Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп'ютерних наук
(повна назва)

Кафедра програмної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Дослідження методів створення та супроводу мережевих баз даних
(тема)

Виконав:
студент (ка) 2 курсу, групи ІПЗм-22-6

Коберник К.С.
(прізвище, ініціали)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)
Тип програми освітньо-наукова

Керівник проф. Шубін І. Ю.
(посада, прізвище, ініціали)

Допускається до захисту
Зав. кафедри _____ З.В.Дудар
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерних наук
Кафедра програмної інженерії
Рівень вищої освіти другий (магістерський)
Спеціальність 121 – Інженерія програмного забезпечення
Тип програми освітньо-наукова програма
Освітньо-наукова програма 121 Інженерія програмного забезпечення
(шифр і назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«___» _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Коберник Карену Сергійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи «Дослідження методів створення та супроводу мережевих баз даних»

Затверджена наказом по університету від 29.03.2024 р. № 250 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 14.06.2024 р.

3. Вихідні дані до роботи: *опис досліджуваних мережевих баз даних, вимоги до розробки архітектури бази даних для проведення досліджень за обраною предметною областю, мови програмування Python, середовища розробки PyCharm*

4. Перелік питань, що потрібно опрацювати в роботі: *провести дослідження існуючих мережевих баз даних та обрати технології і програмні рішення які будуть актуальними для реалізації гнучкого програмного застосунку та аналіз методів інтеграції мережевих баз даних, збудувати архітектуру програмного забезпечення з урахуванням визначеної бізнес моделі, дослідження методів створення нереляційних баз даних у вигляді застосунків, створення програмного забезпечення орієнтованого на реалізацію нереляційних баз даних за мережевою моделлю даних*

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Аналіз предметної галузі та постановка задачі	03 квітня 2024 р.	*виконано*
2	Огляд існуючих методів	10 квітня 2024	*виконано*
3	Розробка алгоритмів, проектування та розробка ПЗ	15 квітня 2024	*виконано*
4	Підготовка пояснювальної записки	20 квітня 2024 р.	*виконано*
5	Спецчастина	28 квітня 2024 р.	*виконано*
6	Підготовка презентації та доповіді	03 травня 2024 р.	*виконано*
7	Попередній захист	05 травня 2024 р.	*виконано*
8	Нормоконтроль, рецензування	07 червня 2024	*виконано*
9	Занесення роботи в електронний архів	08 червня 2024	*виконано*
10	Допуск до захисту в зав. кафедри	11 червня 2024 р.	*виконано*

Дата видачі завдання 30 березня 2024 р.

Студент _____ Коберник К.С.
(підпис)

Керівник роботи _____ проф.Шубін І.Ю.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ / ABSTRACT

Пояснювальна записка містить: 52 с., 10 рис., 17 джерел, 5 додатків.

МЕРЕЖЕВІ БАЗИ ДАНИХ, МЕТОДИ СТВОРЕННЯ ТА СУПРОВОДЖЕННЯ МЕРЕЖЕВИХ БАЗ ДАНИХ, NOSQL БАЗИ ДАНИХ, НЕРЕЛЯЦІЙНІ БД У ВИГЛЯДІ ГРАФІВ.

Об’єктом дослідження – є методи створення та супроводу мережевих баз даних.

Метою роботи є дослідження методів та програмного забезпечення для покращення реалізації СУБД мережевих баз даних для обробки великої кількості даних та створення прототипу мережевої бази даних з оптимізацією для аналізу.

У результаті роботи були ідентифіковані засоби розробки та супроводу мережевих NoSQL баз даних та модель СУБД з покращеними методами супроводу баз даних, було розроблено застосунок для пере опрацювання SQL даних у NoSQL базу з оптимізацією для аналізу та збору статистики.

NETWORK DATABASES, METHODS OF CREATING AND MAINTENANCE OF NETWORK DATABASES, NOSQL DATABASES, NON-RELATIONAL GRAPH DATABASES.

The purpose of the work is to research methods and software for improving the implementation of DBMS of network databases for processing a large amount of data and creating a prototype of a network database with optimization for analysis.

As a result of the work, tools for the development and maintenance of network NoSQL databases and a DBMS model with improved database maintenance methods were identified, an application was developed for processing SQL data into a NoSQL database with optimization for analysis and statistics collection.

Я, Коберник Карен Сергійович, студент гр. ІПЗм-22-6, здобувач вищої освіти на другому (магістерському) рівні кафедри «Програмна інженерія», заявляю: моя кваліфікаційна робота на тему «Дослідження методів створення та супроводу мережевих баз даних.», що буде представлена в екзаменаційну комісію для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в електронному архіві відкритого доступу ElArKhNURE. Всі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлений з діючим положенням «Про протидію академічному плагіату в ХНУРЕ», згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

SQL – Structured Query Language

МБД – Мережева База Даних

СУБД – Система Управління Базою Даних

ЦП – Центральний Процесор

БД – База Даних

API – Application Programming Interface

HTTP – Hypertext Transfer Protocol

URL – Uniform Resource Locator

TCP – Transmission Control Protocol

ВСТУП

В умовах стрімкого темпу розвитку інформаційного суспільства важливим напрямком наукових досліджень та практичних застосувань є дослідження методів створення та супроводу мережевих баз даних (МБД).

Сучасна тенденція зберігання та обробки великих обсягів інформації визначається постійним розвитком та покращенням комп'ютерних технологій, що підвищує попит розробки ефективних та надійних рішень для управління та підтримки цих баз даних. У даному контексті, дослідження спрямоване на аналіз та розробку оптимальних методів створення мережевих баз даних, які забезпечать ефективне управління ресурсами та швидкий доступ до інформації. Також розглядаються методи супроводу, включаючи аналіз стійкості, механізми резервування та забезпечення безпеки даних.

Актуальність створення нових програмних засобів мережевих баз даних обумовлюється тим що з ходом часу попередні застосунки стали застарілими за технологіями та оптимізацією, тому якщо створити гнучкий додаток який реалізує мережевий підхід до роботи з даними то з'являється можливість позбутися всіх недоліків мережевих баз даних та залишити всі переваги швидкості та економії пам'яті.

Дослідження включає в собі вивчення наявних стандартів та протоколів для мережевих баз даних, а також визначення оптимальних інструментів для їх гнучкої реалізації. Додаткова увага приділяється особливостям роботи з великими обсягами даних та потребам у зміні самої структури мережевої бази даних користувачем.

Мета даного дослідження полягає в розробці нових методів та інструментів для створення та супроводу мережевих баз даних, які відповідають вимогам сучасного інформаційного середовища.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Аналіз пріоритетів при створенні МБД

Створення баз даних за мережевою моделлю включає в себе ряд аспектів, які визначають якість, продуктивність та зручність використання таких баз.

Буде проведено аналіз аналогічних систем управління базами даних (СУБД) де реалізована ієрархічна модель даних для застосування мережевих баз. Популярні мережеві СУБД включають CODASYL DBMS [1], Neo4j[2], IMS (Information Management System).

Ось ключові аспекти, що варто розглядати:

- мови запитань (декларативні та імперативні мови): важливо вибрати мову запитань, яка забезпечить зручний і ефективний доступ до даних. В багатьох мережевих СУБД використовують декларативні мови, які описують, які дані потрібно отримати, тому дуже важливим рішенням буде створити мову запитів аналогічну тим що існують на ринці але додати розширенні функції налаштування методів запиту;
- масштабованість (горизонтальна та вертикальна масштабованість): забезпечення ефективної роботи з базою даних при збільшенні обсягу даних є критичним. Важливо вибрати рішення, яке дозволяє масштабувати базу даних горизонтально (додавання нових серверів) та вертикально (збільшення потужності наявних серверів);
- моделювання взаємозв'язків (використання вказівників у обидві сторони): Однією з ключових особливостей мережевої моделі є здатність висловлювати складні взаємозв'язки між записами. Важливо враховувати, наскільки ефективно створена СУБД буде їх встановлювати;
- підтримка транзакцій (атомарність, консистентність, ізоляція, прочність (ACID)): забезпечення надійності операцій та відновлення бази даних до стану до виникнення помилок є важливим аспектом створення мережевих баз даних;

- підтримка розподілених систем (множинні сервери та географічна розподіленість): якщо проект вимагає роботи з розподіленими даними на різних серверах або географічних регіонах, важливо врахувати можливості СУБД в цьому аспекті;
- підтримка мов програмування (інтеграція з популярними мовами): можливість взаємодії з базою даних з різних мов програмування може бути важливою для розробників;
- зручність розробки і адміністрування (інструменти розробки та адміністрування): наявність зручних інструментів для розробки та адміністрування спрощує життя розробників та адміністраторів;
- візуалізація та структуризація самої бази даних в додатку для користувача: через використання графової моделі бази даних для мережевого методу – потрібно розробити простий інтерфейс візуалізації цих графів що буде набагато легше ніж розробка аналогічного для моделі дерева.

