

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ
СІКОРСЬКОГО»**

Навчально-науковий інститут атомної та теплової енергетики
Кафедра цифрових технологій в енергетиці

"На правах рукопису"
УДК 004.65

«До захисту допущено»

Завідувач кафедри

Наталія АУШЕВА

“ ” _____ 2022р.

Магістерська дисертація

зі спеціальності - 122 Комп'ютерні науки
за освітньо-професійною програмою магістерської підготовки -
Комп'ютерний моніторинг та геометричне моделювання процесів і систем

на тему Оптимізація запитів до великих обсягів даних еколого-економіко-
енергетичного моніторингу

Виконав: студент 2 курсу, групи ТР-11мп

Костриця Сергій Володимирович

(прізвище, ім'я, по батькові)

(підпис)

Науковий керівник доцент, к.т.н, Полягушко Л. Г.

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

(підпис)

Рецензент _____

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

(підпис)

Засвідчую, що у цій магістерській
дисертації немає запозичень з праць
інших авторів без відповідних посилань.

Студент _____
(підпис)

“Київський політехнічний інститут ім. Ігоря Сікорського”

Спеціальності 122 Комп'ютерні науки

« _____ » 2022p.

НА МАГІСТЕРСЬКУ ДИСЕРТАЦІЮ СТУДЕНТУ

5. Орієнтований перелік ілюстративного матеріалу __Актуальність теми дослідження, об'єкт, мета та завдання, об'єкт та предмет дослідження, аналіз аналогічних систем, методи дослідження, загальна концепція, структура модулів, алгоритм, схема інформаційних потоків, USE CASE, клієнтська частина, серверна частина, база даних, початок роботи з системою, експериментальне дослідження результатів, практичне значення, висновки

6. Орієнтований перелік публікацій Полягушко Л.Г, Левицький І.В., Костриця С.В.
Еталонне тестування та аналіз реляційних баз даних моніторингових систем з великими
обсягами даних. III Міжнародна студентська конференція “Розвиток суспільства та
науки в умовах цифрової трансформації”, Луцьк, Україна, 16 грудня 2022

7. Дата видачі завдання « 2 » липня 2022р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1.	Вирішення організаційних питань початку практики: оформлення документації, встановлення каналів комунікації з керівником дипломної роботи	01.09.22р.- 07.09.22р.	
2.	Підготовка матеріалів, що були розроблені за час навчання	01.09.22р.- 14.09.22р.	
3.	Побудова моделей, розробка методів та алгоритмів програмного забезпечення	14.09.22р.- 11.10.22р.	
4.	Демонстрація програмного продукту керівнику. Виправлення помилок	12.10.22р.- 20.10.22р.	
5.	Захист програмного продукту комісії. Виправлення помилок.	21.10.22р.- 25.10.22р.	
6.	Залік	26.10.22р.	
7.	Розроблення стартап-проєкту	27.10.22р.- 15.11.22р.	
8.	Підготовка магістерської дисертації до передзахисту та передача її керівнику. Постановка доповіді	16.11.22р.- 20.11.22р.	
9.	Передзахист магістерської дисертації комісії. Перевірка нормоконтролю	21.11.22р.- 30.11.22р.	
10.	Подання документів на кафедру	05.12.22р.	
11.	Захист	12.12.22р.- 18.12.22р.	

Студент

(підпис)

Костриця С. В.

(прізвище та ініціали)

Науковий керівник

(підпис)

Полягушко Л. Г.

(прізвище та ініціали)

РЕФЕРАТ

Магістерська дисертація за темою “Оптимізація запитів до великих обсягів даних еколого-економіко-енергетичного моніторингу” виконана студентом кафедри цифрових технологій в енергетиці НН ІАТЕ Кострицею Сергієм Володимировичем зі спеціальності 122 “Комп’ютерні науки” за освітньо-професійною програмою “Комп’ютерний моніторинг та геометричне моделювання процесів та систем” та складається зі: вступу; 5 розділів: постановки задачі оптимізації запитів до великих обсягів даних еколого-економіко-енергетичного моніторингу, огляду практичних методів вирішення поставленого завдання, опису програмної реалізації веб-орієнтованої системи комплексного еколого-економіко-енергетичного моніторингу, опису роботи програмного забезпечення та стартапу; висновків до кожного з цих розділів; загальних висновків; списку використаних джерел, який налічує 51 джерело та 1 додаток. Загальний обсяг роботи 104 сторінки.

Актуальність теми. Інформація є найважливішим ресурсом в сучасному світі. Для зберігання інформації в цифровому вигляді використовуються бази даних, обсяг яких може сягати дуже великих розмірів, що в свою чергу впливає на час пошуку необхідних даних та швидкість обробки запитів сервером баз даних. Сучасні системи керування базами даних мають безліч різних механізмів для оптимізації роботи серверу, але їх правильному налаштуванню не приділяють необхідної уваги. Тому актуальним є створення набору рекомендацій з оптимізації бази даних, підтверджений експериментально та застосований до системи комплексного еколого-економіко-енергетичного моніторингу.

Метою роботи є формування набору рекомендацій з оптимізації бази даних системи комплексного еколого-економіко-енергетичного моніторингу (КЕЕЕМ) та його застосування до веб-орієнтованої системи КЕЕЕМ для пришвидшення роботи системи та обробки інформації з бази даних.

Завдання дослідження:

1. Проаналізувати існуючі техніки оптимізації систем реляційних баз даних.

2. Проаналізувати веб-орієнтовану систему KEEEM та її базу даних на доцільність використання методів оптимізації.

3. Удосконалити структуру, пошукові запити та налаштування параметрів серверу бази даних.

4. Сформувати набір рекомендацій по оптимізації бази даних та продемонструвати його роботу на прикладі веб-орієнтованої системи KEEEM.

Об'єкт дослідження. Комп'ютерні інформаційні системи і технології зберігання великих обсягів моніторингової інформації у реляційних базах даних.

Предмет дослідження. Моделі та методи оптимізації роботи реляційних баз даних великих обсягів.

Методи дослідження. Еталонне тестування, синтаксичний метод оптимізації, вартісний метод оцінки плану виконання запиту.

Практичне значення одержаних результатів. Розроблений та застосований до системи комплексного еколого-економіко-енергетичного моніторингу набір рекомендацій дасть змогу користувачам системи отримати швидкий доступ до необхідних даних, що пришвидшить вирішення проблем, які безпосередньо впливають на здоров'я населення.

Апробація результатів дисертації. Результати дисертації було представлено на III Міжнародній студентській конференції "Розвиток суспільства та науки в умовах цифрової трансформації"(2022) [1].

Ключові слова. Реляційна база даних, методи оптимізації, еталонне тестування, веб-орієнтована система, комплексний моніторинг.

ABSTRACT

The master's dissertation on the topic "Optimization of queries to large volumes of environmental, economic and energy monitoring data" was completed by a student of the Department of Digital Technologies in Energy of NN IATE Kostrytsia Serhii Volodymyrovych majoring in 122 "Computer Science" under the educational and professional program "Computer monitoring and geometric modeling of processes and systems" and consists of: introduction; 5 sections: setting the task of optimizing requests for large volumes of environmental-economic-energy monitoring data, review of practical methods for solving the task, description of software implementation of a web-oriented system of complex environmental-economic-energy monitoring, description of software operation and startup; conclusions to each of these sections; general conclusions; of the list of used sources, which includes 51 sources and 1 appendix. The total volume of work is 104 sheets.

Actuality of topic. Information is the most important resource in the modern world. To store information in digital form, databases are used, the volume of which can reach very large sizes, which in turn affects the time of searching for the necessary data and the speed of processing requests by the database server. Modern database management systems have many different mechanisms for optimizing server performance, but their correct configuration is not given the necessary attention. Therefore, it is urgent to create a set of recommendations for optimizing the database, confirmed experimentally and applied to the system of complex environmental, economic and energy monitoring.

The purpose of the work is the formation of a set of recommendations for optimizing the database of the system of integrated ecological, economic and energy monitoring (IEEEM) and its application to the web-oriented system of IEEEM to speed up the operation of the system and processing of information from the database.

Objectives of the study:

1. To analyze the existing optimization techniques of relational database systems.

2. To analyze the web-oriented IEEEEM system and its database for the feasibility of using optimization methods.

3. To improve the structure, search queries and settings of database server parameters.

4. Form a set of recommendations for database optimization and demonstrate its operation using the example of the web-oriented IEEEEM system.

Object of study. Computer information systems and technologies for storing large volumes of monitoring information in relational databases.

Subject of study. Models and methods of optimizing the operation of relational large volume databases.

Research methods. Benchmark testing, syntactic method of optimization, cost method of query execution plan evaluation.

Practical significance of the obtained results. The set of recommendations developed and applied to the system of integrated ecological, economic and energy monitoring will enable users of the system to get quick access to the necessary data, which will speed up the resolution of problems that directly affect on health of the population.

Approbation of the results of the dissertation. The results of the dissertation were presented at the III International Student Conference "Development of Society and Science in the Conditions of Digital Transformation" (2022) [1].

Keywords. Relational database, optimization methods, benchmark testing, web-oriented system, complex monitoring.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧКИ.....	11
ВСТУП.....	12
1 ПОСТАНОВКА ЗАДАЧІ ОПТИМІЗАЦІЇ ЗАПИТІВ ДО ВЕЛИКИХ ОБСЯГІВ ДАНИХ ЕКОЛОГО-ЕКОНОМІКО-ЕНЕРГЕТИЧНОГО МОНІТОРИНГУ	15
1.1 Потенційні користувачі.....	15
1.2 Перелік задач програмного продукту.....	15
1.3 Опис модулів програмного продукту	16
1.3.1 Модуль оптимізації серверу бази даних	17
1.3.2 Модуль оптимізації структури бази даних	17
1.3.3 Модуль оптимізації SQL-запиту.....	18
1.4 Опис модулів системи еколого-економіко-енергетичного моніторингу.....	20
1.5 Огляд існуючих інструментів оптимізації баз даних.....	21
1.5.1 EverSQL Query Optimizer	22
1.5.2 APEXSQL Plan.....	23
1.5.3 Idera DB Optimizer.....	24
1.5.4 DbForge Studio for MySQL	25
1.5.5 AppOptics APM.....	26
1.6 Висновки до розділу	27
2 ОГЛЯД ПРАКТИЧНИХ МЕТОДІВ ВИРІШЕННЯ ПОСТАВЛЕНОГО ЗАВДАННЯ	28
2.1 Обробка запиту до реляційних СКБД	28
2.2 Метод оптимізації, оснований на синтаксисі	30
2.3 Метод оптимізації, оснований на вартості.....	32
2.4 Порядок оптимізації виконання запиту.....	34
2.5 Налаштування застосунків	36
2.6 Висновки до розділу	37

3	ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ ВЕБ-ОРІЄНТОВАНОЇ СИСТЕМИ КОМПЛЕКСНОГО ЕКОЛОГО-ЕКОНОМІКО-ЕНЕРГЕТИЧНОГО МОНІТОРИНГУ	38
3.1	Принцип та порядок роботи системи комплексного еколого-економіко-енергетичного моніторингу	38
3.2	Засоби розробки програмного продукту	39
3.2.1	Мова програмування JavaScript	40
3.2.2	Мова гіпертекстової розмітки HTML	42
3.2.3	Каскадна таблиця стилів CSS	42
3.2.4	Бібліотека React	43
3.2.5	Система контролю версій Git	45
3.2.6	Фреймворк Express	46
3.2.7	Реляційна база даних MySQL	47
3.2.8	Бібліотека React Ag-Grid для роботи з таблицями	48
3.2.9	Бібліотека Read-Excel-File	49
3.3	Опис програми та її алгоритмів	50
3.4	Висновки до розділу	59
4	ОПИС РОБОТИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	60
4.1	Інсталяція програмного забезпечення	60
4.2	Вимоги до обчислювальної техніки	60
4.3	Демонстрація та представлення сценаріїв програмної системи	61
4.3.1	Головна сторінка	61
4.3.2	Карта викидів	63
4.3.3	Довідники	64
4.4	Експериментальні дослідження параметрів функціонування бази даних КЕЕЕМ	66
4.4.1	Запит 1. Відображення маркерів на інтерактивній мапі	67
4.4.2	Запит 2. Отримання детальної інформації про місто	72
4.4.3	Запит 3. Додання нового полігону	74
4.5	Висновки до розділу	75

5 СТАРТАП-ПРОЄКТ	76
5.1 Опис ідеї стартапу	76
5.2 Технологічний аудит ідеї проєкту	78
5.3 Аналіз ринкових можливостей запуску стартап-проєкту	79
5.4 Розроблення ринкової стратегії проєкту	86
5.5 Розроблення маркетингової програми стартап-проєкту	88
5.6 Висновки до розділу	91
ВИСНОВКИ	92
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	94
ДОДАТОК А	99

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧКИ

KEEEM	Комплекс еколого-економіко-енергетичного моніторингу
СКБД	Система керування базами даних
Фреймворк	Інфраструктура програмних рішень, що полегшує розробку складних систем
SQL	(англ. Structured Query Language) – структурована мова запитів
HTML	Мова гіпертекстової розмітки
DOM	(англ. Document Object Model) - об'єктна модель документа
CSS	(англ. Cascading Style Sheet) — каскадна таблиця стилів
SPA	(англ. Single Page Application) — односторінкова web-сторінка, що маніпулює DOM
API	(англ. Application Programming Interface) — програмний інтерфейс додатків
React	Бібліотека для розробки клієнтських інтерфейсів
Express	Бібліотека для розробки серверних застосунків
NodeJS	Платформа для розробки рішень на основі мови програмування JavaScript

ВСТУП

У сучасному світі найважливішим ресурсом є інформація. Завдяки науково-технічному прогресу інформація стала поширюватися у цифровому вигляді, так само змінилися методи її зберігання. Наразі популярним є використання реляційних баз даних для швидкого доступу, обробки та безпечного зберігання цифрової інформації. Розміри баз даних можуть коливатися в межах від кількох кілобайтів до сотень терабайтів, в залежності від об'єму даних, що в них міститься. Зрозуміло, що чим більше база містить в собі даних (записів в таблицях), тим більшим є час пошуку необхідної інформації в них, що в свою чергу впливає на швидкість обробки запитів сервером.

Сучасний інтернет складається з великої низки веб-ресурсів, які використовують бази даних для зберігання та швидкого доступу до різноманітної інформації, без яких неможлива їх повноцінна робота. Дані в реляційних базах даних зберігаються в таблицях, у вигляді записів рядків з певним для кожної таблиці набором колонок. В свою чергу бази даних зберігаються і працюють на веб-сервері. Для доступу та маніпулювання даними в базі використовується структурована мова запитів (SQL).

SQL складається з набору операторів, які поділяються на наступні групи за призначенням:

—

DML – група операторів маніпулювання даними (управління вмістом бази даних та вилучення інформації з неї);

— DDL – оператори, які описують дані для управління об'єктами бази даних;

— TCL – набір операторів управління транзакціями;

— DCL – група операторів визначення доступу до даних.

SQL є стандартизованою мовою, але в ній є багато різновидів, які називаються діалектами.

Для налагодження та маніпулювання базами даних використовуються системи керування базами даних (СКБД). Одні з найбільш популярних реляційних СКБД – MySQL, Oracle Database, PostgreSQL і MS SQL Server. В даній роботі буде розглядатися найпоширеніша із них, а саме СКБД MySQL.

