основы / Де Мерс, Н. Майкл ; пер. с англ. – М. : Дата+, 1999. – 489 с.
4. Іщук О. О. Просторовий аналіз в ГІС : навч. посіб. / О. О. Іщук, М. М. Коржнев, О. Є. Кошляков ; за ред. акад. Д. М. Гродзинського. – К. : ВПЦ "Київський університет", 2003. – 195 с.
5. Карпінський Ю.О. Геопросторовий аналіз: навч. посіб. /Карпінський Ю.О., Лященко А.А., Кравченко Ю.В. – К.: КНУБА, 2016.-184с.
6. Кейк Д. Геоінформаційні технології та інфраструктура геопросторових даних: у шести томах. Том 2: Системи керування базами геоданих для інфраструктури просторових даних. Навчальний пособник /Кейк Д., Лященко А.А., Путренко В.В., Хмелевський Ю., Дорошенко К.С., Говоров М. – К.: Планета-Прінт, 2017. – 456 с.
7. Патракеєв І.М. , Толстохатько В.А., Поморцева О.Є.Бази даних: проектування та використання для обліку нерухомого майна Х.: ХНУМГ, 2014. – с. 176. Рекомен-довано МОН України, лист №1/11-7213 від 14.05.2014
8. Патракеєв І.М. Геопространственные технологи в моделировании градостроительных систем. Монография. Харк. нац. ун-т гор. хоз-ва им.А.Н. Бекетова – Х.: ХНУГХ, 2014. – 208 с. ISBN 978-966-695-339-4

Додаткова

9. Баранов Ю. Б. Геоинформатика: толковый словарь основных терминов / Ю. Б. Баранов, А. М. Берлянт, Е. Г. Капралов и др. – М. : ГИСАссоциация, 1999. – 204 с.
10. Берлянт А. М. Виртуальные геоизображения / А. М. Берлянт. – М. : Научный мир, 2001. – 56 с.
11. Берлянт А. М. Картография : учебник для вузов / А. М. Берлянт. – М. : Аспект Пресс, 2001. – 336 с.
12. Бугаевский Л. М. Геоинформационные системы / Л. М. Бугаевский, В. Я. Цветков. – М. : Златоуст, 2000. – 222 с.
13. Воробьева А. А. Геоинформационные системы территориального управления / А. А. Воробьева. – СПб. СПб НИУ информационных технологий, механики и оптики, 2012. – 130 с.
14. Гиттис В. Г. Основы пространственного прогнозирования в геоинформатике / В. Г. Гиттис, Б. В. Ермаков. – М. : ФИЗМАТЛИТ, 2004. – 256 с.

15. Гурьянова Л. В. Введение в ГИС / Л. В. Гурьянова. – Мн. : БГУ, 2008. – 135 с.
16. Замай С. С. Программное обеспечение и технологии геоинформационных систем / С. С. Замай, О. Э. Якубайлик. – Красноярск : КГУ, 1998. – 110 с.
17. Зейлер М. Моделирование нашего мира : руководство ESRI по проектированию базы геоданных / М. Зейлер ; пер. с англ. – М. : СП ООО "Дата+", 2004. – 254 с.
18. Иванников А. Д. Геоинформатика / А. Д. Иванников, В. П. Кулагин, А. Н. Тихонов и др. – М. : МАКС-ПРЕСС, 2001. – 349 с.
19. Капралов Е. Г. Геоинформатика / Е. Г. Капралов, А. В. Кошкарев, В. С. Тикунов и др. – М. : Академия, 2005. – 480 с.
20. Карпик А. П. Методологические и технологические основы геоинформационного обеспечения территорий : монография / А. П. Карпик. – Новосибирск : СГГА, 2004. – 260 с.
21. Кольцов А. С. Геоинформационные системы : учеб. Пособие / А. С. Кольцов, Е. Д. Федорков. Воронеж : ГОУВПО "Воронежский государственный технический университет", 2006. – 203 с.
22. Королев Ю. К. Общая геоинформатика / Ю. К. Королев. – М. : СП ООО "Дата+", 1998. Ч. I. Теоретическая геоинформатика. – Вып. 1. – М. : СП ООО "Дата+", 1998. – 118 с.
23. Краснощеков Р. В. ГИС-технологии : словарь терминов и понятий / Р. В. Краснощеков. – Гомель : ГГУ им. Ф. Скорины, 2009. – 87 с.
24. Ладичук Д. О. Бази геоінформаційних даних / Д. О. Ладичук, В. І. Пічура. – Херсон : ХДУ, 2007. – 103 с.
25. Митчелл Э. Руководство по ГИС-анализу / Э. Митчелл. – 2000. Ч. 1. Пространственные модели и взаимосвязи. – 2000. – 177 с.
26. Морозов В. В. Геоінформаційні технології в агросфері / В. В. Морозов, К. С. Лисогоров, Н. М. Шпоринська. – Херсон : ХДУ, 2007. – 223 с.
27. Морозов В. В. ГІС в управлінні водними і земельними ресурсами : навч. посіб. / В. В. Морозов. – Херсон : Вид-во ХДУ, 2006. – 88 с.
28. Морозов В. В. Моделювання та прогнозування для проектів геоінформаційних систем / В. В. Морозов, С. Я. Плоткін, М. Г. Поляков та ін. – Херсон : ХДУ, 2007. – 328 с.
29. Середович В. А. Геоинформационные системы (назначение, функции, классификация) : монография / В. А. Середович, В. Н. Клюшниченко, Н.В. Тимофеева. – Новосибирск : СГГА, 2008. – 117 с.