Буде проаналізовано сфери застосування МБД щоб зрозуміти перспективи їх розвитку та масштаби попиту. Мережеві бази даних застосовуються у таких галузях:

Соціальні мережі: мережеві бази даних виявляють велике значення в соціальних мережах. Вони використовуються для аналізу графових структур соціальних взаємин. МБД дозволяють ефективно виявляти та обробляти складні взаємозв'язки (наприклад рекомендації реклами) що є важливим для підтримки функціоналу соціальних мереж.

Телекомунікації: у галузі телекомунікацій МБД використовуються для зберігання інформації про мережеву інфраструктуру та для управління послугами для абонентів. Вони дозволяють ефективно моделювати зв'язки між абонентами, оптимізувати маршрути та забезпечувати якість обслуговування.

Наукові дослідження: у сфері наукових досліджень МБД використовуються для моделювання складних систем, аналізу взаємозв'язків у великих наукових спільнотах та для управління даними в галузі досліджень. Завдяки здатності

працювати з великим обсягом різноманітних даних, МБД допомагають вченим зосереджуватися на важливих аспектах своєї роботи та економити дорогоцінний час нариклад як використання у біомедицині[2] де за одним запитом по діагнозу покажчики мають зв'язок з усіма симптомами та методами лікування.

Логістика та ланцюги постачання: у галузі логістики та постачання МБД використовуються для відстеження та оптимізації різних ланцюгів постачання. Вони дозволяють ефективно керувати взаємозв'язками між виробниками, постачальниками та споживачами, що сприяє зниженню витрат та підвищенню ефективності.

Використання МБД з NoSQL: у сучасному середовищі NoSQL бази даних стають все популярнішими, особливо для роботи з великими обсягами неструктурованих даних. Мережеві бази даних у NoSQL використовують графову модель для представлення та зберігання даних (див. рис. 1.1).

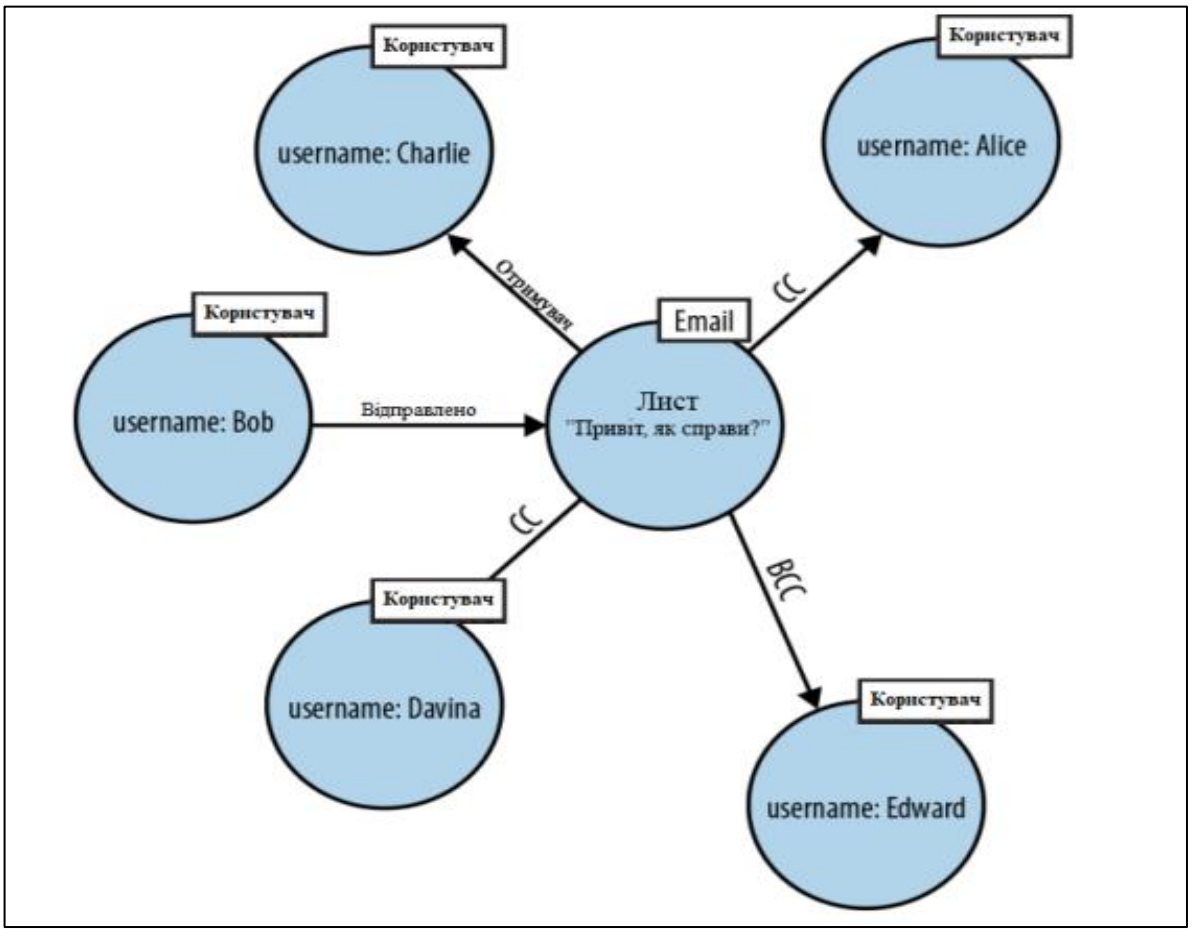

Рисунок 1.1 – Схема зберігання листа Email у графах (рисунок виконаний самостійно)

Це дозволяє зв'язувати дані і створювати складні взаємозв'язки між ними [3]. Прикладом може слугувати графова база даних Neo4j, яка використовує мережеву модель для представлення графових структур.

Переваги використання NoSQL:

– гнучкість у представленні та моделюванні складних взаємозв'язків;
– здатність ефективно опрацьовувати великі обсяги графових даних;
– можливість швидкого доступу до пов'язаних об'єктів через зв'язки в графі.

Недоліки використання МБД та NoSQL:

– складність в налаштуванні деяких типів запитів;
– складність на початку розгортання для окремої програмної системи.

Більшість систем NoSQL розроблено для агрегації даних[4], групування інформації на основі певних критеріїв і типу бази даних (наприклад, зберігання документів або пари ключ-значення). Цей дизайн підтримує базові обмежені операції, забезпечуючи єдине специфічне представлення даних. Зосереджуючись на агрегуванні одного набору даних за раз, ці системи сприяють розподілу численних сегментів даних у мережі комп’ютерів, дотримуючись колективного виміру (наприклад, документ у документоорієнтованих базах даних). Однак цей підхід вимагає обчислення інших уявлень і аналізів шляхом обробки або копіювання даних.

1.2 Аналіз веб-застосунків з використанням МБД у вигляді графів

Мережеві бази даних (МБД) в науково-дослідницьких системах є потужним інструментом для зберігання та аналізу складних взаємозв'язків між різними об'єктами та явищами. Вони дозволяють ефективно моделювати та вивчати структури та взаємини в різних наукових галузях, сприяючи розвитку нових знань та вдосконаленню наукових досліджень.

Основні користувачі систем МБД є:

– дослідники та вчені: взаємодіють з МБД для доступу до наукових даних та аналізу результатів досліджень;

- аналітики та спеціалісти з обробки даних: займаються витягуванням знань з об'ємних наукових даних, розробкою аналітичних моделей;
- інженери та розробники: забезпечують підтримку та розробку інфраструктури для МБД в рамках наукових проектів;
- адміністратори даних: відповідають за безпеку та ефективність роботи систем МБД в наукових лабораторіях;
- звичайні користувачі: продивляються відкриті реєстри для знайомства з досягненнями науки та техніки;
- викладачі: ознайомлюють студентів з результатами досліджень для освітнього процесу.

Можна побачити, що кількість цільових груп дуже велика та цільові групи не поділяються за віком, що напряму залежить від розміру самих баз даних – які в нашому випадку великі та будуть тільки збільшуватись.