Сервер бази даних щосекунди оброблює тисячі запитів, які надходять до нього внаслідок роботи користувачів веб-ресурсів. Очевидно, що це викликає неабияке навантаження на сервер і, як наслідок, зменшується швидкість відгуку веб-сервісу на дії користувача. Це впливає на ранжування (популярність) веб-сайту у пошукових системах, а що найбільш важливе - від швидкості виконання запитів до систем критичної інфраструктури, до яких в свою чергу належать системи моніторингу за довкіллям, може залежати здоров'я людей [3].

Тому є доцільним розгляд вищезазначених проблем. Одним з шляхів їх вирішення є збільшення потужності комп'ютерів, на яких знаходяться системи керування базами даних. Однак, результат залежить від того, наскільки система ефективно використовує надані їй ресурси, тому найбільшої ефективності можна досягнути завдяки оптимізації структури бази даних, коректних налаштувань серверу та оптимізації SQL-запитів.

Внаслідок правильно виконаної оптимізації можна досягнути наступних результатів:

- зменшення навантаження на сервер;
- збільшення швидкості виконання запитів;
- запобігання блокуванню веб-ресурсу, яке може виникнути внаслідок перевантаження серверу та порушення роботи сусідніх веб-сайтів на загальному хостингу;
- економія ресурсів хостингу;
- мінімізація часу очікування завантаження веб-ресурсу.

Виділимо основні пункти оптимізації, які є найбільш ефективними для БД:

- оптимізація серверу бази даних;
- оптимізація структури бази даних;

— оптимізація SQL-запитів.

При оптимізації бази даних в першу чергу слід звернути увагу на наступні рекомендації:

— потрібно проаналізувати параметри СКБД, оскільки найчастіше параметри серверу, вказані за замовчуванням, розраховані на дуже малі бази даних. Це означає, що вони не передбачають великого навантаження, що також стосується й апаратного забезпечення сервера;

— дуже часто проблеми при роботі з великими обсягами даних виникають при вибірці даних шляхом об'єднання стовпців з різних таблиць. Щоб прискорити цей процес необхідно впевнитися в тому, що всі стовпці проіндексовані. Швидкість операції буде значно вище, якщо з'єднуються стовпці однакових типів;

— необхідно контролювати цілісність посилань. Маються на увазі ті випадки, коли налаштовані каскадні видалення та оновлення. В той час, як користувач намагається щось видалити, СКБД шукає всі зв'язані сутності і намагається зрозуміти, слід їх видаляти чи ні. В такому випадку можна не видаляти, а помічати як видалені. Чи намагатися якимось порвати цілісність посилань, але це може привести до неузгодженості, коли користувач випадково забуде видалити разом з основним записом всі зв'язані. В цьому випадку в базі будуть накопичуватися «загублені» записи. Вони, як мінімум, будуть марно займати місце, а як максимум - можуть привести до неправильної роботи бізнес-логіки. Тож якщо користувач дуже часто видаляє щось з БД, то переривати цілісність посилань потрібно дуже обережно, обов'язково перевіривши, що видалилось дійсно все.

У результаті сформований набір рекомендацій повинен якнайкраще підходити саме для системи комплексного еколого-економіко-енергетичного моніторингу, але внаслідок детального дослідження особливостей роботи реляційних баз даних, його розширену версію можна брати як базис для оптимізації баз даних з великими обсягами даних.

1 ПОСТАНОВКА ЗАДАЧІ ОПТИМІЗАЦІЇ ЗАПИТІВ ДО ВЕЛИКИХ ОБСЯГІВ ДАНИХ ЕКОЛОГО-ЕКОНОМІКО-ЕНЕРГЕТИЧНОГО МОНІТОРИНГУ

Завданням роботи є набір рекомендацій з оптимізації роботи бази даних, підтверджений експериментально та застосований до системи комплексного еколого-економіко-енергетичного моніторингу. Оптимізація дасть змогу користувачам (експертам) системи отримати швидкий доступ до необхідних даних, що пришвидшить вирішення проблем, які безпосередньо впливають на здоров'я населення.

1.1 Потенційні користувачі

Система комплексного еколого-економіко-енергетичного моніторингу є експертною системою, яка спрямована на визначення заходів, що потрібно зробити для різних екологічних випадків, з точки зору законності, коштів та способу проведення. Тому в першу чергу система призначена для експертів системи, яким швидкий доступ до даних є незамінним у їх роботі. Комплекс також надає відкритий доступ до даних моніторингу для неавторизованих користувачів, тому будь-хто може швидко переглянути дані про проведені / заплановані заходи і відслідкувати прогрес вирішення екологічних проблем.

Набір рекомендацій орієнтований на систему KEEEM, але розробники можуть брати його в увагу при оптимізації баз з великими обсягами даних.

1.2 Перелік задач програмного продукту

Перелік задач, які необхідно реалізувати в програмному продукті:

- мінімізації часу очікування завантаження сайту моніторингової системи;
- зменшення навантаження на сервер бази даних моніторингової системи;
- збільшення швидкості виконання запитів.

1.3 Опис модулів програмного продукту

Завдання програмного продукту досягаються внаслідок взаємної роботи модулів оптимізації, в даному випадку система складається з трьох таких модулів, що впливають на швидкість роботи основних компонентів комплексу. Ці компоненти зображені на схемі нижче (рисунок 1.1).

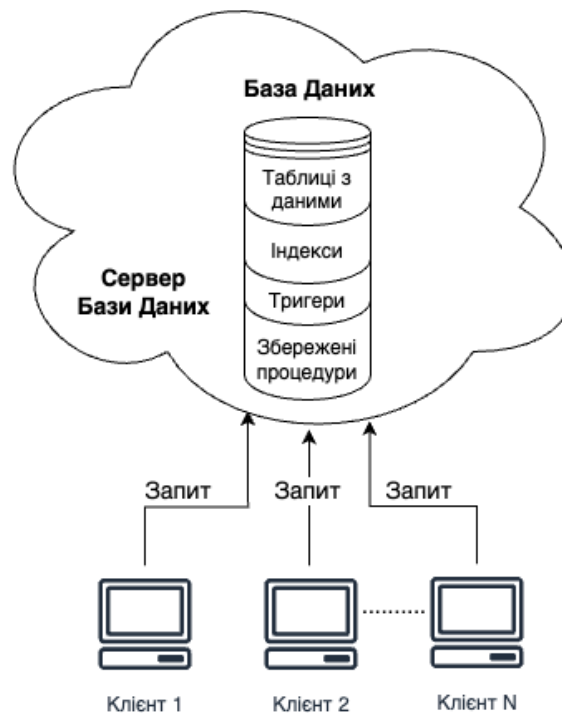


Рисунок 1.1 — Схема цілових компонентів модулів оптимізації

Складовими комплексу є сервер, база даних, що складається з таблиць наповнених даними з певною структурою, індексів, тригерів, збережених процедур та інших компонентів. Клієнти системи виконують запити до бази даних, тому вони також є його складовою. До серверу надходять безліч SQL-запитів, де переважаючими є запити вибірки, тому саме до їх оптимізації необхідно приділити максимум уваги. Потрібно впевнитися, що запити вибірки не містять в собі надлишкової інформації, яка не буде використовуватися клієнтською стороною, бо від кількості трафіку залежить швидкість його передавання. Тож чим більше непотрібної інформації передається, тим повільніше будуть функціонувати модулі комплексу.

1.3.1 Модуль оптимізації серверу бази даних

Даний модуль виконує аналіз параметрів СКБД та надає поради для їх коректного налаштування під потреби конкретної бази даних.

До оптимізації серверу також належить конфігурація кешування запитів, де зберігається план виконання, який відповідає запиту певної семантики. Кеш використовується для зменшення часу відповіді СКБД для часто використовуваних запитів, тому правильна його конфігурація відіграє значну роль в оптимізації.

Збережені процедури є важливими при оптимізації, оскільки код процедури компілюється при її першому виконанні і надалі зберігається на сервері у скомпільованому вигляді, тому перетворення складних SQL-запитів, що часто використовуються у збережені процедури підвищить продуктивність системи.

Вхідна інформація модуля:

- початкова конфігурація параметрів серверу;
- початкова конфігурація кешування запитів на сервері;
- SQL-запити для потенційного перетворення на збережені процедури.

Вихідна інформація модуля:

- оптимізована конфігурація параметрів серверу;
- оптимізована конфігурація кешування запитів на сервері;
- збережені процедури.

1.3.2 Модуль оптимізації структури бази даних

Структура бази даних відіграє важливу роль при масштабуванні системи, тому потрібно правильно її побудувати. Але коли існує вже побудована структура – необхідно виконати її аналіз, знайти проблемні місця та виправити їх. Це завдання виконує модуль оптимізації структури бази даних.

Необхідним кроком для оптимізації структури бази даних є аналіз використовуваних індексів. Вони допомагають виконувати пошук певних значень, або діапазонів значень у великій таблиці, уникаючи таким чином повного сканування таблиці. Тому їх правильне використання дозволить пришвидшити виконання запитів вибірки до бази даних.

В реляційних базах даних кожна таблиця може мати первинний ключ. Первинний ключ - це поле або набір полів, що унікально ідентифікують запис в таблиці. Коректним підходом при побудові структури БД є обов'язкове використання первинних ключів у кожній таблиці.

Зв'язування таблиць в реляційних БД досягається використанням зовнішніх ключів, які являють собою одне або декілька полів у таблиці, що містять посилання на поле або поля первинного ключа іншої таблиці. Зовнішній ключ в реляційних БД задається обмеженням FOREIGN KEY.

Вхідна інформація модуля:

- початковий список індексів таблиць бази даних;
- початковий список зв'язків між таблицями бази даних;
- початковий список ключів таблиць бази даних.

Вихідна інформація модуля:

- оптимізований список індексів таблиць бази даних;
- оптимізований список зв'язків між таблицями бази даних;
- оптимізований список ключів таблиць бази даних.

1.3.3 Модуль оптимізації SQL-запиту

При написанні SQL-запитів потрібно дотримуватись певних правил, які допоможуть максимально ефективно виконати запит. Наприклад одною з найбільш розповсюджених помилок при написанні запитів вибірки є діставання даних рядка зі всіма полями таблиці, за допомогою конструкції `SELECT * FROM [Table]`, де Table – ім'я таблиці по якій виконується вибірка. Коли отримані дані не використовуються в повну міру - це може бути надлишковим, що викликає часові затримки при виконанні запиту.

Також важливим є розуміння того, що арифметика в фільтрах, функції та повнотекстовий пошук нівелюють використання індексів. Таким чином запити не будуть швидкими, користувачеві потрібно бути або спрощувати їх, або використовувати специфічні рішення, такі як плагіни для повнотекстового пошуку чи функціональний індекс.

Блокування у реляційних базах даних допомагає уникнути колізій при зчитуванні рядку таблиці у той час, коли інший клієнт проводить операцію запису до того ж самого рядка, що приводить до некоректного зчитування даних. Модуль оптимізації SQL-запитів виконує аналіз механізмів блокування, застосовуваних в базі даних та визначає при використанні якого механізму блокування можна отримати вигоду у продуктивності.

У СКБД MySQL є такі основні механізми зберігання даних як MyISAM та InnoDB, які відрізняються механізмами блокування, що в них застосовуються. MyISAM використовує табличне блокування, що забезпечує високу швидкість блокування для великих таблиць [13]. Для InnoDB за замовчуванням використовується блокування на рівні рядків. Однак, під час явного блокування використовується блокування таблиць, незалежно від механізму збереження даних. У випадку явного блокування можна отримати вигоду у продуктивності за рахунок одноразового призначення блокування та затримки запису оновлених індексів на час блокування.

Вхідна інформація модуля:

- початкові SQL запити;
- початкові налаштування блокування таблиць.

Вихідна інформація модуля:

- оптимізовані SQL запити;
- оптимізовані налаштування блокування таблиць.

1.4 Опис модулів системи еколого-економіко-енергетичного моніторингу

Структуру взаємодії між різними модулями схематично представлено на рисунку 1.2.

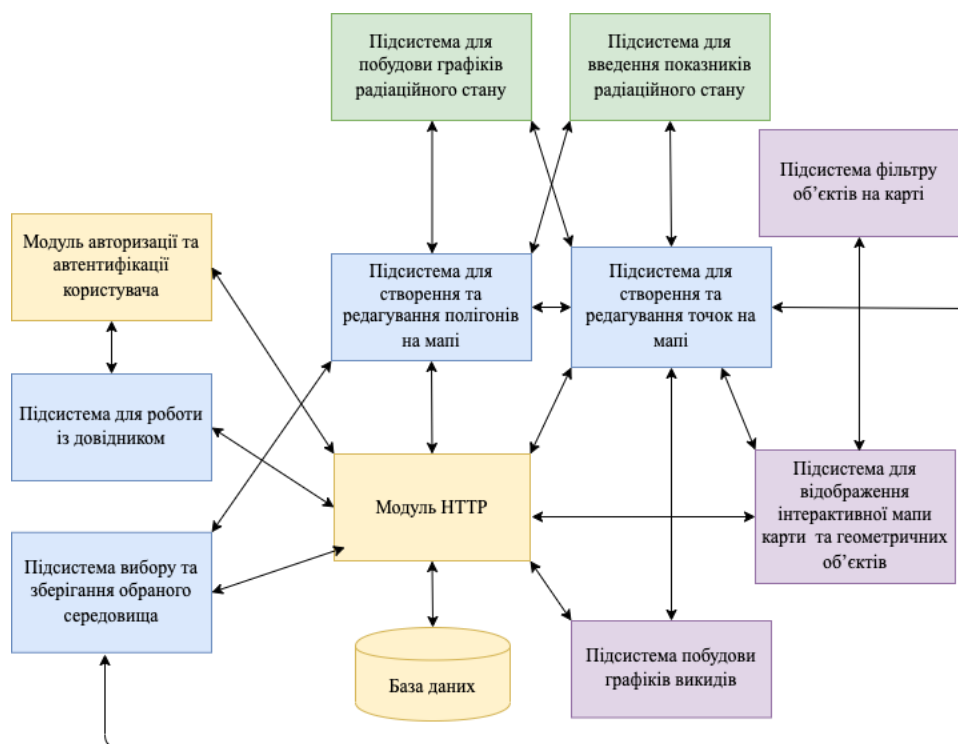


Рисунок 1.2 — Структура взаємодії між різними модулями KEEM

Функція модуля економіста полягає в можливості експерта швидко оцінити показники економічно-екологічного моніторингу за такими критеріями як «витрати» та «економічна ефективність» [4]. Він має змогу запропонувати економічно-вигідний захід для вирішення певної екологічної проблеми.

Основною задачею медичного працівника є моніторинг впливу поточних проблем екології на здоров'я та оцінка подальшої оцінки життя людей в певному регіоні. Хоча медичний працівник не є ключовою фігурою в сучасних системах моніторингу довкілля, його аналіз ризиків та впливу на здоров'я може допомогти в розробках заходів для покращення певної екологічної ситуації.

Еколог є головною фігурою в системі, оскільки саме проблеми навколишнього середовища є предметом дослідження та контролю ситуації. Еколог використовує розроблений модуль для роботи з картами для нанесення різної інформації про

забруднення або регіони шкідливих викидів в атмосферу, а також використовує ці дані для моніторингу, проведення розрахунків та поглибленого дослідження екологічної ситуації в заданому регіоні [4].

