МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

Факультет прикладної математики
Кафедра програмного забезпечення комп’ютерних систем

КУРСОВА РОБОТА

з дисципліни “Бази даних”

спеціальність 121 – Програмна інженерія

на тему: Система кліматичних показників

Студент **Звєрєв К.В** _____
групи КП-93 (підпис)

Викладач **Петрашенко А.В.** _____
к.т.н, доцент кафедри СПіСКС (підпис)

Захищено з оцінкою _____

Київ – 2020

АНОТАЦІЯ

Дана курсова робота включала в себе здобуття практичних навичок у створенні прикладних програмних додатків, які взаємодіють із базою даних PostgreSQL. Було виокремлено наступні етапи розробки додатку:

- Створення системи отримання/генерації та фільтрації даних
- Створення системи реплікації даних
- Створення системи аналізу даних предметної галузі
- Створення системи резервування/відновлення даних

Результатом виконання курсової роботи стала реалізація усіх пунктів, описаних вище, та отримання кінцевої інформаційно-аналітичної системи, яка виконує аналіз даних, отриманих із зовнішніх ресурсів, та взаємодіє із реляційною СУБД PostgreSQL.

ЗМІСТ

ВСТУП

Основною метою даної курсової роботи є створення інформаційно-аналітичної системи за допомогою мови програмування Python у середовищі розробки PyCharm, яка взаємодіє із базою даних PostgreSQL.

Призначенням програмного додатку є здійснення аналізу температурних показників на основі вимірів отриманих за певний проміжок часу в різних містах світу, з метою відслідковування хронології показників та надання можливості для порівняння показників певних міст в заданому періоді.

Створення інформаційно-аналітичного додатку є актуальною задачею, оскільки він дозволяє значно прискорити, а також забезпечити процес аналізу великих обсягів даних, неможливий для здійснення людиною.

У процесі розробки було виокремлено наступні основні етапи:

- Створення системи генерації/отримання та фільтрації даних, основним призначенням якої є генерація/отримання даних для подальшої їх фільтрації та аналізу.
- Створення системи реплікації даних, яка передбачає розділення функціональності отримання даних та їх передачі програмному додатку, для розподілення навантаження на базу даних та зменшення можливості втрати даних.
- Створення системи аналізу даних, яка необхідна для аналізу отриманих раніше даних та їх візуалізації у вигляді графіків та діаграм.
- Створення системи резервування/відновлення даних, необхідної для збереження стану бази даних та можливості відновлення даних на основі обраного користувачем файлу стану.

Усі ці пункти є важливими для програмного додатку, оскільки забезпечують стабільну роботу системи, а також надійність збереження даних.

Окрім пунктів, наведених вище, варто відзначити і процес підвищення швидкодії отримання даних із бази даних на основі застосування індексів БД, який також було виконано.

І. АНАЛІЗ ІНСТУМЕНТАРІЮ ДЛЯ ВИКОНАННЯ КУРСОВОЇ РОБОТИ.

Розробка даного курсового проекту базувалась на використання такої СУБД, як PostgreSQL. PostgreSQL- це потужна об’єктно-реляційна система керувань баз даних з відкритим кодом, котра використовує та доповнює мову SQL, у поєднанні з багатьма функціями, котрі безперечно зберігають та масштабують складні робочі навантаження даних. [1]. Дана СУБД сумісна з всіма основними операційними системами, також PostgreSQL має багато функцій, для забезпечення захисту цілісності даних та надання можливості керувати даним незалежно від розміру вибірки даних. Дана СУБД я досить розширювальною, адже ви можете визначати власні типи даних, створювати власні функції, та використовувати програмний код різних мов програмування, не перекомпільовуючи саму базу даних. [1]

Мова програмування – Python 3.8. Ключовим фактором у виборі мови програмування був її широкий спектр застосування у сфері Data Science та Big Data завдяки широкому спектру бібліотек.

Було використано функціонал деяких бібліотек: pandas має досить гнучкий інструментарій та чудово підходить для первинної обробки та аналізу даних, було використано вбудовану потужну структуру для аналізу даних DataFrame (drop – для відкидання стовбців що не несуть цінної інформації для побудови графіків, set_index – для заміни стандартного індексу на більш зручний (використано дату), count_values – для підрахунку значень що повторюються) [2]. Візуалізація даних відбувається з допомогою використання бібліотеки matplotlib, що є досить зручною у використанні разом з DataFrame.

Середовище розробки – PyCharm, оскільки дане середовище забезпечує комфортний процес написання коду, має вбудовані засоби його відлагодження та виконання. Крім того, PyCharm має комфортний інтерфейс для пошуку та встановлення необхідних бібліотек.

Засоби генерації даних – отримання даних виконувалося шляхом здійснення http запитів до Visual Crossing Weather-API (посилання на ресурс було надано у технічному завданні). Даний метод генерації даних забезпечує отримання достовірних даних і отримання адекватної статистики у подальшому при аналізі.

Засоби для роботи з СУБД –sqlalchemy, надає повний набір добре відомих шаблонів корпоративного рівня стабільності, сконструйованих для високопродуктивного доступу до бази даних[3].

ІІ. СТРУКТУРА БАЗИ ДАНИХ

База даних містить у собі наступні таблиці:

- Positions – таблиця з сутностями місць розташування проведення вимірів, містить наступні поля address_id, city, latitude, longitude, country_id.
- Countries – таблиця з сутностями країн, містить наступні поля country_id, country, country_code.
- Indexes – таблиця сутності показників, містить наступні поля city_id, date_id, min_temperature, max_temperature, wind_speed, wind_direction, relative_humidity, presipitation.
- Date_time – таблиця з інформацією про час показника. Поля: date_id, index_date.

Зв`язки між сутностями:

- Countries – Positions: ONE to MANY
- Positions – Indexes: ONE to MANY
- Date_time – Indexes: ONE to MANY

ІІІ. ОПИС ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1. Загальна структура програмного забезпечення

Програмний додаток було реалізовано у консольному варіанті із застосуванням шаблону MVC на базі операційної системи Windows 10.