Проаналізовано також схожості в взаємодії користувачів із системами МБД:

- аналітика та обробка даних: користувачі активно використовують аналітичні інструменти для витягування значущих результатів із складних даних;
- моделювання та симуляція: інженери та вчені використовують системи МБД для моделювання складних систем та сценаріїв;
- взаємодія з багатовимірними даними: користувачі мають можливість працювати з великою кількістю різнорідних даних та побудовою графіків взаємозв'язків.

Більшість з застосувань мережевих баз даних з графовим методом обрали його над деревоподібними та реляційними методами через його переваги в роботі з великими кількостями даних.

Однією з ключових переваг баз даних з графовим методом є їх здатність налагоджувати зв’язок у дві сторони як від батька до нащадка так і у іншу сторону що дозволяє використовувати таку БД зі складними мережевими структурами. У таких базах даних візуалізовано інформація представлена у вигляді графу, де вершини відображають об'єкти, а ребра - взаємозв'язки між ними. Це особливо

актуально для представлення та аналізу даних у сферах, де важливі взаємодії та залежності, наприклад.

Ще однією вагомою перевагою графових баз даних є їхня здатність ефективно виконувати запити та операції з даними, пов'язаними з графовою структурою.

У порівнянні з традиційними реляційними базами даних, графові можуть швидко знаходити шляхи між вершинами, визначати групи об'єктів та виконувати аналітику на основі графової структури. Застосування графового методу визначається не лише його зручністю для моделювання та аналізу, але і високою ефективністю при обробці великих обсягів даних.

Сучасні графові бази даних використовують оптимізовані алгоритми, які дозволяють швидке виконання запитів і відсутність перегрузки при роботі з великими графовими структурами. Завдяки своїй ефективності та виразності - графовий метод стає обраною стратегією для тих, хто працює з даними, де ключовим є вивчення та розуміння взаємозв'язків.

Від соціальних мереж до бізнес-аналітики, графові бази даних відкривають нові можливості для ефективного управління та аналізу складних систем, дозволяючи виявляти та використовувати різноманітні взаємозв'язки в інформації а проста структура графів призводить до набагато простіших і виразніших моделей даних, ніж ті, що створюються за допомогою традиційних реляційних чи інших баз даних NoSQL.

Після пошуку конкретних застосунків Графових баз даних у популярних платформах було також досліджено широке використання таких баз у системах рекомендації даних за пошуком або за попередніми налаштуваннями уподобань інформації – що також поширює імовірне застосування таких баз даних.

LinkedIn є яскравим прикладом сучасної програми, яка використовує бази даних графів. Зокрема, LinkedIn використовує свою графічну базу даних LIquid для автоматизації індексування та доступу в реальному часі до зв’язків між членами, школами, навичками, компаніями, посадами, роботами, подіями тощо.

Цей граф, відомий як економічний граф, містить понад 270 мільярдів ребер і обробляє робоче навантаження в 2 мільйони запитів на секунду[5].

Систему рекомендацій LinkedIn «люди, яких ви можете знати» (PYMK) було перенесено на LIquid, що призвело до значних покращень запитів за секунду (QPS), затримки та використання ЦП. Система LIquid розроблена для масштабування в десять разів порівняно з поточним розміром, пропонуючи високу доступність і ефективний доступ до даних і запитів.

1.3 Постановка задачі

Вибір правильних технологій для створення МБД є критичним кроком, який впливає на продуктивність, масштабованість та безпеку системи, враховуючи велику кількість технологій на яких можливо реалізувати таку СУБД для МБД буде дуже складно спрогнозувати усі технології які будуть застосовані впродовж розробки застосунка. Але найнеобхідніші технології та мови програмування зазначити для постановки задачі не проблема.

Після порівняння і аналізу плюсів та недоліків схожих систем було складено такий перелік:

- мережеві бази даних оптимально підходять для моделювання складних взаємозв'язків, де кожен запис може мати кілька батьківських та дочірніх вузлів;
- вони пропонують більш гнучку структуру, ніж реляційні БД, дозволяючи створювати більш складні зв'язки між об'єктами та вносити різноманітну інформацію;
- в деяких випадках, мережеві БД можуть бути ефективнішими при обробці певних типів запитів, особливо тих, що вимагають навігації по складних зв'язках;
- мережеві БД краще підходять для ситуацій, де необхідно управляти дуже складними та взаємопов'язаними даними, які складно представити у реляційних таблицях;

– бази даних на графах забезпечують прямий доступ до даних через покажчики, що може збільшити швидкість доступу в майже усіх сценаріях їх використання;
– мережеві БД зменшують ізоляцію даних, що може поліпшити інтеграцію та обмін даними в середовищі з великою кількістю взаємозв'язків.

1.4 Опис основних методи створення МБД

Після аналізу станів нереляційних баз даних, їх застосування та специфіки використання з урахуванням всіх аналогів на ринці, недоліків та плюсів у порівнянні з реляційними аналогами – було визначено основні частини програмного застосунку:

– додаток який включає у себе стандартні для СУБД розділи(логін реєстрація, особистий кабінет, баланс, розділ з базами даних, сторінку консолі до окремої БД для маніпуляцій);
– розділ керування доступу до різних видів користувачів та їх особистими даними, балансом;
– функції вибору створення шаблону бази даних який надає користувачеві можливість обрати раніше створений прототип налаштованої БД та переробити під свої потреби;
– розділ з навчальним посібником щодо функцій та підказок щодо користування БД;
– розділ “кейси використання” розділ з прикладами реальних кейсів використання СУБД, який допомагає користувачам зрозуміти, як вони можуть ефективно використовувати продукт у своїй роботі;
– скритий розділ керування контентом на сайті та окремим змістом інших розділів;
– зашифроване сховище особистих даних;
– функція налаштування та створення бекап-збереження копії баз даних, особистих даних користувачів та їх журналів дій, ця функція також буде працювати в залежності від кількості відправлених запитів від

користувачів та кількістю операцій, які виконуються у даний час або кількістю завантажених слотів операційної пам’яті на відокремленому сервері;

- сервіс створення списків за запитом користувача у відокремленої консолі для БД;
- сервіс API для поліпшення роботи з застосунком та надання користувачам можливості написання своїх застосунків або ботів для автоматизації редагування баз даних(наприклад користувачі зможуть написати автоматизацію загрузки своїх застарілих звітів з даними у Microsoft Excel та перенесуть автоматично всі дані та залеж6ності до БД);
- функції редагування базами даних з поліпшеними доступами та додатковою аутентифікацією задля забезпечення захисту від видалення/пошкодження БД користувачами які тільки працюють з запитами або почали вивчати додаток.

Коли користувач відкриває веб-сайт у своєму браузері, запускається процес, що включає взаємодію між клієнтськими та серверними технологіями.

Спочатку, вводячи URL або клацаючи по посиланню, браузер надсилає запит на веб-сервер, де знаходиться цільовий сайт через інтернет за допомогою протоколу HTTP (Гіпертекстовий транспортувальний протокол передачі даних) або HTTPS(Той-самий протокол який використовує криптографічні пари відкритих та закритих ключів для додаткового захисту даних).Сам запит надсилається через транспортний протокол TLS[6] який забезпечує захищену передачу даних на сервер. Веб-сервер обробляє запит і надсилає відповідь, яка зазвичай містить HTML-код сторінки, а також CSS-файли для стилізації та JavaScript-файли для додаткової логіки скриптінгу або анімацій на фронт-енд частині. HTML визначає структуру та вміст сторінки, CSS визначає стиль, а JavaScript додає інтерактивні елементи. Після отримання цих даних, браузер починає процес рендерингу сторінки: інтерпретує HTML-код, застосовує стилі, визначені в CSS, та виконує JavaScript-код. Це призводить до створення візуального представлення сторінки, яке відображається на екрані. Залежно від

складності сайту, можуть використовуватися додаткові технології, такі як AJAX для асинхронного завантаження контенту, фреймворки (React, Bootstrap або Angular) для управління інтерфейсом, а також різноманітні бібліотеки та плагіни. Цей процес відображення веб-сайту в браузері є результатом складної взаємодії між браузером, веб-сервером та веб-технологіями, які працюють разом, щоб забезпечити гладке та ефективне відображення контенту.

HTML (HyperText Markup Language) є мовою розмітки, яка використовується для створення структури веб-сторінок.

В основі HTML лежать теги, які диктують браузеру, як відобразити вміст. Кожен тег має свою специфіку і призначення - від визначення заголовків (наприклад, <h1>), параграфів (<p>), зображень (<img>), до більш складних функцій, як створення форм (<form>) чи включення скриптів (<script>).

DOM є програмним інтерфейсом для HTML та XML документів. Він представляє структуру документа у вигляді дерева об'єктів, де кожний вузол дерева відповідає частині документа (наприклад, елементу HTML).