Роль експерта юриста у КЕЕЕМ полягає у формуванні набору визначених законом країни заходів, які регламентують певну екологічну ситуацію, що досліджується у регіоні, а також надання оцінки законності запропонованих іншими експертами заходів щодо покращення екологічного стану визначеної області або ліквідації пагубних наслідків діяльності людини. Він користується інформацією про екологічний стан досліджуваної зони і прогнозом його розвитку та пропонує комплекс заходів щодо покарання порушників чинного законодавства. До кожного заходу експерт-юрист прикріплює відповідний законодавчий документ, формуючи таким чином законодавчу базу.

Особа, що приймає рішення, враховуючи дані, що надали інші експерти, підводить підсумки для визначення доцільності та можливості впровадження певного заходу направленого на покращення екологічної ситуації. Крім того, особа, що приймає рішення виступає також в ролі адміністратора системи, тому може реєструвати нових експертів, а також редагувати чи зберігати у віддаленій базі даних інформацію про вже існуючих експертів.

1.5 Огляд існуючих інструментів оптимізації баз даних

Інструменти оптимізації запитів SQL визначають найкращий спосіб виконання запиту, аналізуючи різні плани запитів і визначаючи, який із них забезпечує найкращу продуктивність.

До показників ефективності належать:

- час виконання запиту;
- тривалість запиту;
- процесорний час;
- логічне та фізичне читання.

1.5.1 EverSQL Query Optimizer

Гарним прикладом інструменту оптимізації запитів є програмне забезпечення «EverSQL Query Optimizer». Ця система виконує оптимізацію запитів SQL і надає можливість спостереження за базою даних з використанням штучного інтелекту. Інструмент не потребує доступу до конфіденційних даних бази даних, тому він є безпечним для використання у комерційному та державному секторі.

Розумні алгоритми на основі штучного інтелекту автоматично оптимізують SQL-запити, переписуючи їх та додаючи індекси.

Функція порівняння коду надає користувачам представлення, що змінилося внаслідок оптимізації запитів (рисунок 1.3).

original → modified	
@@ -1,16 +1,22 @@	
1 SELECT	1 SELECT
2 p.id,	2 p.id,
3 - count(DISTINCT c.id)	3 + (
	4 + SELECT
	5 + count(DISTINCT c.id)
	6 + FROM
	7 + comments AS c
	8 + WHERE
	9 + c.PostId = p.id
	10 + LIMIT
	11 + 1
	12 +)
4 FROM	13 FROM
5 posts AS p	14 posts AS p
6 - LEFT JOIN comments AS c ON c.PostId = p.id	15 WHERE
7 WHERE	16 p.AnswerCount > 3
8 p.AnswerCount > 3	17 AND p.title LIKE '%optimized%'
9 AND p.title LIKE '%optimized%'	18 + AND p.CreationDate >= '2017-01-01 00:00:00'
10 - AND DATE(p.CreationDate) >= '2017-01-01'	
11 - GROUP BY	19 ORDER BY
12 - p.id	20 p.CreationDate
13 ORDER BY	21 LIMIT
14 p.CreationDate	22 100
15 LIMIT	
16 100	

Рисунок 1.3 — Функція порівняння коду ПЗ «EverSQL Query Optimizer»

Завдяки ненав'язливому датчику інструмент постійно відстежує продуктивність бази даних, генеруючи інформацію про оптимізацію, з покроковим планом реалізації.

Крім того, ПЗ «EverSQL Query Optimizer» пропонує рекомендації щодо індексації для кількох запитів одночасно. Інструмент підтримує СКБД MySQL, MariaDB та PerconaDB. Система є платною та обмежена лише оптимізацією запитів.

1.5.2 APEXSQL Plan

Інструмент оптимізації запитів «APEXSQL Plan» дозволяє користувачам створювати діаграми плану виконання. Програмне забезпечення дозволяє зменшувати, змінювати та спостерігати за планом виконання у режимі реального часу з інформацією про потік даних.

На рисунку 1.4 наведено приклад інтерфейсу користувача ПЗ «APEXSQL Plan» і плану виконання запиту.

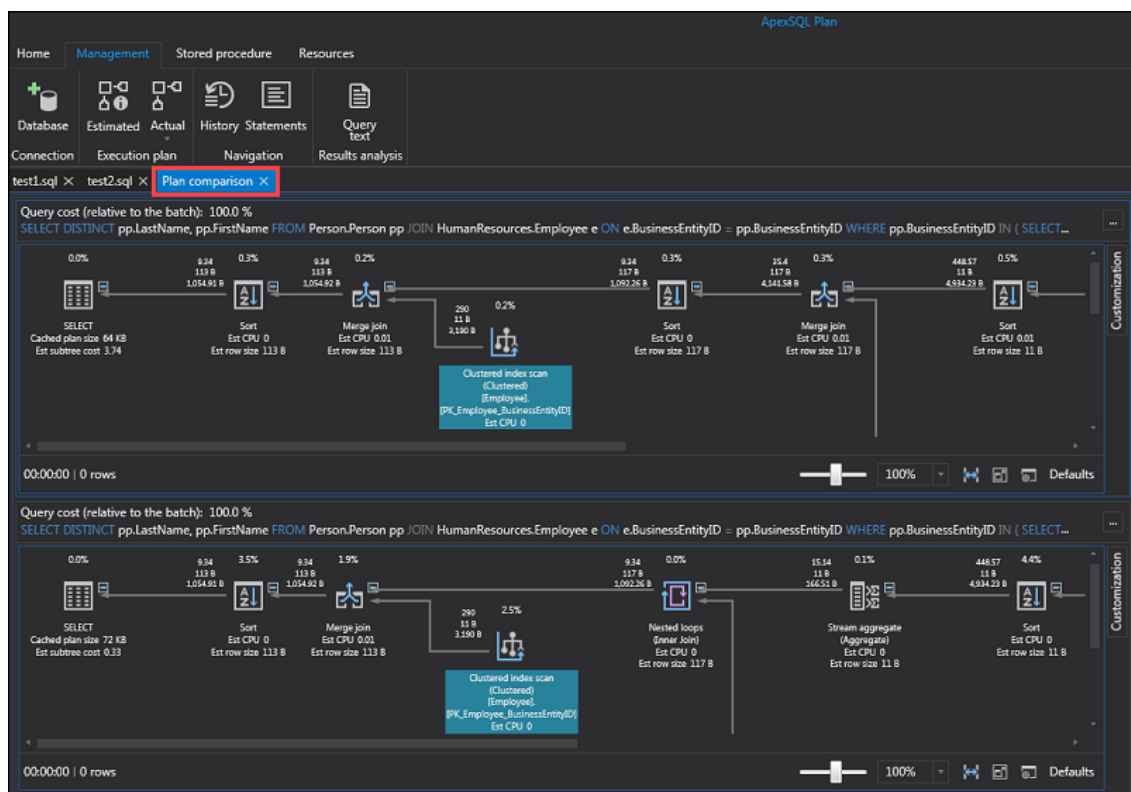


Рисунок 1.4 — Інтерфейс користувача ПЗ «APEXSQL Plan»

До інших функцій інструменту належать:

- керування властивостями для кожної операції плану виконання;
- налаштування складових (вузлів, зв'язків);
- управління та перегляд деталей для кожного потоку даних у плані виконання;
- визначення відсутності індексів і їх відтворення у плані виконання.

Головним недоліком ПЗ «APEXSQL Plan» є обмеженість роботи лише з СКБД SQL Server та нездатність переписувати існуючі запити.

1.5.3 Idera DB Optimizer

Програмне забезпечення «Idera DB Optimizer» - це оптимізатор баз даних, сумісний із SQL Server, Oracle, DB2 та Sybase, і він чудово підходить для підприємств середнього розміру.

Інструмент здатний визначати проблеми з продуктивністю в запитах до бази даних і пропонувати рекомендовані зміни для оптимізації запитів.

Оптимізатор має потужні можливості для візуалізації роботи бази даних, приклад графічного аналізу запитів у ПЗ «Idera DB Optimizer» нижче (рисунок 1.5).

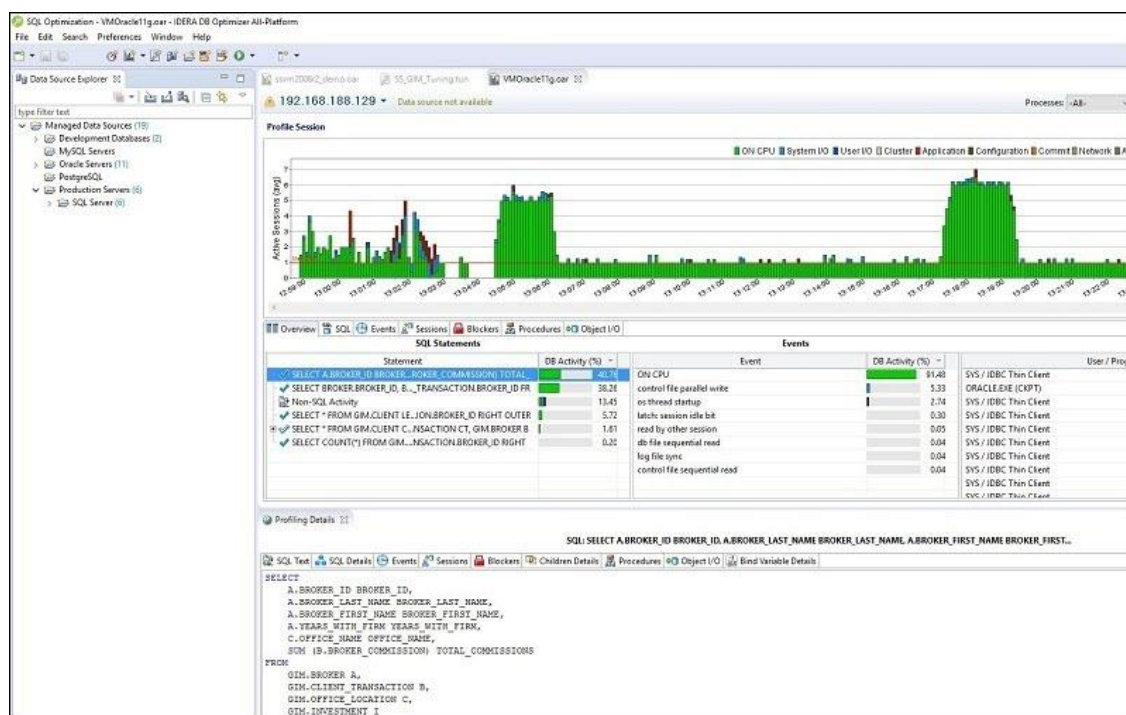


Рисунок 1.5 — Графічний аналіз запитів у ПЗ «Idera DB Optimizer»

Додаткові функції оптимізатора:

- моніторинг проблемних місць продуктивності;
- візуальне представлення аналізу часу очікування;
- рекомендація альтернативних шляхів виконання SQL-запитів;
- функція рекомендованої зміни запиту в один дотик.

До недоліків програмної системи можна віднести неспроможність роботи з СКБД MySQL, орієнтир на бази даних середніх розмірів та платна основа.

1.5.4 DbForge Studio for MySQL

Інструмент оптимізації запитів «DbForge Studio» містить різні інструменти керування сервером для здійснення моніторингу продуктивності та налаштування запитів SQL.

Програмне забезпечення «DbForge Studio» допомагає контролювати продуктивність запитів і оптимізувати їх, пропонуючи форматування SQL, фрагменти SQL і навігацію по коду для полегшення керування запитами (рисунок 1.6).

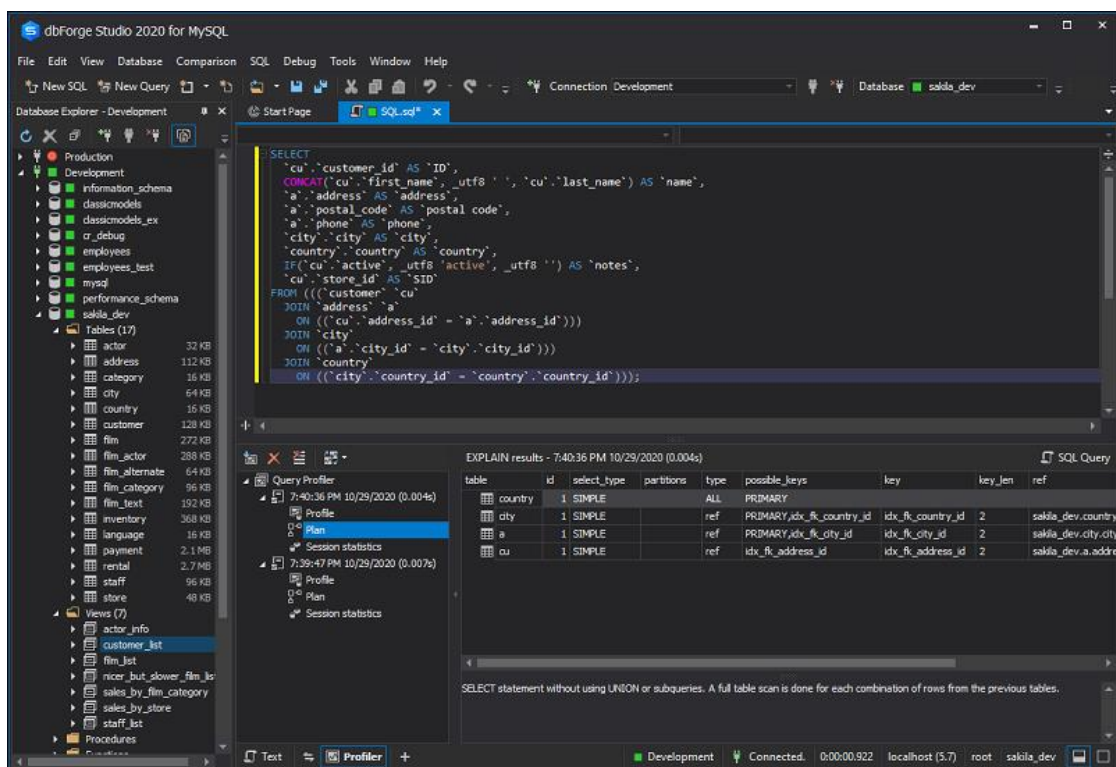


Рисунок 1.6 — Інтерфейс оптимізатора запитів ПЗ «DbForge Studio»

Особливості оптимізатора запитів:

- конструктор і профайлер запитів, інструмент оптимізації і дослідження проблем продуктивності запитів;
- система автодоповнення при введенні запитів;
- провідник та коригувальник коду запиту;
- дизайнер звітів проблем продуктивності;
- менеджер для вирішення проблем фрагментації індексів.

Недоліком системи є її платна основа та орієнтир тільки на оптимізацію запитів.

1.5.5 AppOptics APM

Особливістю ПЗ «AppOptics APM» є його хмарна основа. Він надає інструментарій моніторингу продуктивності, який містить утиліти оптимізації бази даних, визначає першопричину проблем із продуктивністю запитів і допомагає користувачам їх вирішити.

До підтримуваних СКБД належать MySQL, SQL Server, Oracle, PostgreSQL та Apache. Зображення панелі моніторингу інфраструктури зображено на рисунку 1.7.



Рисунок 1.7 — Приклад панелі моніторингу ПЗ «AppOptics APM»

Можливості інструменту:

- перевірка конкретних запитів на проблеми з продуктивністю;
- фільтрація неефективних об'єднань і стовпців, які потребують індексації;
- механізм аналізу для забезпечення ефективної взаємодії запитів і бази даних;
- аналіз бази даних незалежно від типу хостингу.

Головним недоліком системи є її платна основа, оптимізаційний функціонал системи в основному сконцентрований на запитах, коли сервер залишається без уваги.