2. Опис модулів програмного забезпечення

Програмний додаток складається із наступних модулів:

- Model – призначений для отримання даних зі slave-сервера для подальшого їх аналізу
- View – призначений для візуального представлення результатів аналізу даних у вигляді графіків та діаграм
- Controller – виконує збір даних шляхом запитів до модуля model та забезпечує фільтрацію та аналіз зібраних даних.
- Main – ядро програмного додатку.
- Api_data – виконує формування запиту та отримання даних з API.

3. Опис основних алгоритмів роботи

Робота програмного додатку базується на взаємодії модулів та алгоритмах, реалізованих у бібліотеці pandas, а саме використано можливість групування даних за певними параметрами, сортування, обрахунку середнього значення та підрахунку кількості повторів певних показників. Також дані що розміщенні в структурі DataFrame досить легко перетворювати в графіки. У разі виникнення помилок використані можливості перехоплення помилок за допомогою конструкції try…except та виведення відповідного повідомлення.

IV. АНАЛІЗ ФУНКЦІОНУВАННЯ ЗАСОБІВ РЕПЛІКАЦІЇ

PostgreSQL дозволяє реалізацію реплікації за допомогою наступних методів:

- Трансляція файлів (головний сервер працює в режимі постійної архівації змін, тим часом як кожен резервний сервер виконує постійне отримання даних у вигляді WAL файлів від головного сервера).
- Потокова реплікація (працює таким ж самим чином, як і при трансляції файлів, проте резервний сервер може працювати із меншими затримками, ніж у першому випадку. Резервний сервер підключається до головного, який передає для нього потік записів WAL у момент їх додавання, не чекаючи на їх повне заповнення)
- Каскадна реплікація (резервний сервер приймає підключення реплікації та потоки WAL від інших резервних серверів, які виступають посередниками).
- Синхронна реплікація (при такому методі реплікації кожна фіксація транзакції очікує на підтвердження того, що транзакція була записана у журнал транзакції на обох серверах: головному та резервному) [4].

У процесі виконання курсової роботи було обрано другий варіант реалізації реплікації, реалізований за допомогою вбудованих можливостей СУБД.

Перевіримо роботу реплікації при виході з ладу головного сервера:

1. Перевіримо наш резервний сервер на коректність роботи:

```
database@db:~$ sudo -i -u postgres
[sudo] password for database:
postgres@db:~$ psql
psql (11.10 (Ubuntu 11.10-1.pgdg20.04+1))
Type "help" for help.

postgres=# create database test;
ERROR:  cannot execute CREATE DATABASE in a read-only transaction
postgres=#
```

Як бачимо, можливість створення нової бази даних на ньому заблокована, отже, робота цього сервера коректна.

2. Створимо тригер файл за шляхом /tmp/postgres_rec, як було вказано у файлі recovery.conf при налаштуванні реплікації, та перевіримо роботу резервного сервера вдруге:

```
database@db:~$ sudo su postgres
postgres@db:/home/database$ touch /tmp/postgres_rec
postgres@db:/home/database$ psql
psql (11.10 (Ubuntu 11.10-1.pgdg20.04+1))
Type "help" for help.

postgres=# create database test1;
CREATE DATABASE
postgres=#
```

Резервний сервер почав виконувати роль головного сервера.

3. Після того, як роботу старого головного сервера було налагоджено, виконуємо його конфігурування, як резервного сервера, надаючи йому підключення реплікації до нового головного сервера.

Перевіримо роботу реплікації при виході з ладу допоміжного сервера:

За відсутності з’єднання з реплікою відбувається додаткове з’єднання з мастером.

```
D:\course>C:/Users/костя/AppData/Local/Microsoft/WindowsApps/python.exe d:/course/main.py
Error connection with PostgreSQL replica, use of master power
------------------------------------------------------
1) Work with some city
2) Work with some country
3) Compare some city
4) Compare some country
5) Chenge info
6) Make prognoze indexes
7) Make database backup
8) Restore database
9) Exit
------------------------------------------------------
```

V. АНАЛІЗ ФУНКЦІОНУВАННЯ ЗАСОБІВ РЕЗЕРВУВАННЯ/ВІДНОВЛЕННЯ ДАНИХ

СУБД PostgreSQL пропонує наступні методи реалізації резервування та відновлення даних:

- Використання періодичного резервного копіювання за допомогою вбудованої утиліти pg_dump.
- Резервне копіювання на основі базових копій та архівів WAL.

Перший метод є достатньо швидким і простим у реалізації, проте забезпечує виконання відновлення лише за наявності backup-файла, тим часом як другий метод є складнішим у реалізації, потребує більше ресурсів та пам'яті [5].

У процесі розробки програмного додатку було застосовано перший метод резервування та відновлення, що забезпечує вибір користувачем файлу резервної копії та відновлення у будь-який із раніше збережених станів.

У програмному додатку резервне копіювання та відновлення можуть бути використані у разі виникнення наступних ситуацій:

- У випадку втрати як даних бази, так і самої бази
- При виникненні потреби відновити один з попередніх станів сховища даних, збережених раніше

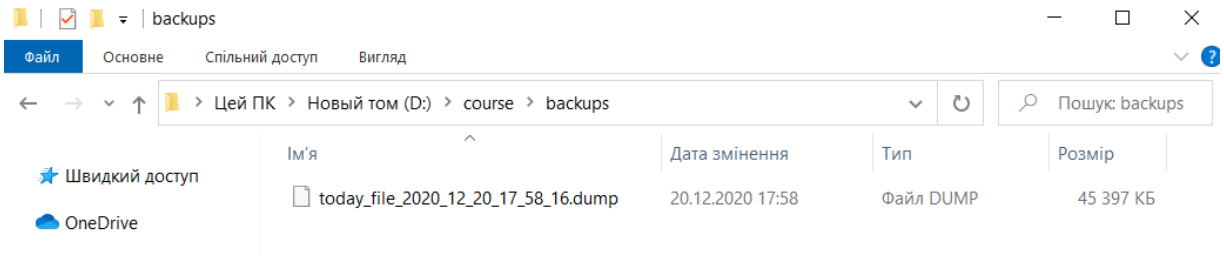
Наведемо приклади використання описаної функціональності:

1. Створимо файл відновлення поточного стану бази даних:

```
D:\course>C:/Users/костя/AppData/Local/Microsoft/WindowsApps/python.exe d:/course/main.py
-----------------------------------------------------
1) Work with some city
2) Work with some country
3) Compare some city
4) Compare some country
5) Chenge info
6) Make prognoze indexes
7) Make database backup
8) Restore database
9) Exit
-----------------------------------------------------

Enter command: 7
Enter name file: today_file
Successful backup! spend time: 3.612121820449829 seconds
-----------------------------------------------------
```

При перевірці директорії із файлами відновлення бачимо, що відповідний файл було створено:



2. Тепер виконаємо відновлення, задля цього в коді є спеціальний скрип котрий видаляє та поновлює нашу базу даних:

   Запустимо відновлення наших даних за допомогою програми:

```
------------------------------------------------------
1) Work with some city
2) Work with some country
3) Compare some city
4) Compare some country
5) Chenge info
6) Make prognoze indexes
7) Make database backup
8) Restore database
9) Exit
------------------------------------------------------

Enter command: 8
                              filename
0  today_file_2020_12_20_17_58_16.dump
1                    close this window
Enter file index: 0
Connection closed
Database "course" will be permanently removed.
Are you sure? (y/n) y
SELECT pg_catalog.set_config('search_path', '', false)
DROP DATABASE course;
SELECT pg_catalog.set_config('search_path', '', false)
CREATE DATABASE course ENCODING 'UTF-8';
Successful restore!, spend time:  3.237952709197998 seconds
------------------------------------------------------
1) Work with some city
2) Work with some country
3) Compare some city
4) Compare some country
5) Chenge info
6) Make prognoze indexes
7) Make database backup
8) Restore database
9) Exit
------------------------------------------------------
```

Дані успішно відновлено.

VI. АНАЛІЗ РЕЗУЛЬТАТІВ ПІДВИЩЕННЯ ШВИДКОДІЇ ВИКОНАННЯ ЗАПИТІВ

SELECT запити мають нижчу швидкодію при збільшенні обсягів даних та фільтрації за полями, які не входять до складу PK. Це спричинено тим, що у такому випадку виконується послідовне сканування усіх даних, що є досить неефективним. Саме тому у таких випадках (в середньому при наявності більш ніж 10000 рядків сутностей) застосовується індексація.

У таблиці Date_time було використано Brin індекс для поля index_date формату timestep, внаслідок чого вдалося отримати приріст швидкодії при отриманні показників з певного періоду. У середньому, час планування операції зазнав або незначного приросту, або зменшення, тим часом як час виконання операції став у середньому меншим на 4 мілісекунди. Візуальне представлення цих даних було наведено у додатку.

VII. ОПИС РЕЗУЛЬТАТІВ АНАЛІЗУ ПРЕДМЕТНОЇ ГАЛУЗІ

Аналіз даних предметної галузі було виконано для наступних сценаріїв:

- Аналіз температурних показників, вимірів швидкості та напрямку вітру, відносної вологості та кількості опадів в обраний період часу для міст та країн. Вивід у вигляді стовбчастих діаграм найуживаніших показників.
- Вивід діаграм котрі характеризують різницю вимірів між певними містами в обраний період часу з додатковими графіками що вказую найуживаніше значення різниці.
- Аналітичне прогнозування майбутніх температурних показників для визначеної країни з урахуванням попередніх показників на визначеному проміжку.

На осі абсцис здебільшого зображено часові проміжки, тільки в допоміжних графіках на осі абсцис розміщено числову шкалу.

На осі ординат розміщено назви вимірів такі як: градуси Цельсія для вимірів температури, мм для кількості опадів та градусна міра для напрямку вітру. Візуальне представлення отриманих результатів аналізу наведено у додатку.

ВИСНОВКИ

Підіб`ємо підсумки по кожному з етапів виконання курсовою роботи:

- Для виконання курсової роботи було підібрано вдалий інструментарій у вигляді різних програмних бібліотек та програмного забезпечення, що дало змогу реалізувати усі програмні пункти роботи.
- Структуру бази даних було спроектовано таким чином, щоб вона відповідала необхідним нормальним формам та забезпечувала найоптимальніший шлях для отримання даних із зовнішнього API.
- Програмний додаток було реалізовано за допомогою мови програмування Python 3.8 на базі операційної системи Windows 10. Програмний код відповідає шаблону проектування MVC.
- Було проаналізовано усі засоби реплікації, доступні у СУБД PostgreSQL, внаслідок чого механізм реплікації було реалізовано за допомогою потокового методу, що забезпечує швидку асинхронну взаємодію між головним та резервним серверами. Тестування реплікації показало, що вона працює належним чином та забезпечує швидкий процес налагодження системи у разі відмови головного сервера.
- Внаслідок аналізу методів резервування/відновлення було обрано метод, який забезпечує менше використання ресурсів системи та дозволяє вибір точки відновлення серед створених користувачем раніше. У відповідному пункті було проаналізовано роботу відповідного механізму та підтверджено його коректну роботу.
- За допомогою механізму індексації в PostgreSQL вдалося отримати підвищення швидкодії виконання запитів.
- Було реалізовано механізм аналізу та візуалізації даних у вигляді діаграм та графіків за допомогою зовнішніх бібліотек мови програмування Python, таких як pandas, matplotlib, що забезпечує швидку роботу даними та їх аналіз.