TCP (Transmission Control Protocol) є одним з основних протоколів, що використовуються в Інтернеті для надійної передачі даних між мережевими пристроями.

Протокол HTTP використовує TCP для гарантії того, що дані будуть доставлені надійно і в правильному порядку. TCP діє на транспортному рівні моделі OSI, забезпечуючи управління потоком і корекцію помилок.

Коли дані надходять від клієнта до сервера або навпаки, TCP відповідає за те, щоб усі пакети даних були доставлені правильно та в правильному порядку через реалізацію відновлення втрачених пакетів з даними та неможливість отримати документ якщо якісь дані були втрачені, що є критично важливим для надійності веб-застосунків.

Коли веб-технології почали розвиватися, ресурсами були файли в гіпертекстовому форматі та зображення у файловій системі. Це добре працює лише для статичних інформаційних ресурсів. Було розроблено низку технологій та інтерфейсів для створення динамічних веб-сайтів.

Підключення веб-серверів до баз даних є ключовим аспектом багатьох сучасних веб-додатків, оскільки це дозволяє веб-серверам обробляти запити та надавати динамічний вміст користувачам який вже є у базі даних або зберігати введені дані користувачей для подальшого використання ними. Процес підключення і взаємодії веб-серверів з базами даних включає кілька важливих компонентів та технологій.

Мови програмування на бекенд, де відбувається обробка даних та їх взаємодія з базами даних, часто використовуються такі мови програмування, як:

- JavaScript (Node.js): популярний через свою універсальність, оскільки JavaScript також використовується на фронтенді та для нього написано багато бібліотек та фреймворків для візуалізації та надання контенту інтерактивності або для обробки даних на бекенді;
- Python: завдяки своїй читабельності та великому набору бібліотек майже для будь-якої програмної системи, Python є відмінним вибором для розробки веб-додатків;
- Java: використовується в багатьох великих корпоративних середовищах завдяки своїй масштабованості та надійності;
- PHP: широко використовується для розробки веб-сайтів та веб-додатків, часто у поєднанні з MySQL;
- Ruby (Ruby on Rails): відомий своєю швидкістю розробки та чистотою коду але вже дуже застарілий.

Процес підключення до баз даних включає використання спеціальних бібліотек або драйверів, які дозволяють веб-додаткам взаємодіяти з базами даних. Ці бібліотеки забезпечують API або інтерфейси для виконання запитів до бази даних, отримання відповідей та обробки даних.

У нашому випадку потрібно підключати програмну систему до кількох типів баз даних водночас – до реляційної для менеджменту користувачів, логування їх дій та зберігання особистої інформації та до МБД над якою будуть проводитися всі маніпуляції коли користувачі почнуть створювати свої бази даних та вносити чи редагувати якісь дані.

Під час роботи з NoSQL базами даних, які часто використовуються для великих обсягів даних або даних, що не потребують жорсткої структури, використовуються специфічні методи підключення та інтерфейси[7].

Приклади NoSQL баз даних включають:

- MongoDB: це одна з найпопулярніших документо-орієнтованих NoSQL баз даних. Вона зберігає дані у форматі, схожому на JSON, що робить її дуже гнучкою для роботи з великими обсягами даних;
- Cassandra: ця база даних відома своєю високою масштабованістю та надійністю. Cassandra є колоно-орієнтованою системою, яка добре підходить для обробки великих обсягів даних з високою швидкістю запису та читання;
- Redis: це система управління базами даних типу "ключ-значення", яка часто використовується як система кешування та зберігання сесій у веб-додатках через свою високу швидкість та ефективність;
- Couchbase: подібно до MongoDB, Couchbase є документо-орієнтованою NoSQL базою даних. Вона пропонує гнучкість та масштабованість для розподілених веб-додатків і часто використовується в мобільних застосунках;
- Apache HBase: ця база даних є нереляційною колоно-орієнтованою та розподіленою, розробленою на основі Google's Bigtable. HBase часто використовується для великих обсягів даних, особливо у випадках, коли необхідний випадковий доступ до даних.

Для підключення до NoSQL баз даних, розробники часто використовують спеціалізовані драйвери або бібліотеки, які відповідають специфічним вимогам окремих NoSQL систем [8].

Наприклад, для MongoDB використовується Mongoose у JavaScript, який надає зручний інтерфейс для взаємодії з базою даних. Такі бібліотеки не тільки спрощують процес з'єднання, але й дозволяють розробникам більш ефективно управляти даними, використовуючи рідний для їхньої мови програмування синтаксис. Робота з NoSQL базами даних також часто вимагає розуміння їх

специфіки, оскільки вони можуть значно відрізнятися від традиційних реляційних систем. Наприклад, MongoDB використовує документо-орієнтований підхід, Cassandra пропонує колоно-орієнтоване зберігання, а Redis відомий як база даних типу "ключ-значення". Кожна з цих систем має свої переваги та особливості, які визначають їх найкращі сценарії використання. Важливим елементом при розробці веб-додатків, які взаємодіють з базами даних, є забезпечення безпеки. Це включає в себе безпечне зберігання конфіденційної інформації, захист від SQL-ін'єкцій (у випадку з реляційними базами даних) та інших видів атак.

Розробники повинні також враховувати аспекти шифрування даних та використання безпечних методів аутентифікації та авторизації.

Вибір між різними типами баз даних та їх інтеграція в архітектуру веб-додатку має вирішальне значення для досягнення високої продуктивності, масштабованості та безпеки.

2 ОПИС АЛГОРИТМІВ ПОШУКУ У МБД

2.1 Опис алгоритмів пошуку у NoSQL базах

Алгоритми пошуку в NoSQL базах даних відрізняються від традиційних реляційних баз даних через їхню унікальну структуру та способи зберігання даних. Оскільки NoSQL бази даних часто використовують неструктуровані або напів-структуровані дані, алгоритми пошуку мають бути гнучкими та ефективними, щоб впоратися з різноманітністю форматів даних.

Основні особливості пошуку у NoSQL базах даних є індексація: як і в реляційних БД, індексація грає ключову роль у пошуку NoSQL.

Індекс працює подібно до індексу в книзі. Він створюється для одного або декількох полів у таблиці, що значно скорочує час пошуку, оскільки база даних може швидко знаходити місце розташування даних, на які є посилання в індексі, замість того, щоб переглядати кожен рядок таблиці.

Таким чином, індексація є важливою для оптимізації запитів, особливо в великих базах даних. Однак індексація має свої недоліки. Створення та підтримка індексів вимагає додаткового місця для зберігання, а також може сповільнити процеси вставки, оновлення та видалення даних у таблицях, оскільки індекси також потребують оновлення під час цих операцій. Тому важливо врівноважувати потребу в швидкому доступі до даних (через індексацію) з потребою в ефективному управлінні ресурсами. Ефективне використання індексації може значно підвищити продуктивність системи баз даних, полегшуючи доступ до даних та забезпечуючи більш швидкі відповіді на запити.

Однак, через різноманітність тих самих даних - механізми індексації можуть бути складнішими та багаторівневими.

– ключ-значення пошук: у базах даних типу ключ-значення, пошук часто відбувається за ключем. Це швидкий і простий метод, але він обмежений, оскільки не дозволяє складних запитів;

- документно-орієнтований пошук: в цьому типі NoSQL БД, які зберігають дані у форматі схожому на JSON або XML, можуть використовуватися більш складні запити, включаючи пошук за вмістом поля в документі;
- графові запити: у графових базах даних пошук виконується за допомогою проходження по вузлах і ребрах графа. Це дозволяє виконувати складні аналітичні запити та знаходити зв'язки між об'єктами;
- повнотекстовий пошук: деякі NoSQL системи підтримують повнотекстовий пошук, дозволяючи виконувати запити на основі текстового вмісту. Це особливо корисно для пошуку великих обсягів неструктурованого тексту;
- масштабованість: однією з ключових переваг NoSQL є горизонтальна масштабованість, що дозволяє ефективно обробляти великі обсяги даних і складні запити.

Реалізація пошуку в NoSQL базах даних залежить від конкретної системи та її архітектури. Часто використовуються спеціалізовані інструменти та мови запитів, що відрізняються від стандартного SQL, адаптовані до особливостей зберігання та структури даних.

Вибір використання технологій NoSQL може допомогти організаціям отримати конкурентну перевагу на своєму ринку, що робить їх більш гнучкими та краще підготовленими для адаптації до змін умови ведення бізнесу [8].