1.6 Висновки до розділу

В даному розділі було описано постановку задачі магістерської дисертації, де головним завданням дослідження є набір рекомендацій з оптимізації роботи бази даних, застосований до системи комплексного еколого-економіко-енергетичного моніторингу, для забезпечення швидкої роботи користувачам системи (експертам). До переліку задач програмного продукту входить: мінімізація часу очікування завантаження сайту моніторингової системи, зменшення навантаження на сервер бази даних, збільшення швидкості виконання запитів.

Також в розділі було описано взаємодію програмних модулів оптимізації та відображено її на схемі цільових компонентів. Було окремо розглянуто кожен модуль оптимізації, з вказанням необхідної для повноцінної роботи вхідної інформації та результатом роботи модуля у вигляді вихідної інформації.

Додатково до модулів оптимізації було виконано опис модулів системи КЕЕЕМ і схематично відображено структуру їх взаємодії між собою. Надано детальний опис функцій модулів, призначених для відповідних експертів системи та описано принцип їх роботи з функціоналом системи.

Для розуміння напрямку дослідження було виконано аналіз аналогічних програмних засобів, виокремлено їх переваги та недоліки. Більшість аналогів зосереджені лише на оптимізації та реструктуризації запитів до бази даних, коли інші аспекти бази даних, такі як структура та налаштування серверу, ігноруються. Основним недоліком таких систем є їх платна основа та обмеженість конкретними СКБД, що робить їх придатними лише для вузького кола існуючих баз даних. В той же час набір рекомендацій системи КЕЕЕМ покриває всі необхідні потреби до оптимізації бази даних та є безкоштовним для використання у базах з великими обсягами даних та для будь-яких СКБД.

2 ОГЛЯД ПРАКТИЧНИХ МЕТОДІВ ВИРІШЕННЯ ПОСТАВЛЕНОГО ЗАВДАННЯ

Під час дослідження питання оптимізації у реляційних базах даних частіше за все йдеться мова про оптимізацію запитів. До оптимізації реляційних запитів входять два різні аспекти. По-перше, це внутрішнє завдання СКБД, що полягає у визначенні найбільш оптимального (ефективного) способу виконання реляційних запитів. По-друге, це завдання програміста (або кваліфікованого користувача): воно полягає у написанні таких реляційних запитів, для яких СКБД змогла б використовувати ефективніші способи знаходження даних. Розглянемо перший аспект більш детально.

2.1 Обробка запиту до реляційних СКБД

Кожна команда мови маніпулювання даними (DML) може бути виконана у різний спосіб. Визначення найбільш оптимального плану виконання запиту називається його оптимізацією. Вибором цього плану займається спеціальний компонент СКБД – оптимізатор запитів.

Виконання запиту складається з послідовності кроків, кожен із яких або фізично витягує дані з пам'яті, або виконує для цього підготовчу роботу. Послідовність кроків, яку будує оптимізатор, називають планом виконання запиту.

Обробка запиту, що надійшов до системи та представленого деякою мовою запитів, складається з етапів, зображених на рисунку 2.1.

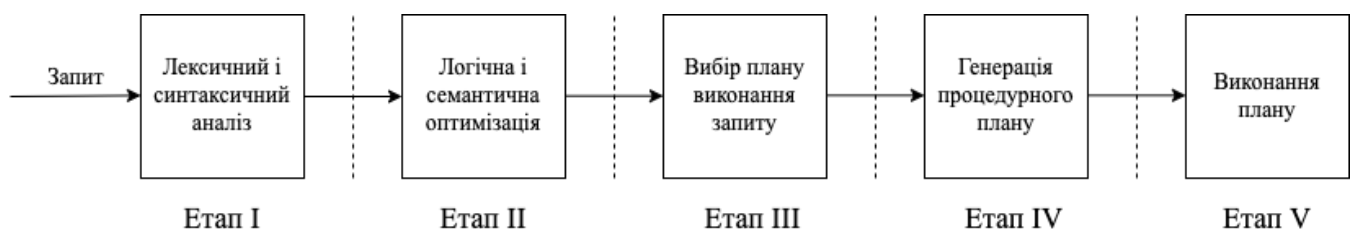


Рисунок 2.1 — Послідовність виконання запитів до реляційних СКБД

На першому етапі запит, поданий спеціальною мовою запитів, піддається лексичному та синтаксичному аналізу. При цьому виробляється його внутрішнє

представлення, що відображає структуру запиту та містить інформацію, яка характеризує об'єкти бази даних, згадані у запиті (залежності, поля та константи). Інформація про об'єкти, що зберігаються в базі даних, вибирається з каталогів бази даних (словника даних). Внутрішнє подання запиту використовується та перетворюється на наступних етапах обробки запиту.

На другому етапі запит у своєму внутрішньому представленні зазнає логічної оптимізації. При цьому можуть застосовуватися різні перетворення, які покращують початкове представлення запиту. Серед цих перетворень можуть бути еквівалентні перетворення, після проведення яких виходить внутрішнє представлення, семантично еквівалентне початковому (наприклад, приведення запиту до певної канонічної форми). Перетворення можуть бути семантичними, коли одержуване представлення не є семантично еквівалентним початковому, але гарантується, що результат виконання перетвореного запиту збігається з результатом запиту в початковій формі, при дотриманні обмежень цілісності, що існують в базі даних.

Третій етап обробки запиту полягає у виборі на основі інформації, що має оптимізатор, набору альтернативних процедурних планів виконання даного запиту відповідно до його внутрішнього представлення, отриманого на другому етапі. Основою є інформація про існуючі шляхи доступу до даних. Єдиний шлях доступу, який можливий у будь-якому випадку – це послідовне читання. Можливість використання інших шляхів доступу залежить від способів розміщення даних у пам'яті (наприклад, кластеризація даних), наявності індексів та формулювання самого запиту. На цьому етапі для кожного плану оцінюється передбачувана вартість виконання запиту за цим планом. При оцінках використовується або доступна оптимізатору статистична інформація про стан бази даних, або інформація про механізми реалізації різних шляхів доступу. З отриманих альтернативних планів вибирається найбільш оптимальний із точки зору деякого (заздалегідь обраного чи заданого) критерію. Внутрішнє представлення цього плану тепер відповідає запиту, що обробляється.

На четвертому етапі за внутрішнім представлення найбільш оптимального плану виконання запиту формується процедурне подання плану. Виконуване

представлення плану може бути програмою в машинних кодах, якщо, як у System R, система орієнтована на компіляцію запитів в машинні коди, або бути машинно-незалежним, але зручнішим для інтерпретації, якщо, як у випадку INGRES, система орієнтована на інтерпретацію запитів. У нашому випадку це не принципово, оскільки четверта фаза обробки запиту не пов'язана з оптимізацією.

Нарешті, на останньому, п'ятому етапі обробки запиту відбувається його реальне виконання відповідно до виконуваного плану запиту. Це виконання відповідної підпрограми, або виклик інтерпретатора з передачею йому для інтерпретації виконуваного плану.

Існують два принципово різні підходи до оптимізації запитів. Якщо оптимізатор використовує інформацію про механізми реалізації шляхів доступу, метод оптимізації оснований на синтаксисі (на правилах). Якщо ж основою є статистична інформація про розподіл даних, це метод оптимізації, оснований на вартості (на витратах). Розглянемо їх детальніше.

2.2 Метод оптимізації, оснований на синтаксисі

З використанням цього методу план складається виходячи з існуючих шляхів доступу та його рангів. Ранжування шляхів доступу виконується на підставі знань про правила та послідовність здійснення цих шляхів. У таблиці 2.1 як приклад наведено ранги шляхів доступу для СКБД Oracle.

Таблиця 2.1 Ранги шляхів доступу до СКБД Oracle

Ранг	Шляхи доступу
1	Один рядок за ROWID (ідентифікатором рядка)
2	Один рядок кластерного з'єднання
3	Один рядок по хеш-кластеру з унікальним або первинним ключем
4	Один рядок за унікальним або первинним ключем
5	Кластерне з'єднання
6	Ключ хеш-кластера
7	Ключ індексного кластера
8	Складений індекс
9	Індекс по одному стовпцю
10	Індексний пошук по закритому інтервалу
11	Індексний пошук по відкритому інтервалу

Продовження таблиці 2.1

12	Сортування об'єднання
13	MAX і MIN за індексованим стовпцем
14	ORDER BY за індексованим стовпцем
15	Повний перегляд таблиці

Ранг шляху доступу визначається, виходячи з знань про послідовність реалізації цього шляху. Наприклад, один рядок за хеш-кластером з унікальним або первинним ключем (ранг 3) буде вибиратися в середньому швидше, ніж один рядок за унікальним або первинним ключем (ранг 4), тому що для отримання ROWID рядка з хеш-кластера достатньо застосувати до значення ключа хеш-функцію, а для отримання ROWID за індексом потрібно зайве звернення до індексу (який може в даний момент бути відсутнім в оперативній пам'яті, що викликає звернення до диска).

План виконання запиту формується з вибраних шляхів доступу з максимальними рангами (рисунок 2.2).

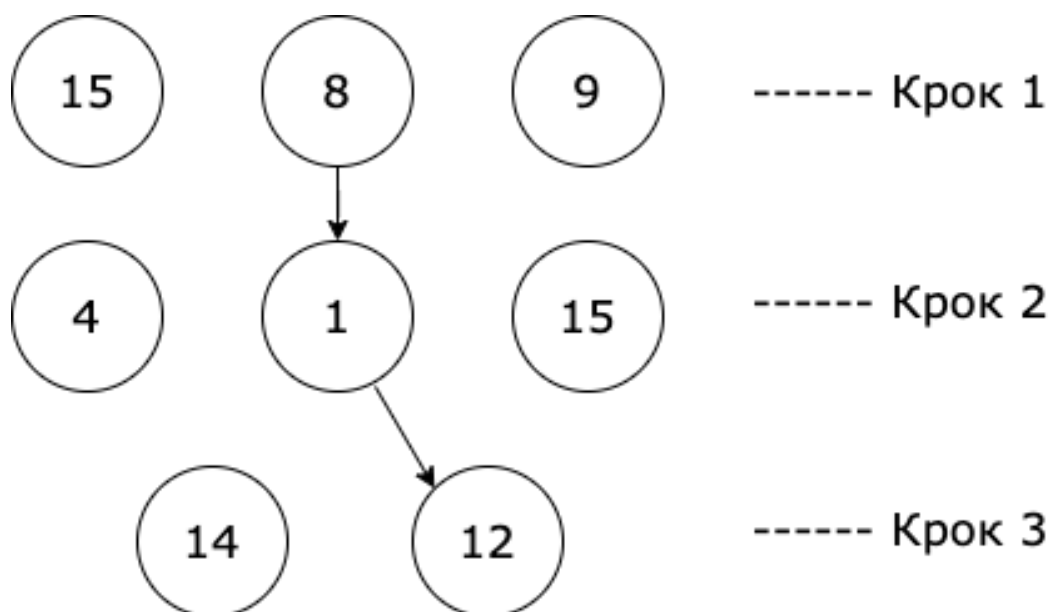


Рисунок 2.2 — Побудова плану виконання запиту методом оптимізації по синтаксису

Метод оптимізації, оснований на синтаксисі, враховує ієрархічну першість операцій. Якщо будь-якої операції існує кращий шлях її виконання, то вибирається той шлях, чий ранг вище, так як в більшості випадків він виконується швидше, чим шлях із нижчим рангом.

2.3 Метод оптимізації, оснований на вартості

У разі використання цього методу оптимізатор спочатку будує кілька можливих планів виконання запиту. Для цього він застосовує деякі евристики, тобто правила, одержані досвідченим шляхом. Ці правила дозволяють звузити простір пошуку оптимального плану завдяки тому, що неефективні плани відкидаються на початку і не розглядаються. Прикладом евристики може бути таке правило: хорошим є такий план, в якому селекція проводиться на ранньому етапі виконання запиту.

Для кожного з побудованих планів розраховується його вартість. Вона визначається на підставі статистики даних у таблицях, до яких звертається команда, та пов'язаними з ними кластерами та індексами. Статистика стосується розподілу значень даних у таблиці і може включати кількість записів у таблиці, мінімальні і максимальні значення даних по стовпцях та інші параметри. Вартість – це оцінка очікуваного часу виконання запиту з використанням конкретного плану виконання. Оптимізатор може враховувати кількість необхідних ресурсів пам'яті, вартість операцій вводу-виводу, процесорного часу та оперативної пам'яті, необхідної для виконання плану. Звичайно, для порівняння планів виконання оптимізатору необхідних критерій.

Як критерій оптимізації може виступати:

— найкраща загальна продуктивність системи. Взагалі кажучи, її можна досягти, якщо з усіх планів вибирати ті, які вимагають найменше ресурсів (пам'яті та ЦП). Це дозволить збільшити ступінь паралельності роботи системи та підвищити загальну продуктивність, хоча при цьому час виконання окремих запитів збільшиться;

— мінімальний час реакції – час, необхідний обробки першого рядка. Цей критерій найчастіше застосовується при роботі в інтерактивному режимі, але тільки для запитів, які не містять функцій, що агрегують, і не вимагають сортування даних результату.

— мінімальні витрати часу на обробку всіх рядків, до яких звертається команда. Цей критерій використовується зазвичай при роботі в пакетному режимі або коли неможливо виводити результат частинами.

З безлічі можливих планів виконання запиту (зазвичай, 2 – 3 плани) оптимізатор відповідно до критерію вибирає найкращий план (рисунок 2.3).

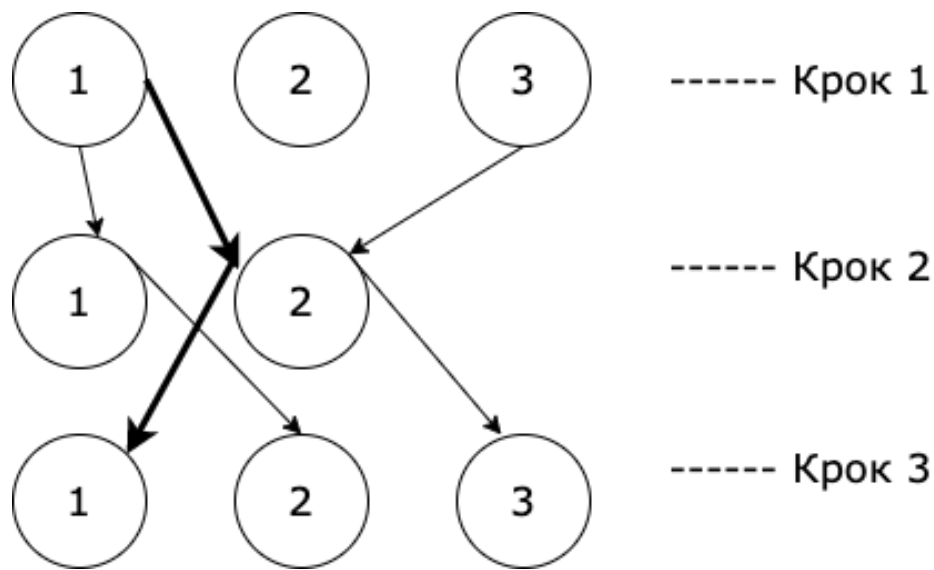


Рисунок 2.3 — Побудова плану виконання запиту за вартісним методом оптимізації

СКБД, які використовують метод оптимізації за вартістю, можуть враховувати розподіл даних та характеристики зберігання таблиць, розподіл значень у стовпцях. Останню характеристику можна визначити за допомогою гістограм. Уся множина значень стовпця впорядковується і розбивається на N інтервалів (рисунок 2.4).