- Засоби генерації та фільтрації даних було реалізовано за допомогою звернень до зовнішнього серверу даних Visual Crossing Weather-API, що дозволяє отримання достовірних для обраної предметної галузі даних.

У процесі виконання курсової роботи було набуто практичних навичок у створенні програмних додатків мовою програмування Python, які взаємодіють із реляційною СУБД, здобуто практичний дослід технологіях аналізу даних.

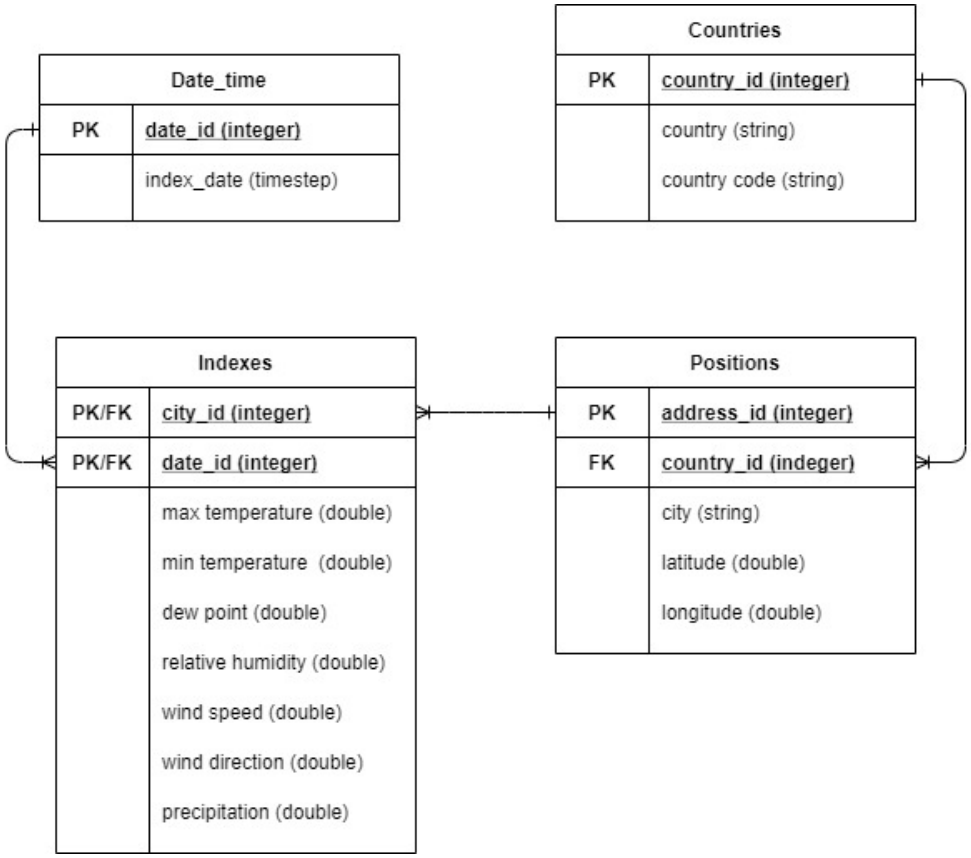
Відповідно до наведених виконаних пунктів роботи можна стверджувати, що основна мета роботи (створення інформаційно-аналітичної системи для аналізу температурних показників) була досягнута – додаток готовий до використання.

ЛІТЕРАТУРА

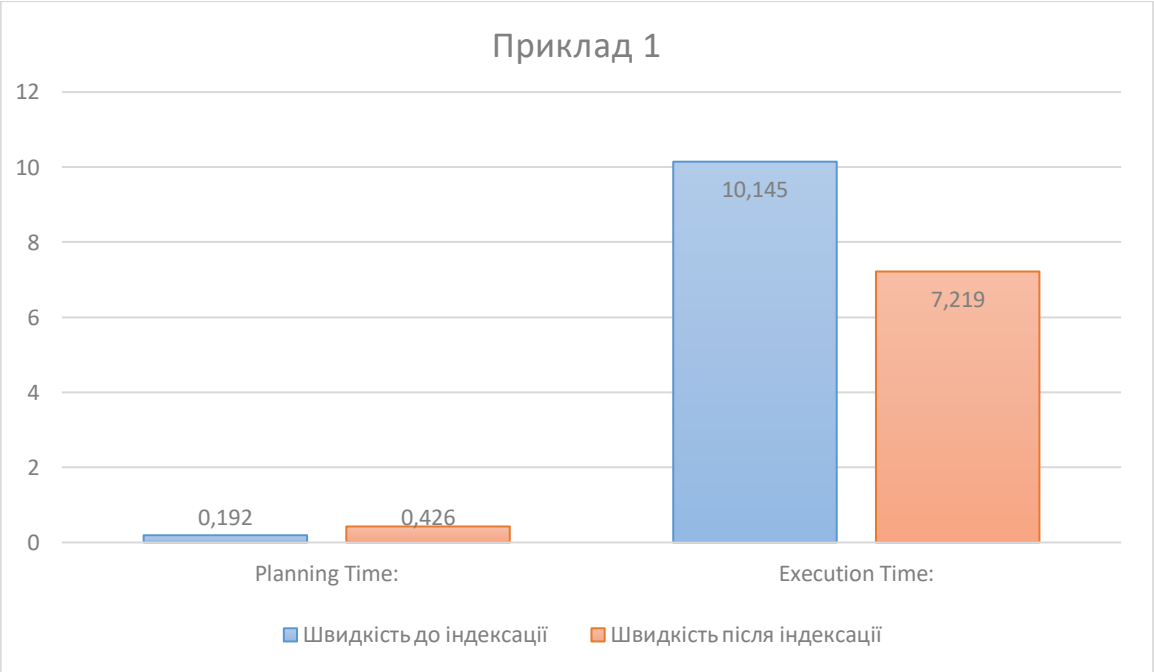
1. PostgreSQL [Електронний ресурс] – Режим доступу до ресурсу: https://www.postgresql.org/about/
2. Введение в pandas: анализ данных на Python [Електронний ресурс] – Режим доступу до ресурсу: https://www.freecodecamp.org/news/the-ultimate-guide-to-the-pandas-library-for-data-science-in-python/
3. SQLAlchemy [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/SQLAlchemy.
4. PostgreSQL: Документация [Електронний ресурс] – Режим доступу до ресурсу: https://postgrespro.ru/docs/postgresql/9.6/warm-standby.
5. Резервное копирование и восстановление в PostgreSQL [Електронний ресурс] – Режим доступу до ресурсу: https://habr.com/ru/post/178567/.