Графові запити до мережевих баз даних у NoSQL є унікальним і важливим інструментом для обробки даних, особливо коли мова йде про складні взаємозв'язки і залежності між об'єктами[9]. Цей тип запитів використовується в графових базах даних, які спеціалізуються на зберіганні та обробці структур даних у вигляді графів.

Основні характеристики графових запитів:

- структура графа: в основі графових баз даних лежить структура графа, що складається з вузлів (відповідають об'єктам або сутностям) та ребер (відповідають взаємозв'язкам між об'єктами);

- гнучкість запитів: графові запити дозволяють виконувати складний пошук залежностей, шляхів, шаблонів та зв'язків, які можуть бути складно представлені в традиційних реляційних базах даних;
- висока ефективність: для сценаріїв, де потрібно аналізувати складні мережі взаємозв'язків, наприклад, соціальні мережі, системи рекомендацій або мережі зв'язків, графові бази даних та запити можуть надавати значну вигоду з точки зору ефективності та швидкості обробки.

Нижче наведено мови запитів для графових баз даних.

Cypher: мова запитів для графових баз даних, зокрема Neo4j. Вона дозволяє формулювати запити, використовуючи декларативний синтаксис для визначення шаблонів у графі.

Основною особливістю Cypher є її декларативний характер, який дозволяє користувачам описувати, що вони хочуть зробити з даними, без необхідності вказувати, як саме це потрібно виконувати. Це робить мову інтуїтивно зрозумілою і легкою в навчанні, особливо для тих, хто вже знайомий з SQL [10].

Cypher використовує вирази, які дозволяють описувати шаблони в графі, на основі яких виконуються запити. Вона включає в себе такі концепції, як вузли, відносини і властивості. Cypher також дозволяє використовувати різноманітні функції для агрегації, фільтрації та трансформації даних.

Важливим аспектом Cypher є її підтримка складних запитів, які включають знаходження шляхів, взаємозв'язків між різними вузлами та аналіз взаємозалежностей в графах. Це робить мову особливо потужною для аналізу мереж, рекомендаційних систем, обробки соціальних мереж та інших областей, де важливі взаємозв'язки та відносини між об'єктами. Наразі, Cypher є ключовим інструментом для роботи з графовими базами даних, надаючи широкі можливості для запитів і аналізу складних даних, при цьому будучи досить зручною та доступною для навчання.

Gremlin: ще одна мова запитів для графових баз даних, яка використовується у Apache TinkerPop[11] та з багатьма популярними графовими базами даних, такими як Neo4j[12], Amazon Neptune. Gremlin підходить для

програмного конструювання запитів, дозволяючи більш гнучко взаємодіяти з графом .

Особливості Gremlin:

– гнучкість: Gremlin підтримує різноманітні операції над графами, включаючи пошук шляхів, вибірку підграфів, агрегацію та багато іншого.
– імперативний стиль: в Gremlin запити формуються як послідовність команд, які модифікують або переглядають стан графу. Це дозволяє користувачам точно керувати процесом обходу графа.
– масштабованість: Gremlin ефективно працює з великими графами, забезпечуючи високу продуктивність операцій, навіть на великих об'ємах даних. Інтеграція з різними графовими базами даних: як вже зазначалось вище – Gremlin сумісний з багатьма популярними графовими базами даних.

3 СУПРОВІД ТА ПІДТРИМКА МБД

3.1 Методи супроводу та підтримки МБД

Супровід та підтримка мережевих баз даних (МБД) та NoSQL баз даних мають свої специфічні аспекти, які важливі для забезпечення стабільності, продуктивності та безпеки цих систем.

Основні аспекти супроводу та підтримки мережевих БД включають наступні елементи.

Моніторинг та оптимізація продуктивності: важливо регулярно моніторити продуктивність мережевих БД. Це включає в себе відстеження часу відгуку, швидкості обробки запитів та ефективності використання ресурсів. Оптимізація може включати тюнінг конфігурації сервера, індексацію даних та використання кешування для підвищення швидкості доступу до даних [13].

Резервне копіювання та відновлення: створення регулярних резервних копій є необхідним для захисту даних від втрати або пошкодження. Важливо також мати надійний план відновлення даних у разі збоїв або катастрофічних подій.

Оновлення та патчі: регулярне оновлення програмного забезпечення мережевих БД є ключовим для запобігання вразливостей безпеки та виправлення помилок. Встановлення патчів та оновлень допомагає підтримувати систему в актуальному стані.

Безпека даних: забезпечення безпеки даних включає в себе контроль доступу, шифрування даних та захист від несанкціонованого доступу.

Технічна підтримка та консультації: надання технічної підтримки користувачам та розробникам, які працюють з мережевими БД, є важливим для вирішення проблем та оптимізації використання системи.

Документація та навчання: підготовка та оновлення документації, а також навчання персоналу, що працює з мережевими БД, є важливими для забезпечення ефективного та правильного використання системи[14].

Управління конфігурацією: правильне управління конфігурацією мережевих БД включає налаштування параметрів зберігання даних, мережевих налаштувань та параметрів безпеки. Наведені Підсхеми мають примітивне

уявлення про структуру БД і це дозволяє програмній системі відправляти добре сформований запит далі, саме ці підсхеми потрібно регулярно оновлювати для коректної роботи застосунку (див. рис. 3.1).

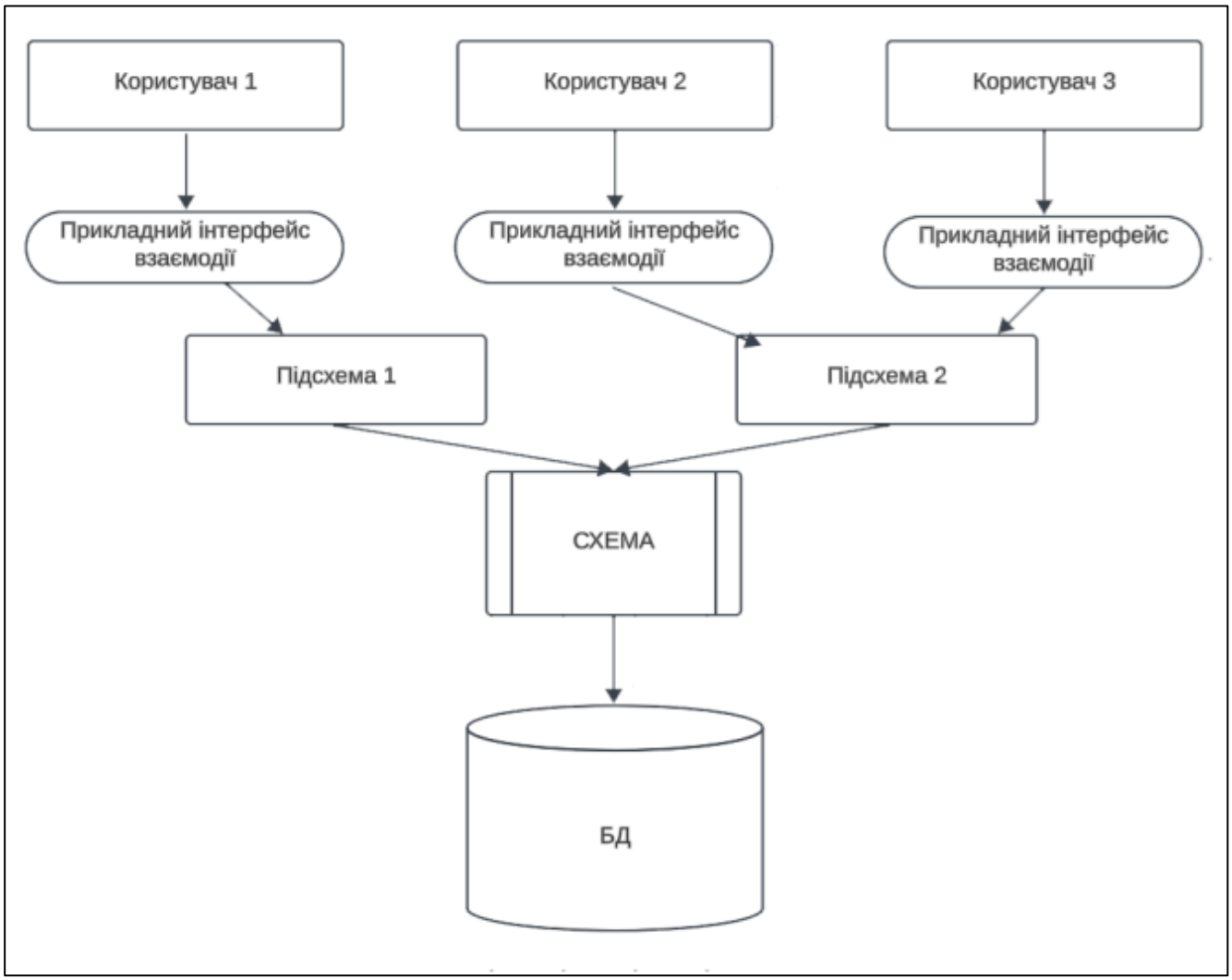

Рисунок 3.1 – Загальна архітектура мережевої СУБД (рисунок виконаний самостійно)

Супровід та підтримка мережевих баз даних вимагають комплексного підходу, що включає різні аспекти, від технічної підтримки до управління конфігурацією та безпекою.