Рисунок 2.4 — Приклад рівномірного та нерівномірного розподілу значень

Гістограма допомагає визначити рівномірність чи нерівномірність розподілу даних. Наприклад, на рисунку 2.4,б представлено нерівномірний розподіл даних для

деякого стовпця F. У словник даних записуються отримані значення стовпця (1, 4, 4, 5, 6, 10, 11, 20, 35, 60, 100). При аналізі запиту, наприклад, з умовою ($F < 5$) система зможе за допомогою цієї гістограми визначити, що через індекс їй доведеться вибрати щонайменше 15% записів таблиці.

Існують різні підходи до порядку збирання статистики. Деякі СКБД постійно збирають статистичну інформацію, але це може зменшити швидкодію системи. Інші дозволяють здійснювати збирання статистики періодично, наприклад, у період мінімального завантаження системи. Треті пропонують адміністратору спеціальні засоби для збору статистики, які запускаються інтерактивно за його командою. В останньому випадку в обов'язок адміністратора (адміністратора даних чи додатків) входить вибір таблиць для аналізу та періодичне оновлення статистики даних.

Гістограми корисні лише у тих випадках, коли вони відображають актуальний розподіл даних. Якщо розподіл змінюється, гістограму необхідно оновлювати; якщо зміна даних не змінює їх розподілу, то оновлення гістограми не потрібно.

Побудова гістограми не має сенсу в таких випадках:

- значення стовпця розподілені рівномірно;
- стовпець не використовується в умовах запитів;
- значення стовпця унікальні і використовуються тільки в предикатах еквівалентності.

2.4 Порядок оптимізації виконання запиту

Оптимізація виконання запиту здійснюється в наступному порядку:

1. Розрахунок виразів та умов, що містять константи.
2. Перетворення складної команди на еквівалентну їй з використанням з'єднання (виконується за потреби).
3. Якщо команда виконується над представленням, то оптимізатор зазвичай об'єднує запит на створення представлення і запит до цього представлення в одну команду.
4. Використання методу оптимізації, оснований на вартості.
5. Вибір шляхів доступу до таблиць, до яких звертається запит.

6. Вибір порядку з'єднання (якщо у запиті з'єднуються кілька таблиць, то оптимізатор визначає, які дві таблиці з'єднуюватимуться першими, яка таблиця буде наступною підключатися до результату і так далі).

7. Вибір операції з'єднання кожної команди з'єднання.

Для прикладу розглянемо наступну команду SQL:

```
SELECT empNo
```

```
FROM emp
```

```
WHERE eName = 'Tom' AND Sal > 2000;
```

Нехай таблиця *emp* має наступне правило цілісності та індекси:

- стовпець *empNo* визначений як первинний ключ, йому відповідає індекс *PK_EMPNO*;

- існує індекс *ENAME_IND* для стовпця *eName*;

- існує індекс *SAL_IND* для стовпця *Sal*.

Можливі наступні шляхи доступу:

- повний перегляд таблиці (ранг 15);

- доступ по індексу *ENAME_IND*. Цей шлях стає доступним за умовою *eName* = 'Tom' (ранг 9);

- доступ за допомогою відкритого інтервалу *SAL > 2000*, використовуючи індекс *SAL_IND* (ранг 11).

Індекс *PK_EMPNO* недоступний, так як не відбувається звернення до поля *empNo*. Оптимізатор вибере доступ по індексу з рангом 9.

При виборі шляху доступу в вартісному методі оптимізації для оцінки кожного плану виконання оптимізатор використовує статистику, на вибір оптимізатора також можуть вплинути підказки, задані користувачем в команді.

Розглянемо запит, який вибирає працівників з номером більше 7500:

```
SELECT *
```

```
FROM Emp
```

```
WHERE empNo < 7500;
```

Статистика для стовпця *empNo*, зокрема, включає значення *HIGH_VALUE* та *LOW_VALUE*. Якщо нема гістограм, то оптимізатор припускає, що значення

рівномірно розподілені на інтервалі [*LOW_VALUE*; *HIGH_VALUE*], і може визначити процент значень, які потрапляють в інтервал до 7500. Доступ буде здійснюватися за індексом, якщо цей процент невеликий (конкретне ключове значення залежить і від інших параметрів, наприклад, кількості записів у блоку).

Розглянемо запит, який обирає всіх працівників чоловіків:

```
SELECT *  
FROM Emp  
WHERE sex = 'м';
```

Оптимізатор припускає, що значення 'м' та 'ж' рівномірно розподілені по таблиці, тобто приблизно половина записів має значення 'м'. В цьому випадку оптимізатор вибере повний перегляд таблиці навіть за наявності індексу за стовпцем *sex*. Якщо є гістограма і розподіл даних нерівномірний, то оптимізатор може прийняти рішення про використання індексу.

2.5 Налаштування застосунків

Налаштування команд SQL, які застосовуються в застосунку – це один із основних способів підвищення продуктивності системи. Це налаштування повинно виконуватися кожним розробником.

Для налаштування застосунків необхідно мати уявлення про порядок і механізми реалізації запитів в СКБД. Підготовлені до виконання SQL-оператори зазвичай вміщуються в роздільну SQL-область. Перед початком виконання запиту система перевіряє, чи є у цій області аналогічний запит: якщо є, то він відправляється на виконання, оминаючи стадію попередньої обробки (компіляції). Формуючи запити таким чином, щоб вони співпадали з уже наявними в SQL-області, можна виключити попередню обробку запиту, що є важливим моментом оптимізації застосунків.

Основні інформаційні потоки між користувачем, оперативною пам'яттю і базою даних наведені на рисунку 2.5.

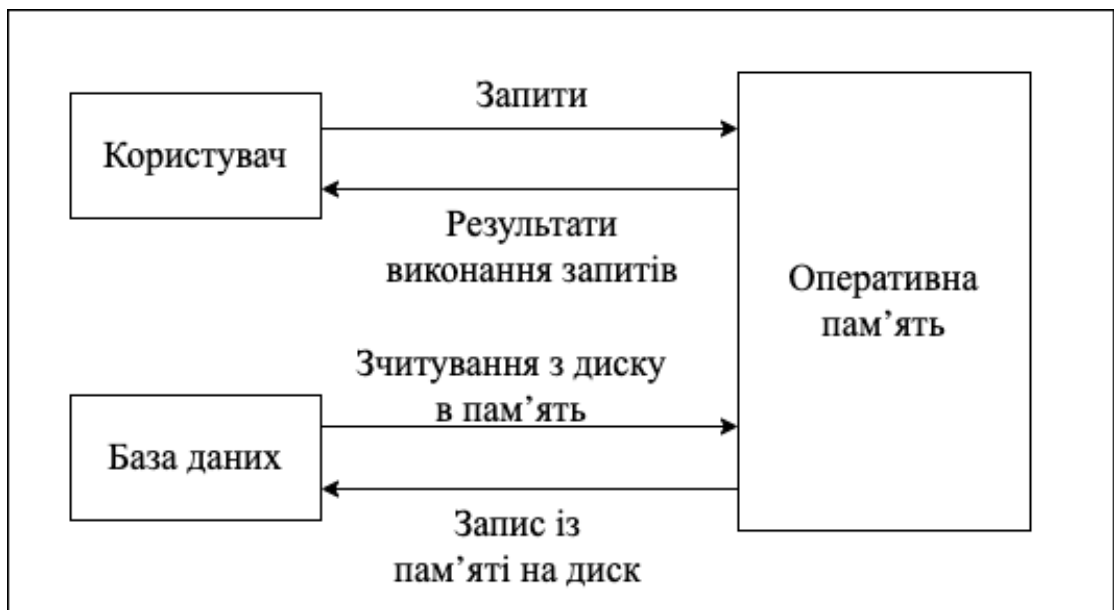


Рисунок 2.5 — Інформаційні потоки в базі даних

В оперативній пам'яті зберігаються всі результати виконаних раніше запитів до тих пір, поки ця пам'ять не буде потрібна для запису результатів наступних запитів.

При налаштуванні команд SQL важливо пам'ятати, що виконуючи налаштування однієї з них можна вплинути (і не завжди добре) на інші команди. Наприклад, при створенні додаткового індексу можуть застосовуватися плани виконання всіх команд, працюючих з цією таблицею.

2.6 Висновки до розділу

В даному розділі було досліджено методи оптимізації, які використовує оптимізатор запитів у своїй роботі, а саме – метод оптимізації оснований на синтаксисі та вартісний метод оптимізації. Описано їх специфіку та деталізовано порядок оптимізації виконання запиту. Для кращого розуміння роботи методів, було додано приклади SQL-запитів з покроковим розбором їх роботи. Додатково було описано принцип налаштування застосунків із вказанням важливих застережень.

Також було схематично зображено основні інформаційні потоки між користувачем, оперативною пам'яттю і базою даних. Описано процес зберігання та очищення результатів запитів (кеш запитів) в оперативній пам'яті сервера баз даних.

3 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ ВЕБ-ОРІЄНТОВАНОЇ СИСТЕМИ КОМПЛЕКСНОГО ЕКОЛОГО-ЕКОНОМІКО-ЕНЕРГЕТИЧНОГО МОНІТОРИНГУ

База даних використовується для зберігання великих обсягів даних моніторингової системи, яка складається з клієнтського застосунку - для локальної роботи та веб-орієнтованої системи - для роботи через мережу Інтернет. В даній магістерській дисертації розглядається база даних веб-орієнтованої системи, оскільки саме до неї надходить велика кількість одночасних запитів від багатьох користувачів, тому для надійної роботи системи – запити до бази повинні виконуватися швидко.

Дана веб система є частиною експертних систем, а саме системи “Комплексного еколого-економіко-енергетичного моніторингу”, але основне призначення веб системи - це відкритий екологічний моніторинг та зображення даних екологічного моніторингу, через що вона також є частиною програм по моніторингу довкілля [5].

Експертна система в свою чергу – це пакет прикладних програм довідкового характеру, що дозволяє користувачу приймати кваліфіковане рішення в процесі його основної діяльності.

3.1 Принцип та порядок роботи системи комплексного еколого-економіко-енергетичного моніторингу

1. Аналітик описує проблему, вводить її назву та опис в базу даних.
2. Юрист знаходить всі документи, що мають відношення до проблеми та прикріплює їх до задачі.
3. Еколог, медичний працівник та економіст збирають всі потрібні параметри і проводять розрахунки. На основі прикріплених документів та отриманих внаслідок аналізу даних експерти пропонують заходи, які мають розв’язати проблему.
4. Юрист розглядає кожен із заходів та надає оцінку щодо законності запропонованих дій.
5. Аналітик, переглядає кожен запропонований захід та описи експертів, вирішує чи доцільно його реалізовувати.

3.2 Засоби розробки програмного продукту

Основою комплексу еколого-економіко-енергетичного моніторингу є клієнт-серверна архітектура.

Загальною мовою програмування всього комплексу є JavaScript [6], яка використовується у зв'язці з мовою гіпертекстової розмітки HTML [7] та каскадною таблицею стилів CSS [8], що надає сторінкам веб-системи гарного вигляду.

Згідно рекомендаціям до сучасних вимог розробки, систему було побудовано на основі бібліотеки з відкритим вихідним кодом React [10].

Текстовим редактором для розробки проєкту слугував зручний та швидкий Visual Studio Code.

Для контролю за версіями проєкту була використана розподілена система управління версіями Git.

Сервер системи побудований на платформі NodeJS з використанням вищезазначеної мови програмування JavaScript.

Системою керування реляційною базою даних проєкту було обрано MySQL з використанням підсистеми зберігання InnoDB [13], що ідеально підходить для побудови бази з великим обсягом даних і швидким доступом.

Зв'язок між використаними інструментами зображено нижче (рисунок 3.1).



Рисунок 3.1 — Інструменти розробки програмного продукту

Також зазначимо додаткові бібліотеки та програми, які використовувалися для спрощення роботи з комплексом економіко-еколого-енергетичного моніторингу:

- бібліотека react-bootstrap для надання системі одного стилю оформлення [9];
- розширення JSX для зручного опису компонентів React [10];
- мінімалістичний фреймворк серверної частини Express [12];
- бібліотека React Ag-Grid для роботи з таблицями;
- інструмент візуального проєктування баз даних MySQL Workbench для адміністрування СКБД MySQL [13].

3.2.1 Мова програмування JavaScript

Основною мовою програмування була обрана мова високого рівня JavaScript, яка забезпечує побудову багатоплатформних продуктів, містить малий обсяг коду компіляції та на якій також побудована платформа NodeJS з менеджером пакетів NPM, який дозволяє легко керувати залежностями проєкту і додавати до застосунку лише необхідні пакети [12].

Оскільки мова JavaScript підтримується більшістю сучасних браузерів, це дозволяє писати системи, без використання транслятору, що перетворює код програми на виконуваний код.

Мова JavaScript розроблювалась для виконання скриптів на клієнтській стороні, щоб надати веб-сторінкам динамічності. Тому для взаємодії з документом JavaScript використовує так звану модель DOM. Модель дозволяє оперувати елементами сторінки, створювати обробники подій та прив'язувати їх до потрібних елементів. Наприклад, коли користувач натискає купити товар, спрацьовує обробник події, що перенаправляє користувача на сторінку для оформлення замовлення. За допомогою цієї моделі можна програмно видаляти, переставляти, додавати цілі блоки HTML, що забезпечує високу гнучкість сторінки та робить її інтерактивною.

Використовуючи цикл подій, JavaScript дозволяє гнучко працювати з асинхронними завданнями, що забезпечує високу швидкодію всього комплексу. Загальна схема роботи циклу подій зображена на рисунку 3.2.

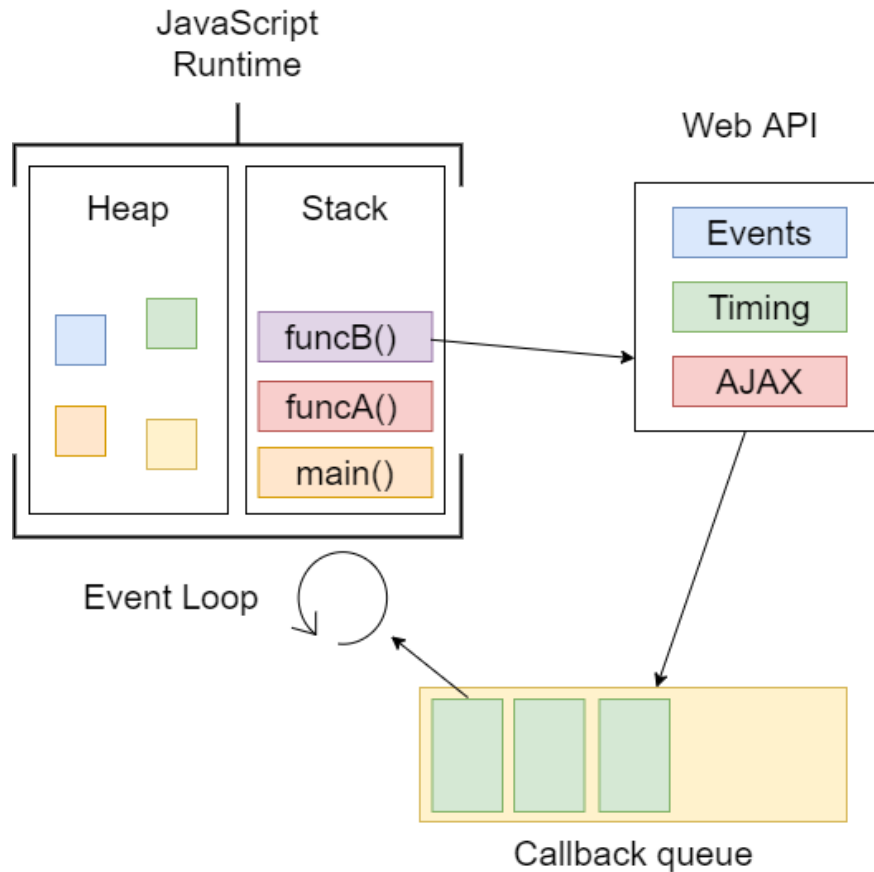


Рисунок 3.2 — Схема роботи циклу подій у JavaScript

Додатково JavaScript спостерігає за розподілом пам'яті в програмі та вчасно очищає непотрібні дані. За це відповідає збирач сміття, що звільняє пам'ять у момент, коли об'єкти більше не будуть використовуватися. Важливим пунктом розподілу пам'яті є те, що JavaScript виділяє її динамічно, тільки за необхідності. Через це вдається позбутися таких типових помилок, як вихід за межі масиву чи операції з неіснуючими комірками пам'яті.