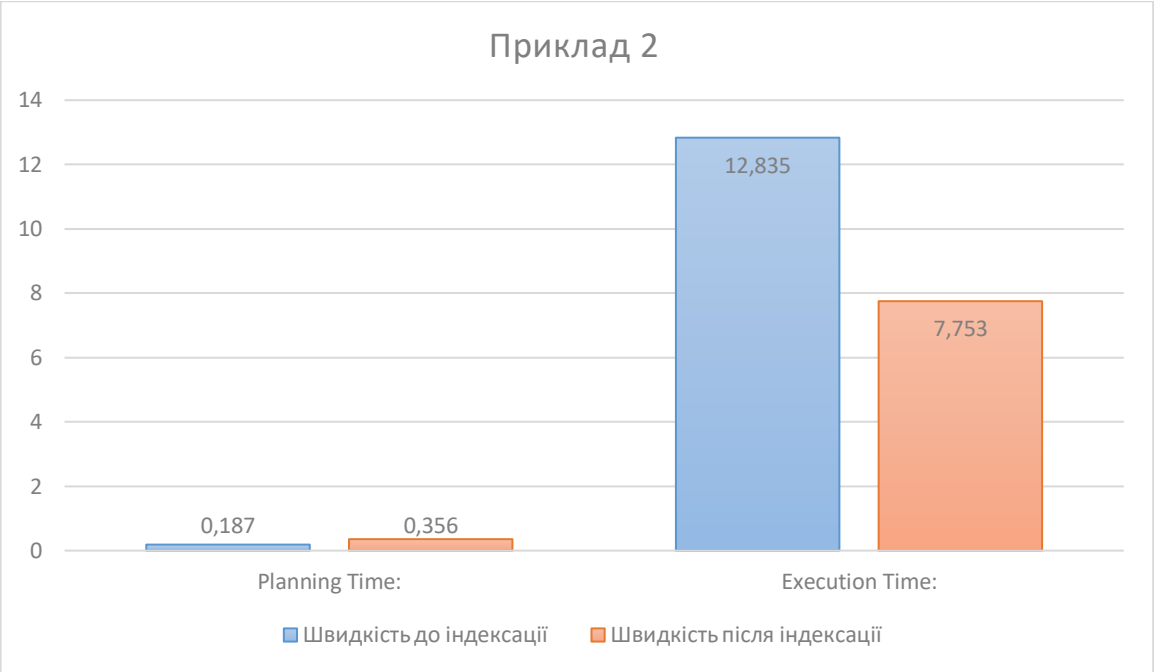
ДОДАТКИ

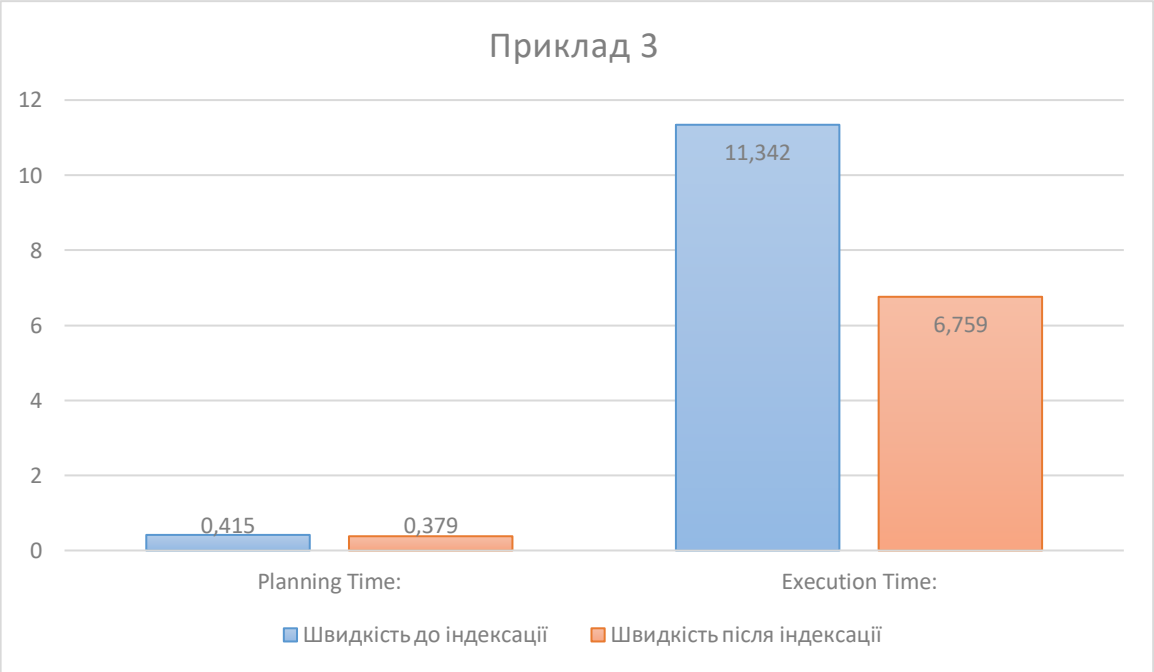
Структура бази даних:



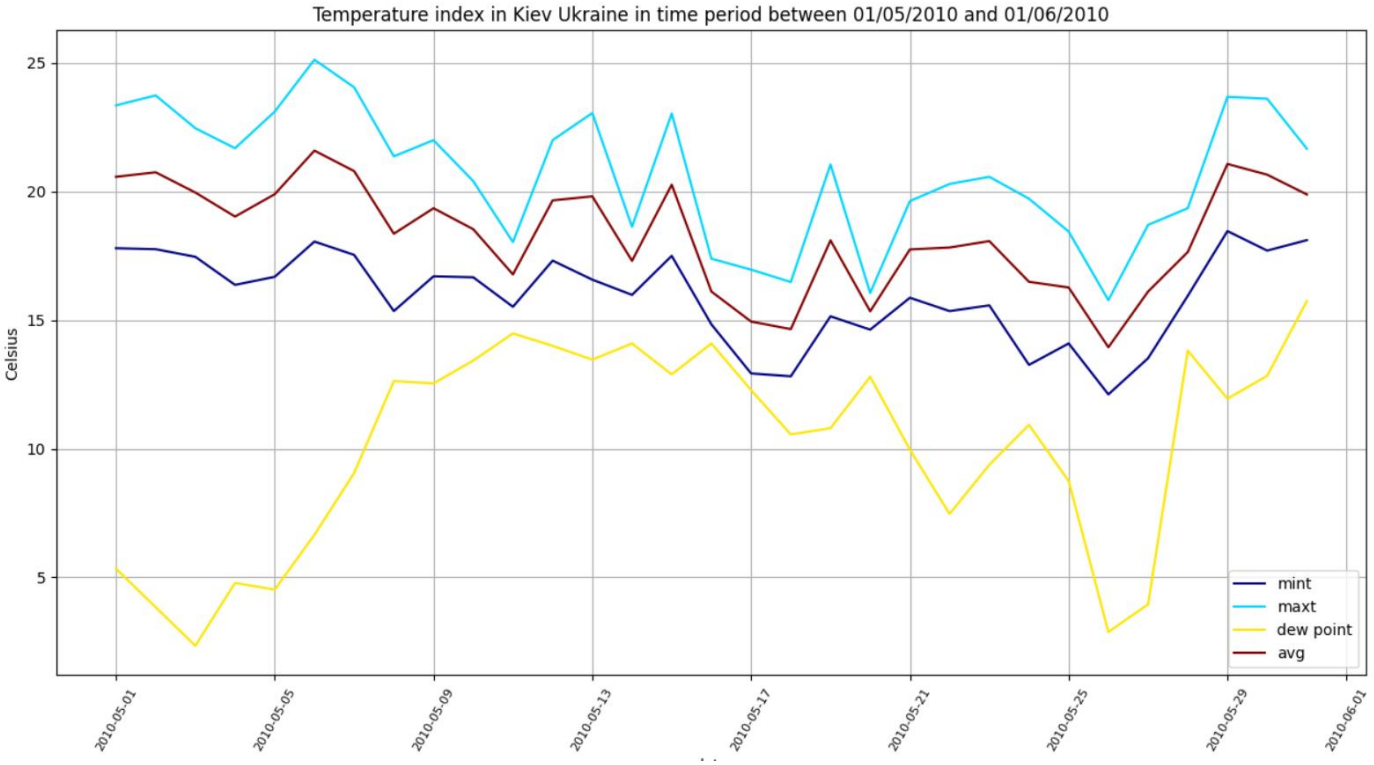
Результати підвищення швидкодії:



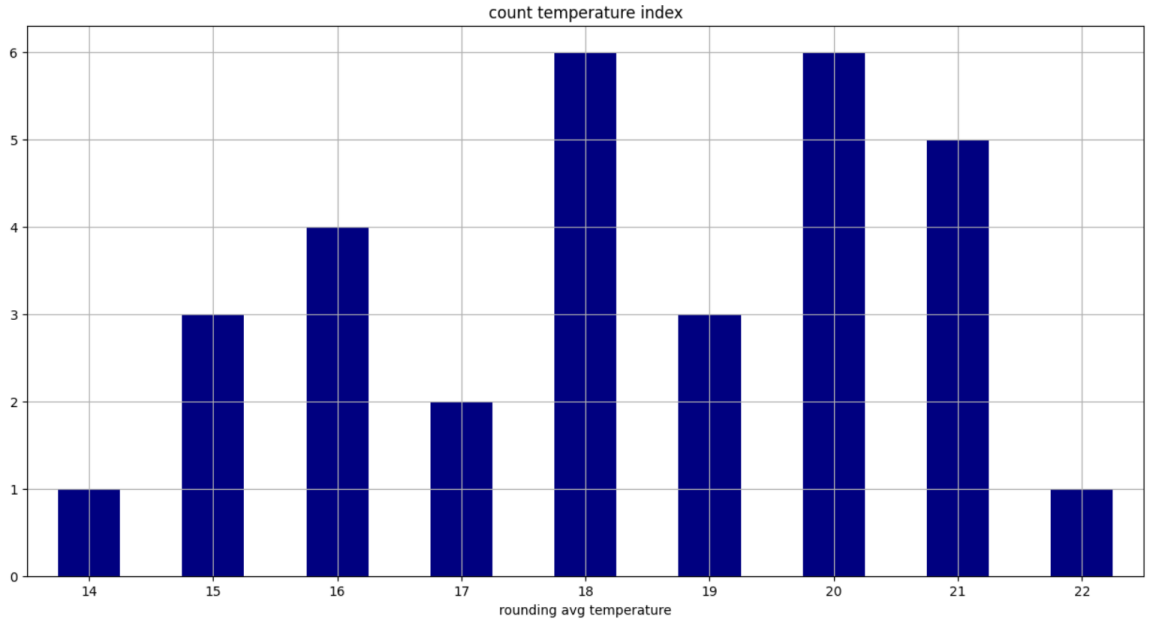

Приклад 2
14
12
10
8
6
4
2
0
12,835
7,753
0,187
0,356
Planning Time:
Execution Time:
Швидкість до індексації
Швидкість після індексації