Одним з найважніших аспектів є управління конфігурацією, яке передбачає налаштування системних параметрів, щоб оптимізувати продуктивність та забезпечити стабільність роботи бази даних. Це може включати настройку мережевих параметрів, параметрів сховища даних та налаштувань безпеки[15].

Безпека даних в мережевих БД також є пріоритетом. Вона охоплює заходи для запобігання несанкціонованому доступу, шифрування даних та захист від зовнішніх загроз.[16] Крім того, важливо регулярно проводити аудит системи на предмет вразливостей. Технічна підтримка користувачів та розробників, які використовують мережеві БД, включає допомогу у вирішенні проблем, що виникають під час експлуатації, консультації щодо оптимального використання ресурсів та навчання персоналу для ефективної роботи з системою[17].

Та аспект який здається легким, але якщо його не буде зроблено не за стандартами або не зрозуміло – ефективність систем буде майже нульова, аспект надання документації та проведення навчань дуже важливе для забезпечення того що всі користувачі розуміють як правильно використовувати і обслуговувати саме вашу мережеву базу даних. Це допомагає мінімізувати помилки, спричинені людським фактором, та підвищує загальну ефективність системи. В цілому, ефективний супровід та підтримка мережевих баз даних забезпечує стабільну, безпечну та продуктивну роботу, що є ключовим для будь-якої сучасної інформаційної системи.

4 ОПИС ПРИЙНЯТИХ ПРОЕКТНИХ РІШЕНЬ

4.1 Опис модулів проекту

При розробці програми було прийнято кілька ключових рішень для забезпечення функціональності, зручності та ефективності застосунку. Програма була розроблена з використанням мови Python і таких модулів як requests, tkinter та OS. Вибір цих модулів був обґрунтований їх можливостями та відповідністю вимогам проекту.

Мова програмування: вибір Python був обумовлений його простотою, широким спектром бібліотек і модулів, а також великою спільнотою розробників, що забезпечує підтримку і швидке вирішення проблем.

Requests: обраний для роботи з HTTP-запитами, оскільки забезпечує простий і зручний спосіб взаємодії з веб-ресурсами, підтримуючи всі методи HTTP та пропонуючи зручний інтерфейс для відправки запитів і обробки відповідей.

Tkinter: використовується для створення графічного інтерфейсу користувача (GUI). Це стандартна бібліотека Python для створення GUI-додатків, яка забезпечує простоту у використанні та потужні можливості для побудови інтерфейсів.

OS: застосовується для взаємодії з операційною системою, включаючи роботу з файловою системою, що дозволяє програмі виконувати такі дії, як зчитування та запис файлів, а також отримання інформації про файлову структуру.

Розробка в PyCharm: інтегроване середовище розробки (IDE): Вибір PyCharm обумовлений його потужними інструментами для розробки Python-програм, такими як інтегрований налагоджувач, підтримка тестування, автозавершення коду та аналіз якості коду.

Сама архітектура проекту була розбита на 3 окремих модуля:

- модуль візуальної частини застосунка;
- модуль NoSQL бази даних;

– модуль збереження та зчитування даних у файли на носії.

Вигляд створених модулів у середовищі розробки PyCharm з розподіленням на “Base” модуль, NoSQLDB модуль та SaveModule (див. рис. 4.1).

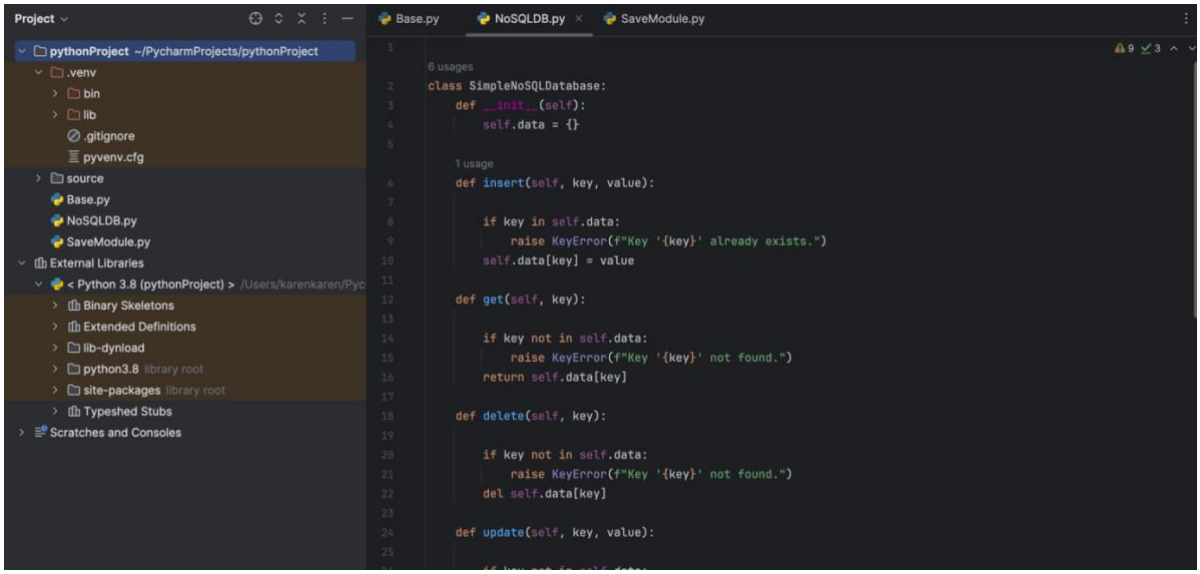

Рисунок 4.1 – Вигляд створених модулів у середовищі розробки PyCharm (рисунок виконаний самостійно)

Модель роботи застосунку з даними була взята зі схожого застосунку який використовувався у експерименті з BigData базами українських бізнесів та їх перекладу та збору статистиці на базі ефективності обробки полей БД [12] (див. рис. 4.2).

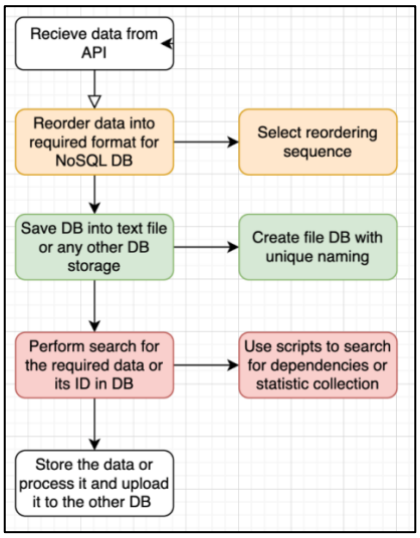

Рисунок 4.2 – UML діаграма структури застосунку (рисунок виконаний самостійно)

Ідея такого переформатування даних полягає у спрощеному індексуванні даних та пошуку неявних або явних залежностей у масивах даних для проведення аналізу, збору статистики або тренуванні моделі штучного інтелекту без підвищених навантажень на штучний інтелект.

5 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

У цьому розділі буде розглянуто створення програмного застосунку на мові Python у Інтегрованому середовищі розробки – PyCharm. Застосунок налаштований на отримання текстового набора даних методом відправки API запиту на зазначений користувачем URL та зазначеними Headers та Query параметрами самої API.

5.1 Створення основного модулю застосунку

Для того, щоб виводити візуальний інтерфейс користувачу – було обрано бібліотеку Tkinter та встановлено її на систему. Під час розробки була використана PyCharm 2024.

Головний інтерфейс програми має 2 кнопки Import Data яка активна та Reorder Data яка стає активною після імпорту даних. (див. рис. 5.1).