Декілька слів хочеться додати про платформу NodeJS, оскільки вся серверна частина даної веб-системи побудована на ній. NodeJS дозволяє працювати зі сторонніми бібліотеками та пакетами, що можуть бути завантажені через пакетний менеджер NPM.

Таким чином мова програмування JavaScript є хорошим вибором для створення веб-систем, включаючи можливість взаємодії з великою кількістю складних інтерфейсів [11].

3.2.2 Мова гіпертекстової розмітки HTML

Для відображення результатів у вікні браузера користувача використовується мова гіпертекстової розмітки HTML, оскільки вона є основним засобом зображення контенту веб-сторінок [7]. Браузер показує сторінку користувачу як результат обробки HTML документа. У поєднанні з каскадною таблицею стилів CSS та скриптами JavaScript формуються веб-сторінки та сучасні сайти.

Актуальна мова розмітки HTML5 дозволяє:

— розробку семантично структурованого HTML документа шляхом виділення таких змістовних блоків, як заголовки, основні частини, списки, секції, параграфи, таблиці;

— створення форм;

— використання внутрішніх сховищ браузера для зберігання даних;

— додавання зображень, відео та інших видів медіаконтенту;

— використання веб-сокетів;

— використання тегу canvas для роботи з графікою.

Важливо також зазначити, що в HTML виділяють поняття «коректності». Для отримання коректного HTML необхідно дотримуватись спеціально визначених правил атрибутів та контенту елементів. Повне дотримання цих правил забезпечує більшу доступність HTML документів та високий рейтинг сторінки в мережі Інтернет. Для редагування HTML можна користуватись будь-яким текстовим редактором, коли для перегляду документу використовується браузер.

3.2.3 Каскадна таблиця стилів CSS

Для візуального оформлення системи комплексного моніторингу була використана каскадна таблиця стилів CSS.

CSS — це мова, що використовується для описання стилів для HTML документа [8]. Це дозволяє оформити вигляд та відобразити HTML документ [7]. Каскадна таблиця використовується для задання кольору, шрифтів, полів, висоти та ширини, позиціювання елементів та іншої візуальної стилістики.

Перевагами поєднання HTML і CSS є їх розподіл обов'язків. HTML відповідає

за зміст та положення елементів веб-сторінки, а CSS — за візуальну складову. Це дозволяє оформити один і той самий HTML документ різними таблицями стилів, що в результаті дає дві різноманітні сторінки та покращує гнучкість всієї системи.

Перевагами каскадної таблиці стилів є:

- можливість стилізувати один і той же елемент у залежності від умов. Наприклад, змінювати стилі при наведенні чи при зменшенні розміру вікна;
- прискорення завантаження веб-сторінок через розподілення окремо стилів та окремо розмітки;
- розподіл файлів каскадної таблиці стилів у залежності від потреб;
- точний контроль над зовнішнім виглядом веб-сторінок.

Задля забезпечення одного стилю веб-системи, було використано бібліотеку стилів react-bootstrap, яка надає можливість використовувати стилізовані елементи та класи стилів [9].

3.2.4 Бібліотека React

У зв'язку зі стрімким розвитком технологій, все більше уваги приділяється готовим інструментам для розробки. Ці інструменти слугують для полегшення процесу розробки та надають можливість легко масштабувати й довго підтримувати програмне забезпечення. Для реалізації системи КЕЕЕМ таким інструментом було обрано бібліотеку React, оскільки вона забезпечує гарну швидкодію та проста у використанні [10]. Робота з цією бібліотекою являє собою написання компонентів, що у взаємодії між собою формують сучасні користувацькі інтерфейси [11].

Кожен компонент має свою чітко визначену роль і виконує поставлене перед ним завдання. Ізолювання логіки всередині компонента дозволяє використовувати компоненти повторно. Компоненти можуть приймати вхідні параметри для обробки згідно з поставленою задачею, що дає змогу не вдаватись у деталі реалізації компонентів, а лише забезпечити обумовлений набір вхідних параметрів («контракт даних») з їх обов'язковим опрацюванням належним чином.

У основі створення складних інтерактивних інтерфейсів React використовує декларативний спосіб опису. Розробнику необхідно лише описати, як різні частини

інтерфейсу виглядають у кожному стані і React ефективно оновить й перемалює лише потрібні компоненти, коли вхідні дані будуть змінені [10]. Для забезпечення такої роботи застосунків, React використовує абстракцію VirtualDOM (рисунок 3.3).

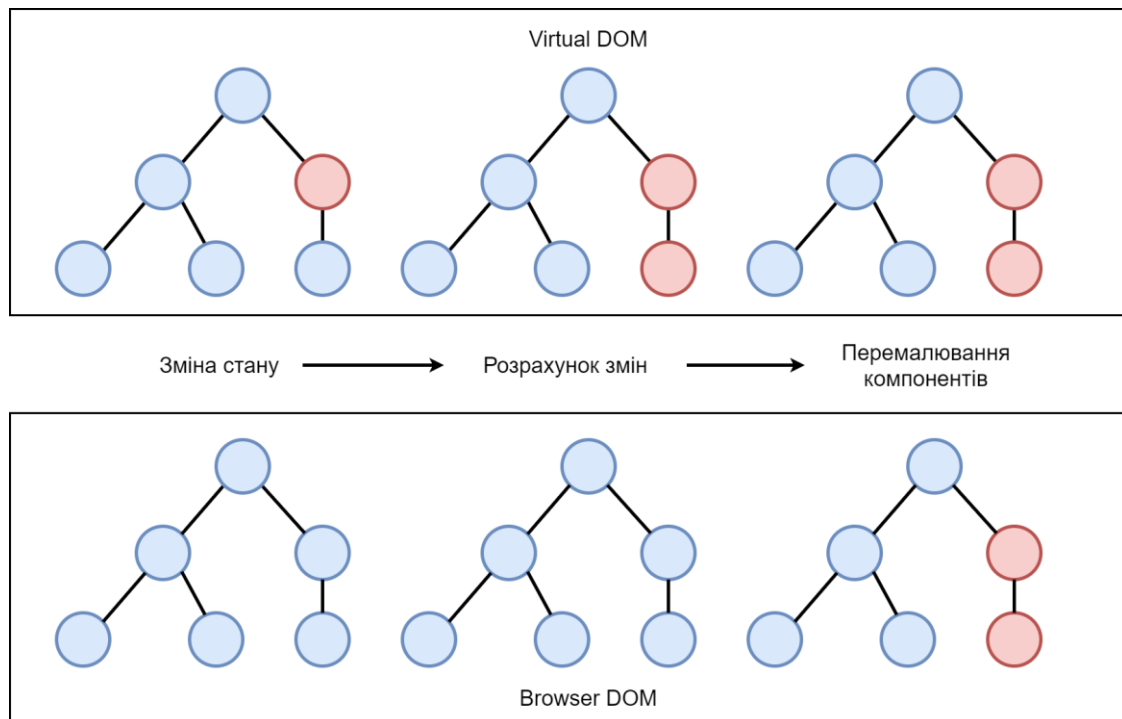


Рисунок 3.3 — Перебудова необхідних компонентів за допомогою VirtualDOM

Ефективним методом полегшенням при описі компонентів є розширення JSX для React, що дозволяє використовувати HTML прямо у шаблонах. Це значно спрощує взаємодію із DOM та спосіб показу динамічних даних, тому що немає потреби напряму звертатись до DOM і змінювати їх вручну. Все це робить JSX, що дозволяє уникнути важко оброблюваних помилок, непередбачуваної поведінки системи та зменшити загальну кількість коду.

Додатково для React була використана офіційна утиліта Create React App, що дозволяє швидко розгортати середовище для розробки і надало можливість сконцентруватись на розробці функціоналу. Всередині цієї утиліти є статичний збирач модулів, який дозволяє правильно і оптимально збирати всю систему. Додатково були введені команди для кінцевої збірки, з максимальною мініфікацією та обфускацією коду, для отримання мінімального розміру фінальної збірки.

3.2.5 Система контролю версій Git

Для задання версій програмного продукту було обрано найбільш популярну систему Git.

До виникнення систем контролю версій, розробка програмного забезпечення часто стикалась із труднощами ведення версій програмного продукту. Часто виникали проблеми вирішення актуального стану проєкту. Іноді невеликі зміни у програмному коді спричиняли повний вихід системи із ладу й пошук таких змін був надзвичайно складним завданням. Існують методи ведення версій, що прив'язувались до дати зміни файлу, а також методи індексів, коли до проєкту після кожної зміни додавався новий індекс. Проте все це не вирішувало основної проблеми підтримки та ведення версій проєкту. Враховуючи такі проблеми, почались розробки спеціально направленою програмного забезпечення, що дозволяє зручно слідкувати за кожною зміною у проєкті.

Мотивацією створити таку систему стала розробка ядра Linux. Такою системою став Git.

На сьогодні Git є однією з найефективніших, надійних і високопродуктивних систем керування версіями, що надає гнучкі засоби нелінійної розробки.

До плюсів Git можна віднести наявність двох репозиторіїв: локального, що зберігається на комп'ютері розробника і дозволяє працювати навіть без мережі Інтернет та віддаленого, що знаходиться на сервері, де відбувається публікація виконаної роботи.

Суть роботи Git полягає у тому, щоб робити «знімки» стану проєкту на момент фіксації. Фіксація - це об'єкт, який містить посилання на знімок стану проєкту, метадані автора, коментарі та вказівники на батьківські фіксації. Можна виконувати злиття фіксацій, розбивати на частини, змінювати місцями, змінювати коментарі, видаляти, копіювати з однієї гілки в іншу. У такий спосіб, вся розробка складається з множини фіксацій, кожен з яких можна знайти і визначити які зміни робились, коли вони були зроблені та ким (рисунк 3.4). Тому не виникає потреби вручну проставляти індекси та робити будь-яку додаткову роботу.

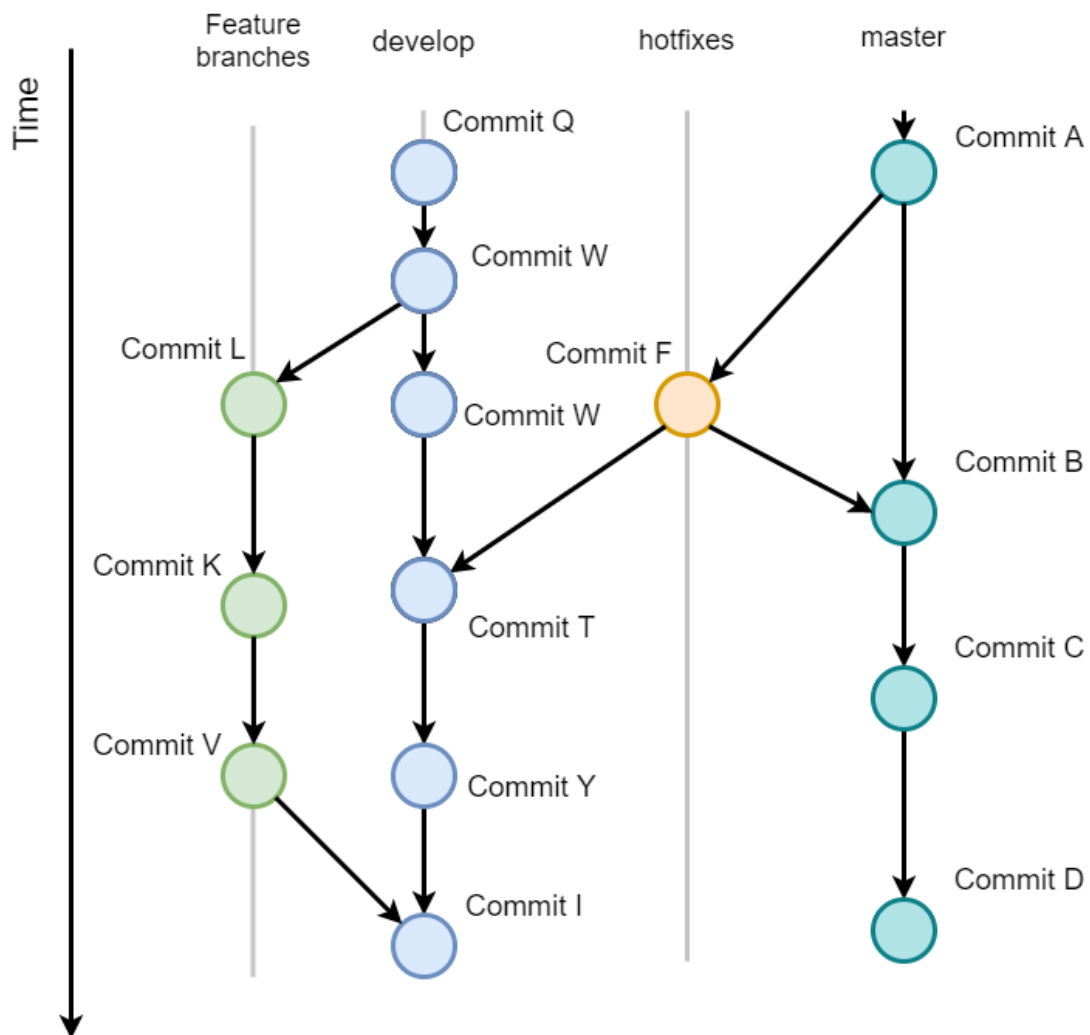


Рисунок 3.4 — Приклад процесу розробки за допомогою Git

Під час розробки веб-системи Git дозволяв стежити за змінами й швидко знаходити помилки у програмі. Також, з метою розподілу різних частин функціоналу, часто використовувались гілки, що забезпечували зручний спосіб для ведення розробки.

3.2.6 Фреймворк Express

Щоб забезпечити швидкодію з малим фінальним розміром серверної частини було обрано фреймворк Express [12]. Цей фреймворк є гнучким, мінімалістичним та швидким інструментом для роботи серверної частини, представляє широкий набір функцій та можливостей для мобільних та веб-застосунків. Маючи у своєму розпорядженні багато службових методів HTTP і проміжних обробників, можна

швидко і легко створити надійний API. Express представляє тонкий шар фундаментальних функцій веб-додатків, які допомагають розробнику працювати з широким набором можливостей NodeJS. Враховуючи мінімалізм фреймворку, масштабування програмного продукту відбувається легко, з можливістю довгої підтримки проєкту в майбутньому [10].