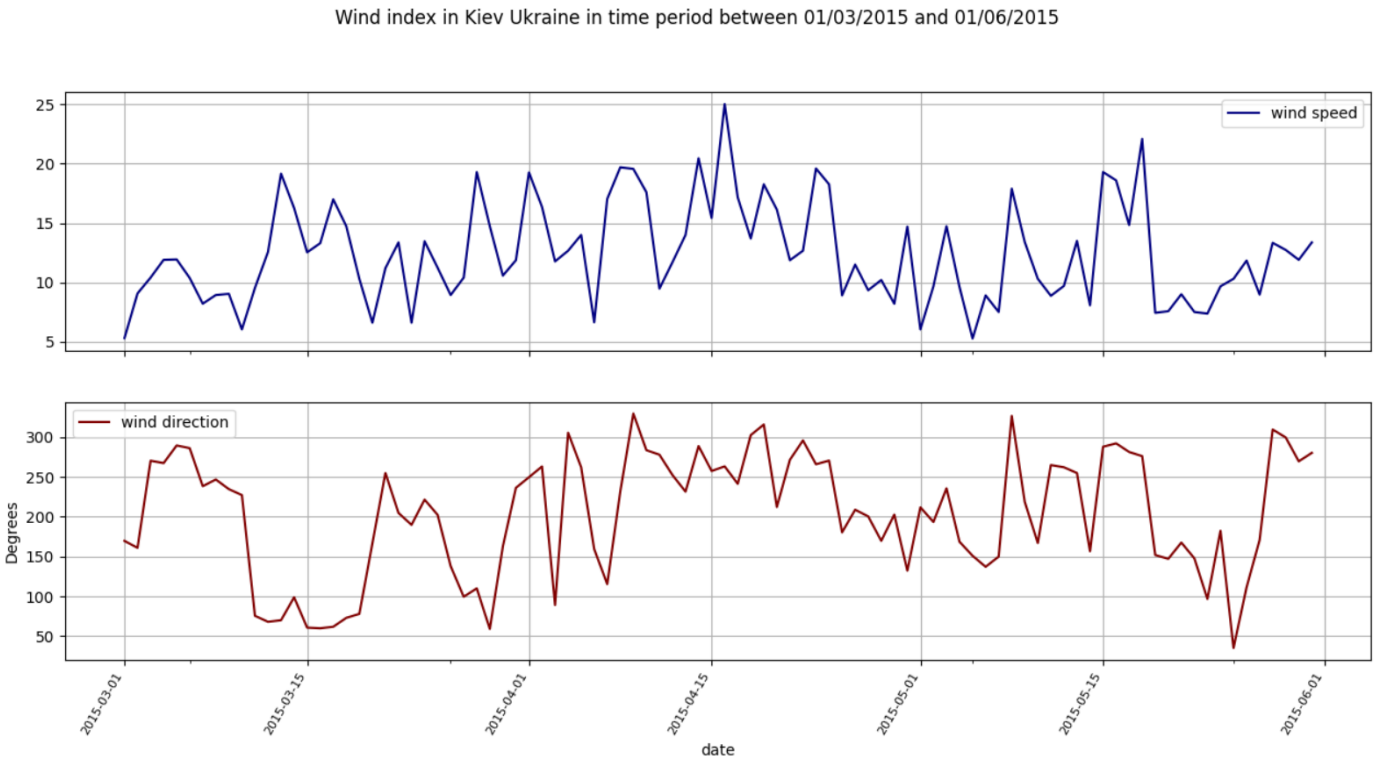

Приклад 3
12
10
8
6
4
2
0
11,342
6,759
0,415
0,379
Planning Time:
Execution Time:
Швидкість до індексації
Швидкість після індексації

Результати аналізу даних:

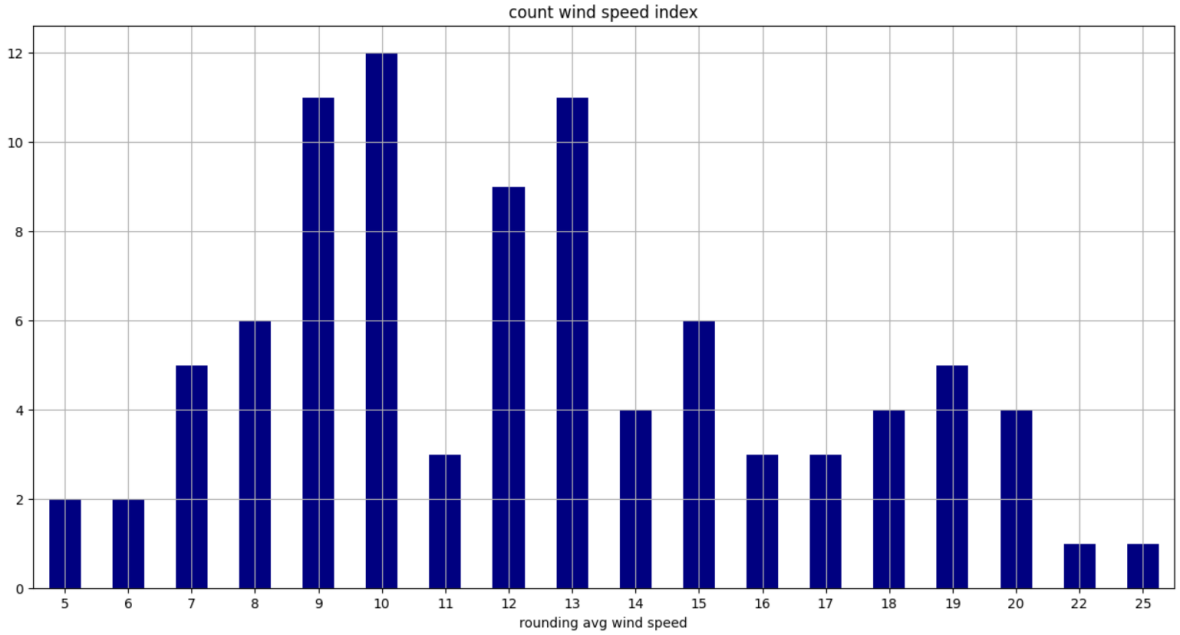


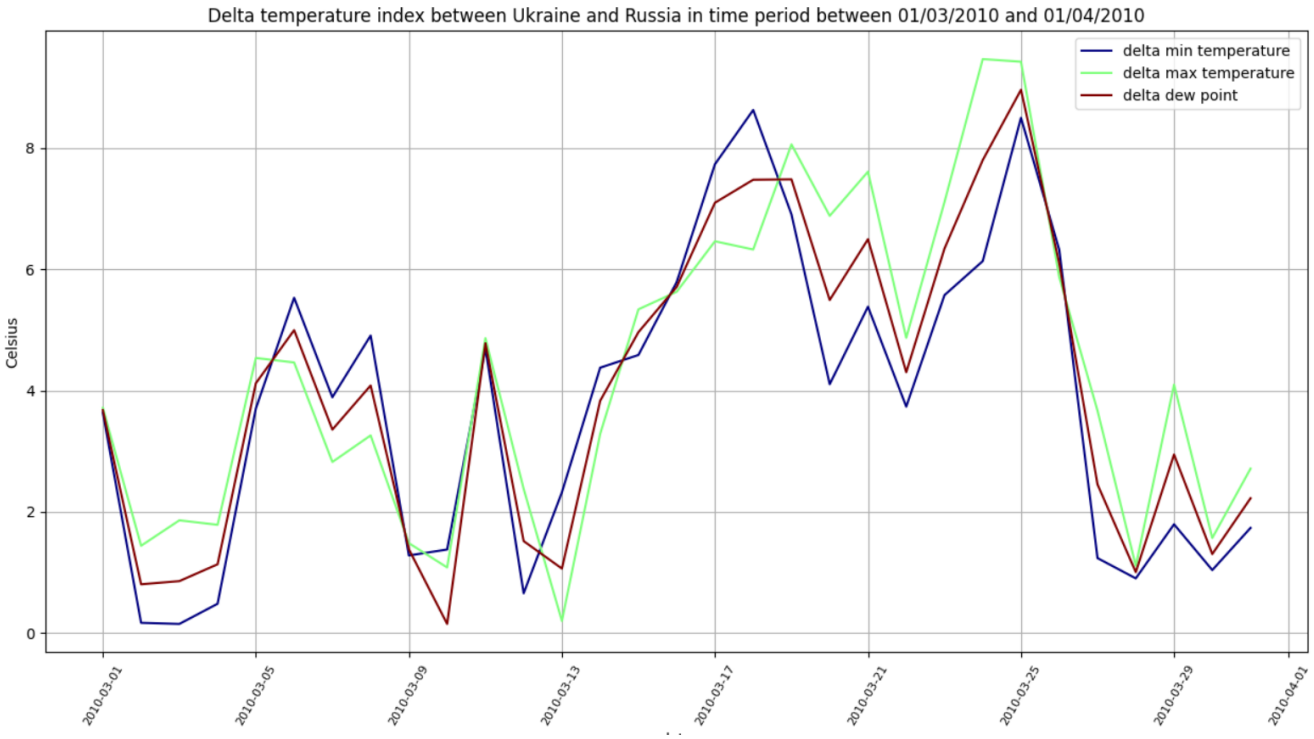
Допоміжний графік



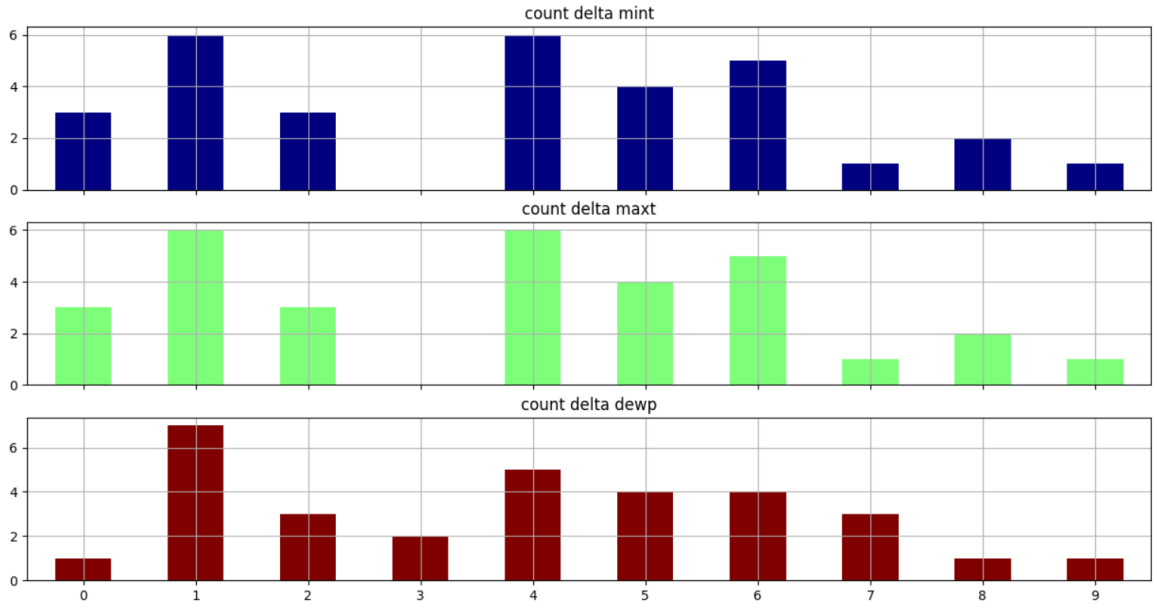


Допоміжний графік





Допоміжний графік



Approximate indexes for Ukraine in next year based the enter data (2021-04-01 00:00:00->2021-06-01 00:00:00)