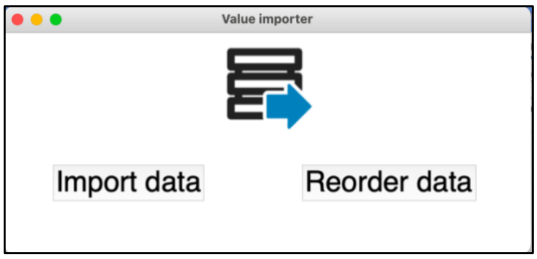

Рисунок 5.1 – Головний інтерфейс застосунку (рисунок виконано самостійно)

Import Data – демонструє інтерфейс імпорту текстових даних з API з урахуванням полей URL, Headers та Query де поле URL є обов’язковим а поля Headers та Query опціональні (див. рис. 5.2).

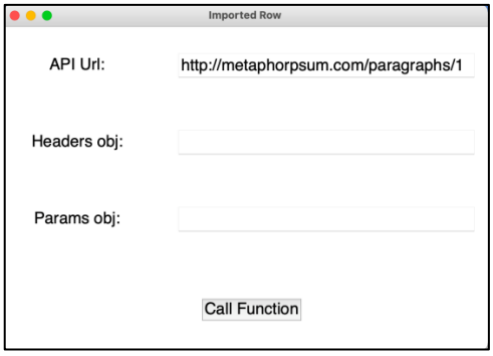

Рисунок 5.2 – Інтерфейс імпорту даних (рисунок виконано самостійно)

Після успішного імпорту даних – користувачу буде показані імпортовані дані та у головному інтерфейсі стане доступна кнопка Reorder Data. Сама програмна реалізація інтерфейсу кнопок та полей вводу реалізована через Tkinter у коді застосунку (див. рис. 5.3).

```
65      label_var2 = tk.Label(new_window, text="Headers obj:", font=custom_font)
66      label_var2.grid(row=1, column=0, padx=padxx, pady=padyy)
67      entry_var2 = tk.Entry(new_window, font=custom_font, width=30)
68      entry_var2.grid(row=1, column=1, padx=padxx, pady=padyy)
69
70      label_var3 = tk.Label(new_window, text="Params obj:", font=custom_font)
71      label_var3.grid(row=2, column=0, padx=padxx, pady=padyy)
72      entry_var3 = tk.Entry(new_window, font=custom_font, width=30)
73      entry_var3.grid(row=2, column=1, padx=padxx, pady=padyy)
```

Рисунок 5.3 – Створення інтерфейсу через бібліотеку Tkinter (рисунок виконано самостійно)

Функція call_api відповідає за відправку API запиту по раніше зазначеному URL з урахуванням параметрів Headers та Query які задав користувач (див. рис. 5.4).

```
def call_api(var1, var2, var3):
    global drecieved
    url = var1
    db2 = NoSQLDB.SimpleNoSQLDatabase()
    querystring = var2
    headers = var3
    response = requests.get(url, headers=headers, params=querystring)
    recieved = response.text
    drecieved = response.text
    db2.load_text(drecieved)
    result = f"Your imported row is: {recieved}"
    messagebox.showinfo( title: "Result", result, parent=new_window)
```

Рисунок 5.4 – Функція call_api (рисунок виконано самостійно)

Інтерфейс для пошуку та переведення отриманих даних через API у нашу NoSQL базу даних (див. рис. 5.5).

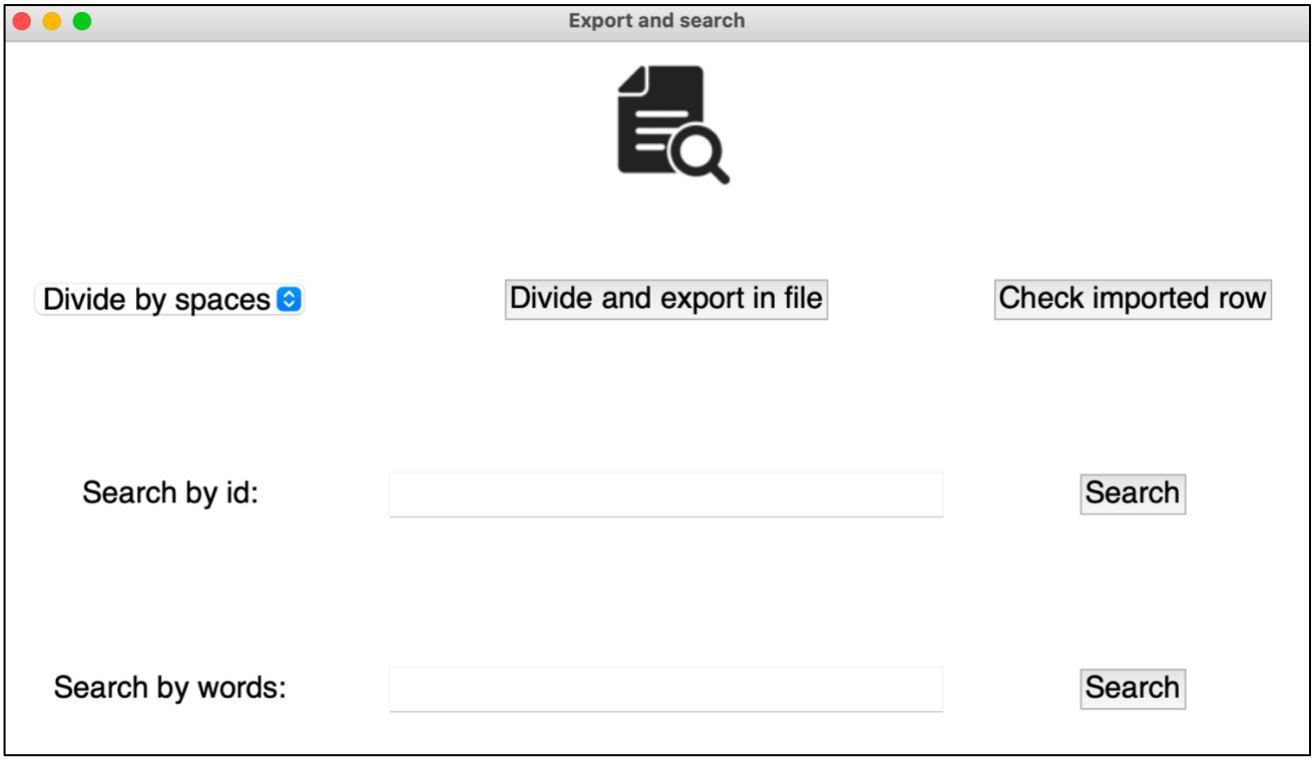

Рисунок 5.5 – Інтерфейс збереження та пошуку даних (рисунок виконано самостійно)

Після того як ми імпортували дані з API – можна їх зберегти у текстовому файлі в форматі NoSQL бази даних у якої кожне слово має порядкове значення котре можна використати як аналог покажчика з графових нереляційних баз даних для знаходження наступних або попередніх даних у базі, також реалізована функція пошуку самих покажчиків за словом до якого вони прив’язані для збору статистики або для аналізу даних.

Модуль NoSQL бази даних з реалізованими методами імпортування тексту та пошуку за показчиком або окремими словами, частинами слів:

```
class SimpleNoSQLDatabase:
    def __init__(self):
        self.data = {}

    def insert(self, key, value):

        if key in self.data:
```

```
            raise KeyError(f"Key '{key}' already exists.")
        self.data[key] = value

    def get(self, key):

        if key not in self.data:
            raise KeyError(f"Key '{key}' not found.")
        return self.data[key]

    def delete(self, key):

        if key not in self.data:
            raise KeyError(f"Key '{key}' not found.")
        del self.data[key]

    def update(self, key, value):

        if key not in self.data:
            raise KeyError(f"Key '{key}' not found.")
        self.data[key] = value

    def search(self, keyword):

        results = {}
        i=0
        error = "There's no such word"
        for key, value in self.data.items():
            if keyword in key or (isinstance(value, str) and keyword
in value):
                i+=1
                results[key] = value
        if (i == 0):
            return error
        else:
            return results
    def searchid(self,keyword):
        results = {}
        error = "There's no such  ID"
        i = 0
        for key, value in self.data.items():
            if keyword in key or (isinstance(value, str) and keyword
```

Продовження коду:

```
in value):
            results[key] = value
            i+=1
            break
   if (i == 0) :
       return error
   else:
       return results


def __str__(self):
```

```
        return str(self.data)

    def load_text(self, text):

        words = text.split()
        for index, word in enumerate(words):
            self.insert(str(index), word)
```

Після виконаного розбиття тексту у NoSQL базу даних – база зберігається у папці проекту з унікальним ідентифікатором назви який створюється з назви рядка та системного часу, приклад (NoSQL words_20240602_181558.txt).