Окремо варто відзначити спільноту та документацію. Через свою гнучкість та мінімалізм Express заслужив довіру великої кількості розробників і біля нього сформувалось міцна спільнота. Це дозволяє розробникам легко ділитись своїми напрацюваннями та кращими методиками (англ. «best practices»). У випадку виникнення будь яких запитань або труднощів із реалізацією того чи іншого функціоналу, існують багато статей з детальними поясненнями, що майже завжди гарантують розв'язок, оскільки хтось уже мав досвід зі схожими проблемами.

3.2.7 Реляційна база даних MySQL

Однією з найпопулярніших систем керування реляційними базами даних є MySQL [13]. База даних являє собою структуровану сукупність даних. Ці дані можуть бути будь-якими — від простого списку до цілих кластерів інформації у корпоративній мережі. Оскільки в реляційній базі даних всі інформація зберігається не купою, а в окремих таблицях, тому досягається значний виграш у швидкості й гнучкості системи. Таблиці пов'язуються за допомогою відношень, що забезпечують можливість об'єднання даних із декількох таблиць. СКБД MySQL було обрано через свою швидкодію, надійність і легкість у користуванні [13].

Підсистемою зберігання було обрано InnoDB [13], оскільки таблиці InnoDB в MySQL забезпечують безпечні транзакції (рівня ACID) з можливістю фіксації транзакції, відкату та відновленню після збою. Для таблиць здійснюється блокування на рівні рядка, а також використовується метод читання без блокувань у команді SELECT. Перелічені функції дозволяють покращити взаємну сумісність і підвищити продуктивність у режимі багатокористувацького доступу, що ідеально підходить для побудови бази даних KEEEM, оскільки вона складається з великих обсягів даних і система потребує швидкого доступу до них.

Одним із корисних інструментів MySQL при розробці даного програмного продукту був MySQL Workbench — програма для візуального представлення та проєктування баз даних (рисунок 3.5).

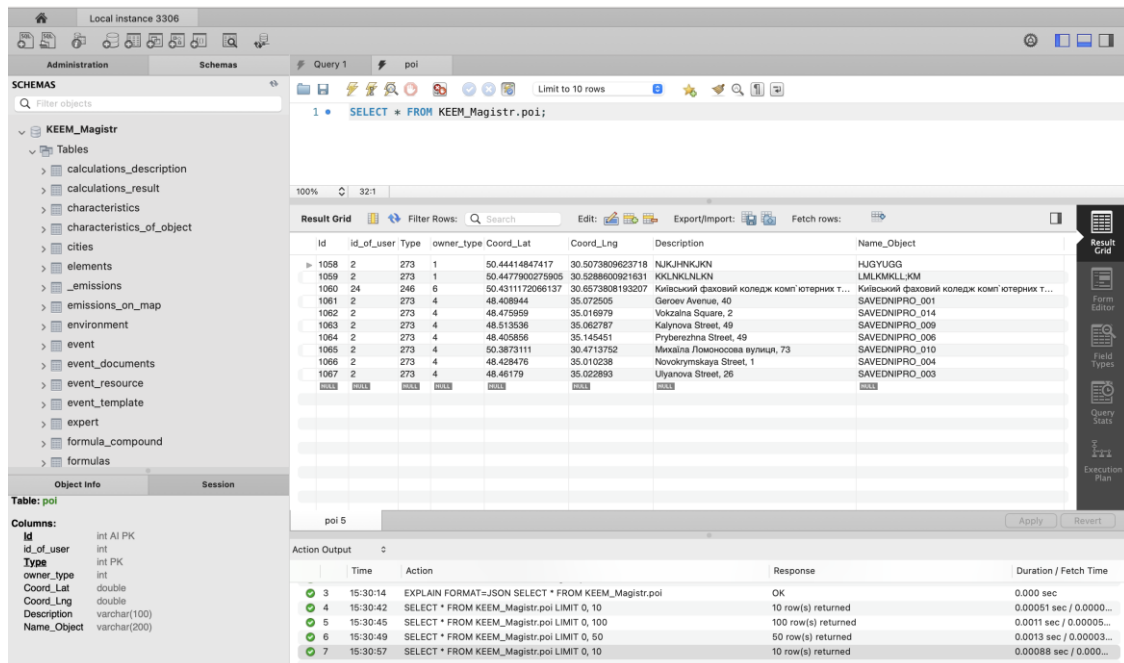


Рисунок 3.5 — Вигляд вікна MySQL Workbench

Цей інструмент дозволяє створювати, моделювати, проєктувати, редагувати та проводити інші дії із базами даних, не використовуючи інших додаткових засобів. У цьому сервісі представлена можливість використання для баз даних як локального MySQL — сервера, так і віддаленого.

Враховуючи все вищесказане, MySQL Workbench часто застосовувався при розробці, оскільки база даних для веб-системи зберігається на віддаленому сервері.

3.2.8 Бібліотека React Ag-Grid для роботи з таблицями

Дана бібліотека відома через свої широкі можливості для роботи із таблицями на веб-сторінках і серверній стороні. Вона має підтримку API для роботи як зі звичайним JavaScript, так і з усіма основними фреймворками та бібліотеками. Бібліотеку можна легко приєднати до Angular, Vue та React застосунків, тому вона високо оцінюється у колах розробників. Оскільки дана веб-система побудована на базі React бібліотеки, відповідно була використана версія Ag-Grid для React.

Приклад роботи бібліотеки зображено на рисунку 3.6.

Абревіатура	code	name	short_name	measure	rigid	voc	hydrocarbon	formula	cas
Визначення	Не знайдено опис	Не знайдено опис	Не знайдено опис	Не знайдено опис	Не знайдено опис	Не знайдено опис	Не знайдено опис	Не знайдено опис	Не знайдено опис

code	name	short_name	measure	rigid	voc	hydrocarbon	formula	cas
0	Загальні викиди	Загальні викиди	мг/м³	1	0	0	-	Містить
1	Залізо та його сполуки	Залізо та його сполуки	мг/м³	0	0	0	Fe	Містить
2	Манган та його сполуки	Манган та його сполуки	мг/м³	0	0	0	Mn	Не містить
3	Суспендовані частинк...	Суспендовані частинки	мг/м³	0	0	0	-	Дорівнює
4	НМЛОС	НМЛОС	мг/м³	0	0	0	-	Не дорівнює
5	Натрій хлорид	Нітрій хлорид	мг/м³	0	0	0	NaCl	Починається з
								Закінчується на
								Пусте
								Не пусте

Рисунок 3.6 — Зовнішній вигляд таблиці побудованої за допомогою Ag-Grid

Головними перевагами даного інструменту є широкі можливості для налаштування зовнішнього вигляду та стилізації таблиць; можливість роботи як на стороні клієнта, так і на стороні сервера, а також опції для вибірки та фільтрації елементів. За окрему плату пропонуються спеціальні панелі та сайд-бари, інтегровані графіки та компоненти. Для даної веб-системи було використано функціонал із безкоштовної версії бібліотеки, що покриває усі необхідні операції з таблицями.

3.2.9 Бібліотека Read-Excel-File

Бібліотека була застосована для забезпечення можливості миттєво заповнити маркери на мапі шляхом імпорту даних з Excel. Бібліотека забезпечує всі необхідні функції для роботи з Excel-файлами. При імпорті можна вказати конкретний лист для читання та схему формату даних.

3.3 Опис програми та її алгоритмів

Діаграма прецедентів — це діаграма, що візуалізує поведінку системи і зображує події, які можуть відбутися в середовищі.

На рисунку 3.7 зображено функції системи у вигляді діаграми прецедентів.

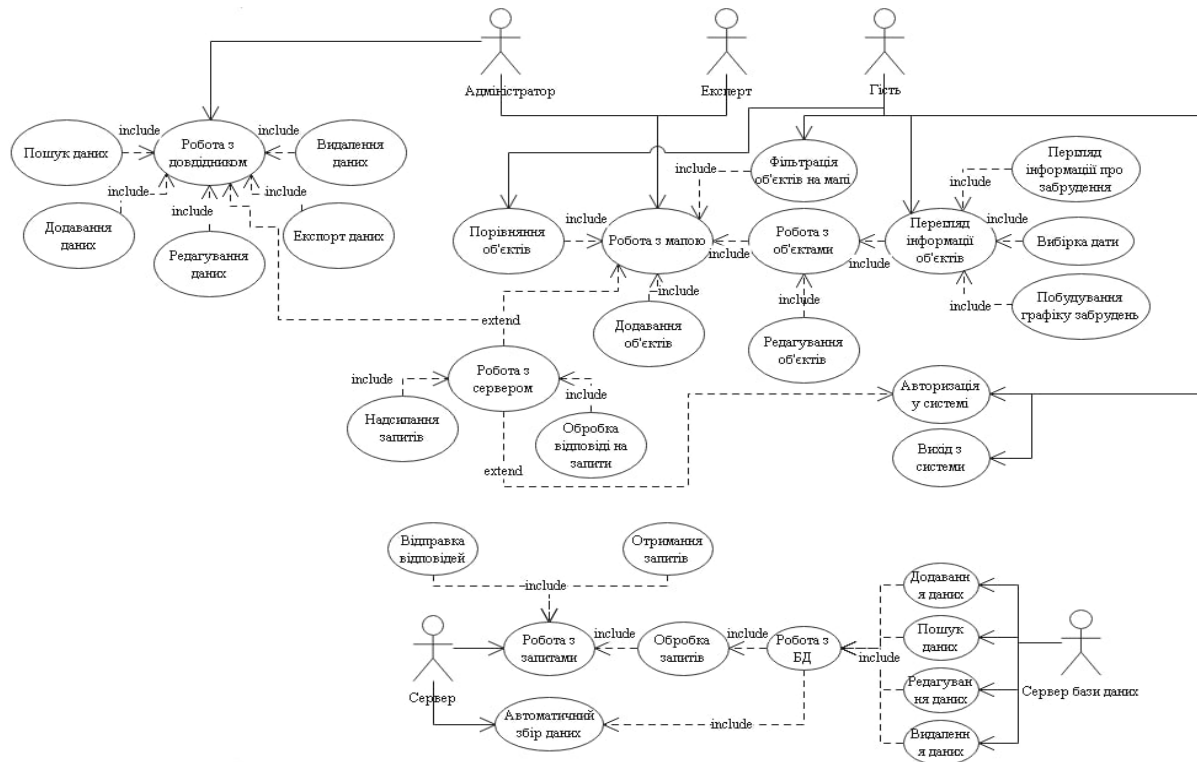


Рисунок 3.7 — Діаграма прецедентів веб-орієнтованої системи КЕЕЕМ

З діаграми прецедентів можна побачити набір акторів, кожен з яких виконує свою частину обов'язків. Так актори адміністратор, експерт та гість працюють лише з клієнтською частиною. Гість має доступ до обмеженого функціоналу, йому доступна можливість перегляду інформації по точкам та по забрудненням, він може порівнювати об'єкти на мапі між собою. Гість може авторизуватися в системі, після чого буде визначений як експерт або як адміністратор.

Експерти мають розширені функціональні можливості, серед них: додавання об'єктів на мапу, редагування існуючих об'єктів на мапі. Експерти мають більше можливостей у системі, але весь функціонал для них є недоступним.

Адміністратор має доступ для всього функціоналу, що був перелічений вище, але для додатково він може працювати з довідкою, завдяки чому частково

редагувати інформацію у базі даних. За допомогою довідки Адміністратор може додавати, редагувати, видаляти або просто шукати дані. Довідник доступний лише йому, інші користувачі не мають до нього доступу. Для повноцінної роботи веб-орієнтованої системи, клієнтська частина повинна отримувати інформацію від серверної частини, який в свою чергу виконує запити до бази даних.

Більш детально весь процес взаємодії описує схема клієнт-серверної архітектури додатку, що зображена нижче (рисунк 3.8) [10].

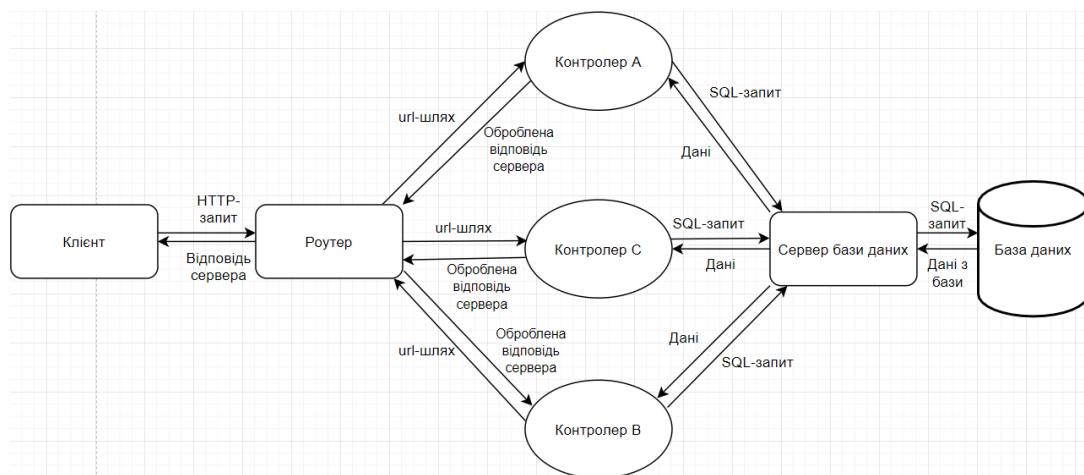


Рисунок 3.8 — Схема клієнт-серверного застосунку

Серверна частина відповідає за обробку запитів клієнта, комунікацію з реляційною базою даних, реалізацію функцій для роботи з даними: вставка даних, їх редагування та вибірка, а також повертання оброблених даних до клієнта. Схема серверної частини зображена на рисунку 3.9.

У свою чергу її головними складовими є конфігурація для роботи з базою даних, список API HTTP-шляхів для взаємодії з клієнтською частиною, контролери й сервіси, як ланка взаємодії з БД [10].