Модуль збереження створеної БД з використанням бібліотеки OS:

```
import os
from datetime import datetime

global filenameIN
filenameIN = ""
def create_and_write_file(base_filename, self):

    global filenameIN
    now = datetime.now()
    timestamp = now.strftime("%Y%m%d_%H%M%S")

    filename = f"{base_filename}_{timestamp}.txt"

    home_dir = os.path.expanduser("~/PycharmProjects/pythonProject")

    file_path = os.path.join(home_dir, filename)

    with open(file_path, 'w') as file:
```

Продовження коду:

```
        for key, value in self.data.items():
            file.write(f"{key}: {value}\n")

    print(f"File '{filename}' has been created in your home directory
and text has been written to it.")
    filenameIN = filename
```

Приклад структури збереженої БД після отриманого параграфу з API (див. рис. 5.6).

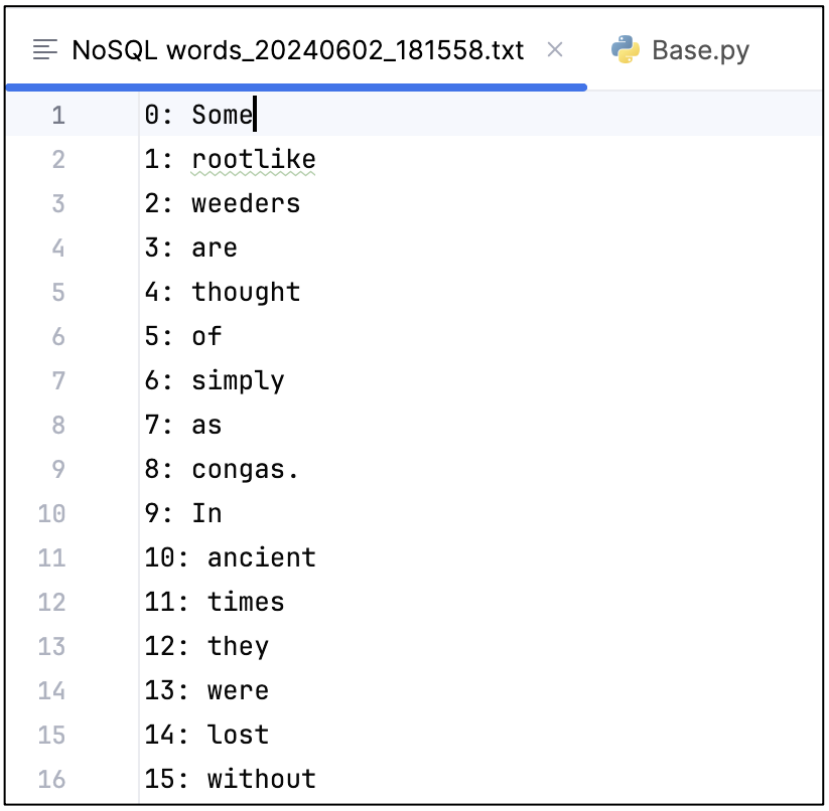

Рисунок 5.6 – Структура збереженої NoSQL БД (рисунок виконано самостійно)

Виконання пошуку по БД може виконуватись за допомогою методів search та searchid які були зазначені вище у описі методів БД, сам візуальний інтерфейс демонструє результати пошуку у вигляді поп-апів над інтерфейсом (Export and search).

ВИСНОВКИ

Після дослідження методів створення та супроводу мережевих баз даних були з’ясовані задачі дослідження, проведен аналіз проблем завдяки якому ці задачі були віришені.

У ході досліджень та вивчення наявних стандартів та протоколів для мережевих баз даних, вивчення особливостей роботи з великими обсягами даних та потребах у зміні структур коммунікації з МБД користувачем - з’ясовано що для удосконалення створення та супроводу мережевих баз даних необхідно розробляти Актуалізовану СУБД яка буде включати у себе усі переваги мережевих баз даних та мінімізує складності опанування та налаштування. Також були наведені та запропоновані усі можливі технології для реалізації такої системи управління базою даних.

У ході створення програмного застосунку з NoSQL базою даних графового типу для текстових даних – були виявлені переваги використання NoSQL баз даних у задачах обробки даних, збору статистики та аналізу поєднання даних будь-яких типів. Також була продемонстрована простота створення баз даних такого типу з використанням поширених технологій.

Переваги такої СУБД будуть такими:

- масштабованість: NoSQL бази даних, як правило, краще масштабуються ніж традиційні реляційні бази даних. Вони можуть ефективно обробляти величезні обсяги даних та високі навантаження на читання/запис, що особливо важливо для великих мережевих систем;
- гнучкість схеми: МБД часто не вимагають жорсткої схеми, що дозволяє більш гнучко працювати з різними типами даних. Це особливо корисно в динамічних середовищах, де структура даних може часто змінюватися;
- висока доступність та відмовостійкість: багато МБД пропонують вбудовані рішення для реплікації даних та відновлення після збоїв, що забезпечує високу доступність та відмовостійкість даних;

- оптимізація для специфічних випадків використання: різні типи NoSQL баз даних (документ-орієнтовані, графові, ключ-значення) оптимізовані для конкретних випадків використання, що дозволяє вибрати найбільш підходящу систему для конкретних задач;
- простота горизонтального сасштабування: МБД майже усі розроблені з урахуванням горизонтального масштабування, що дозволяє додавати більше серверів для обробки збільшеного обсягу даних;
- ефективність для неструктурованих та напівструктурованих даних: МБД ефективно працюють з неструктурованими та напівструктурованими даними, що є важливим у сучасних додатках, де часто використовуються дані такого типу;
- швидкість обробки запитів: для деяких типів запитів МБД можуть пропонувати кращу продуктивність порівняно з традиційними реляційними системами.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Edgar H. Sibley. The CODASYL Data Base Approach: A COBOL Example of Design and Use of a Personnel File. 1974.

2. Santiago Timón-Reina, Mariano Rincón, Rafael Martínez-Tomás. An overview of databases and their applications in the biomedical domain. 2019 vol. 5-10.

3. Adam Fowler. NoSQL For Dummies / Publisher: For Dummies, 2015 – 464 p.

4. Ian Robinson, Jim Webber, Emil Eifrem. Graph Databases: New Opportunities for Connected Data / Publisher: O'Reilly Media, 2015 - 236 p.

5. How LIquid Connects Everything So Our Members Can Do Anything / URL: https://www.linkedin.com/blog/engineering/graph-systems/how-liquid-connects-everything-so-our-members-can-do-anything (дата звернення: 29.04.2024).

6. Chown, P. 2002. Advanced Encryption Standard (AES) cipher suites for Transport Layer Security (TLS). RFC 3268. June.

7. Pramod Sadalage, Martin Fowler. NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence / Publisher: Addison-Wesley Professional, 2012 – 192 p.

8. Dan McCreary, Ann Kelly. Making Sense of NoSQL: A guide for managers and the rest of us / Publisher: Manning, 2013 – 312 p.

9. What Is NoSQL? NoSQL Databases Explained / URL: https://www.mongodb.com/resources/basics/databases/nosql-explained (дата звернення: 01.05.2024).

10. Igor Shubin, Andrii Kozyriev, Method for Solving Quantifier Linear Equations for Formation of Optimal Queries to Databases in: Computational Linguistics and Intelligent Systems 2023, Proceedings of the 7th International Conference on Computational Linguistics and Intelligent Systems. vol. 449-459

11. Neo4j // "Cypher Introduction" - 2019. // https://neo4j.com/docs/cypher-manual/current/introduction/.

12. Apache TinkerPop// Gremlin Query Language// https://tinkerpop.apache.org/gremlin.html - 2015.

13. Victoria Vysotska, Ihor Shubin, Maksym Mezentsev, Karen Kobernyk, Grygoryy Chetverikov, Ukrainian Big Data: The Problem of Databases Localization in: Intelligent Systems Workshop on 8th International Conference on Computational Linguistics and Intelligent Systems (CoLInS-2024), vol. 122-133.

14. Eric Redmond, Jim R. Wilson. "Seven Databases in Seven Weeks: A Guide to Modern Databases and the NoSQL Movement." Publisher: Pragmatic Bookshelf, 2012 – vol. 144-150.

15. Shashank Tiwari. "Professional NoSQL." Publisher: Wrox, 2011 – vol. 80-83.

16. Hassan A. Afyouni. "Database Security and Auditing: Protecting Data Integrity and Accessibility." Publisher: Cengage Learning, 2005 – vol. 30-37.

17. David Litchfield. "The Database Hacker's Handbook: Defending Database Servers." Publisher: Wiley, 2005 – vol. 201-211.