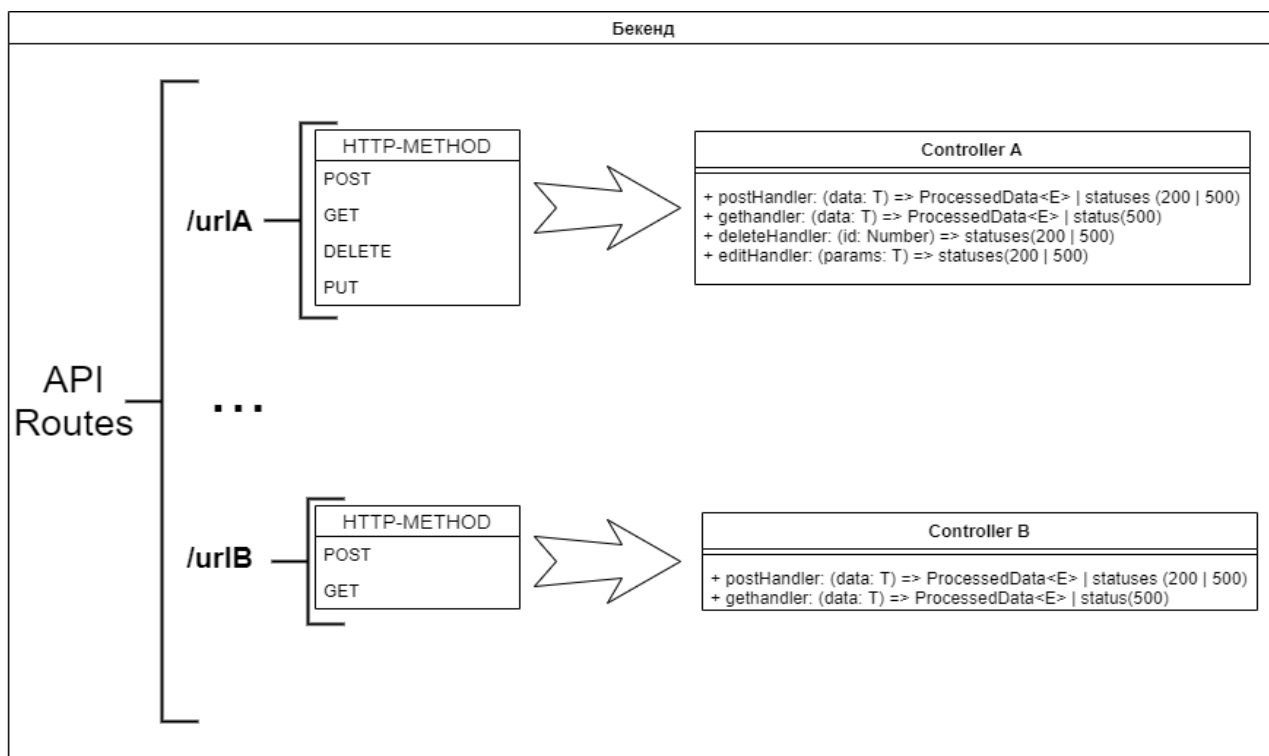


Рисунок 3.9 — Схема серверної частини

Конфігурація містить у собі інформацію, що необхідна для з'єднання з базою даних, а саме адресу бази даних, ім'я користувача, пароль та назву схеми. Додатково можна задавати порт та інші параметри з'єднання. У даній системі конфігурація знаходиться у файлі `mysql-config.js` і має наступний вигляд (рисунок 3.10).

```

1  const mysql = require('mysql');
2  const pool = mysql.createPool({
3    multipleStatements: true,
4    host: 'localhost',
5    user: 'USER_NAME',
6    password: 'PASSWORD_NAME',
7    database: 'DATABASE_NAME',
8    port: '3306',
9  });
10
11 module.exports = pool;
12
```

Рисунок 3.10 — Структура конфігурації для з'єднання з базою даних

Такий спосіб задання дозволяє швидко змінювати підключення. Іноді під час розробки виникала потреба змінити з'єднання з віддаленої бази даних на локальний сервер. Все, що для цього було необхідно зробити — вписати відповідні параметри у

цьому файлі, після чого вся система починала працювати з локальною базою даних.

У таблиці 3.1 наведено пояснення щодо полів даної конфігурації.

Таблиця 3.1 Параметри конфігурації для з'єднання

Назва поля	Пояснення
host	Вказується хост бази даних. Може бути як localhost, якщо сервер бази даних працює локально, так і IP адреса віддаленого серверу бази даних.
user	Ім'я користувача для входу
password	Пароль для входу
database	Назва необхідної бази даних

До даного об'єкту конфігурацій можна включати й додаткові параметри: обмеження за кількістю з'єднань, можливість незахищеного доступу та інші.

Список API HTTP-шляхів для взаємодії із клієнтською частиною має наступну структуру:

`router.<HTTP-method name>(<path>, <middleware?>, <controller handler function>)`

Приклад такого зв'язку HTTP шляху та контролера зображено на рисунку 3.11

```
router.post('/polygon', polygonsController.addPolygon);
router.get('/polygon/:id', polygonsController.getPolygon);
router.put('/polygon/:id', polygonsController.updatePolygon);

router.get('/regions', regionsController.getRegions);

router.get('/points', pointsController.getPoints);
router.get('/advancedpoints', pointsController.getAdvancedPoints);
```

Рисунок 3.11 — Приклад зв'язку HTTP шляху та контролера

Контролери взаємодіють із базою даних та підготовлюють дані для надсилання до клієнтської частини. Кожен із контролерів має відповідні методи, які пов'язують HTTP шлях API веб-системи з функціоналом, що взаємодіє безпосередньо з базою даних та обробляє результати.

Контролер являє собою сутність, що містить у собі семантично закріплену

абстракцію. З точки зору реалізації - це файл, всередині якого є функції, що приймають вхідні параметри, опрацьовують їх і повертають результат. Приклад контролера наведено на рисунку 3.12.

```
const getPoints = (req, res) => {
  const { idEnvironment, startDate, endDate } = req.query;

  const query = `
SELECT DISTINCT
  poi.Id,
  poi.id_of_user,
  poi.Coord_Lat,
  poi.Coord_Lng,
  poi.Description,
  poi.Name_object,
  poi.Type,
  type_of_object.id as Object_Type_Id,
  type_of_object.Name as Object_Type_Name,
  owner_type as owner_type_id,
  owner_types.type as owner_type_name,
  user.id_of_expert
FROM poi
INNER JOIN type_of_object ON poi.Type = type_of_object.id
INNER JOIN owner_types ON poi.owner_type = owner_types.id
INNER JOIN user ON poi.id_of_user = user.id_of_user
INNER JOIN emissions_on_map ON poi.Id = emissions_on_map.idPoi
INNER JOIN environment ON (environment.id = emissions_on_map.idEnvironment OR environment.AttachEnv = emissions_on_map.idEnvironment)
WHERE environment.id = ${idEnvironment} OR environment.AttachEnv = ${idEnvironment}`;

  const pointsPromise = new Promise((resolve, reject) => {
    pool.query(query, (error, rows) => {
      if (error) {
        console.log(error);
        reject(error);
      }

      resolve(rows);
    });
  });

  return Promise.all([pointsPromise, iconsMapPromise])
    .then(([points, iconsMap]) => { ...
    })
    .catch((error) => {
      res.status(500).send({
        message: error,
      });
    });
  });
};
```

Рисунок 3.12 — Приклад контролера для отримання точок на мапі

Додатково контролер може використовувати сервіси для своєї роботи. Приклад сервісу показано на рисунку 3.13.

```

class CalculationService {
  async generateNewCalculationNumber() {
    return new Promise((resolve, reject) => {
      const tableName = 'calculations_description';
      const query = `
        SELECT
          MAX(??) as maxId
        FROM ??
      `;
      pool.query(query, ['calculation_number', tableName], (error, rows) => {
        if (error) {
          reject(error);
        }
        resolve(rows[0].maxId + 1);
      });
    });
  }

  async setCalculationNumberToPoi({ idPoi, calculationNumber }) {
    return new Promise((resolve, reject) => {
      const tableName = 'radiation_poi_calculation';
      const query = `
        INSERT INTO
          ??
          (??)
        VALUES
          (??)`;

      const columnNames = ['calculation_number', 'idPoi'];
      const values = [calculationNumber, idPoi];
      const parameters = [tableName, columnNames, values];
      pool.query(query, parameters, (error, rows) => {
        if (error) {
          reject(error);
        }
        if (rows) {
          resolve();
        }
      });
    });
  }
}

```

Рисунок 3.13 — Фрагмент сервісу

Сервіс – це клас, що призначений для підготовки та обробки даних.

Перелік базового функціоналу сервісу:

- обробка даних;
- підготовка даних;
- виконання SQL-запитів та інших запитів до бази даних;
- взаємодія з іншими сервісами.

У сервісах часто знаходяться функції перетворення даних, викликів додаткових методів, що потрібні для опрацювання вхідних і вихідних даних, запитів до БД.

У функціях контролера чи сервісу формуються відповідні SQL-запити до бази даних. Ці запити можуть формуватися динамічно (рисунок 3.14).

```

const getEmissionsOnMap = (source, id, idEnvironment, limit) => {
  const filteringColumnName = source === SOURCE_POI
    ? 'idPoi'
    : source === SOURCE_POLYGON
    ? 'idPolygon'
    : undefined;
  const idEnvironmentClause = idEnvironment && Array.isArray(idEnvironment) && idEnvironment.length > 0
    ? `AND (environment.id IN (${idEnvironment}) OR environment.AttachEnv IN (${idEnvironment}))`
    : idEnvironment
    ? `AND (environment.id = ${idEnvironment} OR environment.AttachEnv = ${idEnvironment})`
    : '';

  return new Promise((resolve, reject) => {
    const columnNames = [
      'idElement',
      'idEnvironment',
      'ValueAvg',
      'ValueMax',
      'Year',
      'Month',
      'day',
      'emissions_on_map.Measure',
      'elements.short_name',
      'environment.name',
      'gdk.mpc_avrg_d',
      'gdk.mpc_m_ot'
    ];
    const query = `
    SELECT ??
    FROM ??
    INNER JOIN elements ON elements.code = emissions_on_map.idElement
    INNER JOIN environment ON (environment.id = emissions_on_map.idEnvironment OR environment.AttachEnv = emissions_on_map.idEnvironment)
    LEFT JOIN gdk ON gdk.code = emissions_on_map.idElement
    WHERE ?? = ?
    ${idEnvironmentClause}
    ORDER BY emissions_on_map.Year DESC, emissions_on_map.Month DESC, emissions_on_map.day DESC ${limit ? 'LIMIT ' + limit : ''}`;
  });
}

```

Рисунок 3.14 — Приклад формування динамічного запиту

У випадку набору параметрів, SQL запити можуть формуватися параметрично (рисунок 3.15).

```

const polygonPromise = new Promise((resolve, reject) => {
  const tableName = 'poligon';
  const updatedValues = {
    brush_color_r,
    brush_color_g,
    brush_color_b,
    brush_alfa,
    line_color_r,
    line_color_g,
    line_color_b,
    line_alfa,
    line_thickness,
    name,
    description,
  };

  const query = `
  UPDATE ??
  SET ?
  WHERE ?? = ?
  `;

  const values = [tableName, updatedValues, 'Id_of_poligon', id];

  pool.query(query, values, (error, rows) => {
    if (error) {
      reject(error);
    }

    if (rows) {
      resolve();
    }
  });
});

```

Рисунок 3.15 — Приклад формування запиту з параметрами

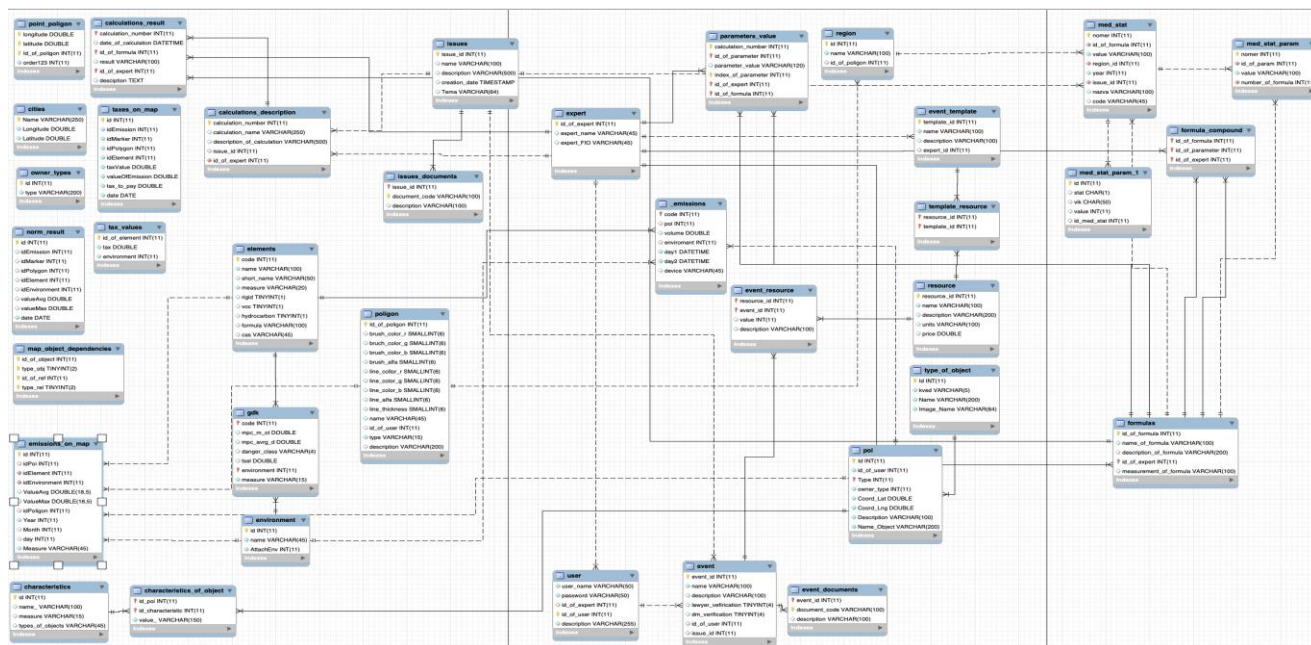
Конструкції «??» та «?» використовуються для задання параметрів. Символи «??» застосовуються для виділення значень лапками, тобто як рядок, а «?» — певної обробки вхідних даних.

Використовуючи символ «?» для різних типів вхідних даних, буде

- для чисел (Number) формат зберігається;
- для логічних значень (Boolean) формат конвертується у true / false;
- для рядків (String) значення обертається у лапки;
- масиви (Array) конвертуються у список значень. Масив ['a', 'b'] перетвориться у 'a','b';
- вкладені масиви (Array<Array<T>>) групуються у списки. Приклад [['a', 'b'], ['c', 'd']] перетвориться у ('a', 'b'), ('c', 'd');

— null / undefined перетворюються у NULL.

У такий спосіб, можна легко конфігурувати SQL-запити, додаючи або видаляючи поля з JavaScript об'єктів та масивів. Ці запити відправляються до віддаленого сервера бази даних, після чого результат обробляється і відправляється користувачу.



правильно спроектирована база данных виключас надли

забезпечує швидкодію та відповідність потоку даних.

Як видно з рисунків, таблиці мають деякі зв'язки, що зображені лініями. Ці зв'язки створені для того щоб пов'язувати між собою таблиці, таким чином дозволяючи збирати об'єми даних, що знаходяться в різних таблицях. Зв'язки бувають трьох типів: один до одного (1:1), один до багатьох (1:N), багато до багатьох (N:N). Зв'язок багато до багатьох формується через пару зв'язків один до багатьох. Це дозволяє задавати відповідність полів однієї таблиці до іншої. Приклад з'єднання один до багатьох зображено на рисунку 3.17, де одній ролі експерта може відповідати декілька користувачів.

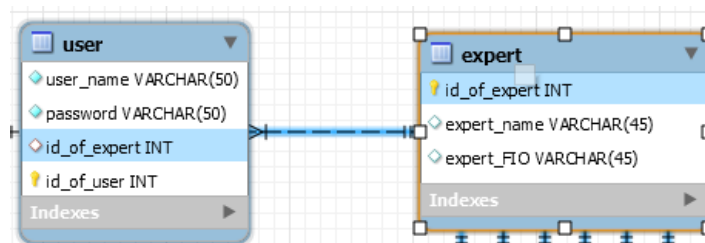


Рисунок 3.17 — Приклад зв'язку один до багатьох (1:N)

На рисунку вище видно, що деякі поля відмічені ключем. Поняття «ключа» у базах даних широко відоме. Ключ — це мінімальний набір атрибутів по значенням яких можна однозначно обрати потрібний екземпляр сутності. Тут мінімальність вказується для того щоб підкреслити, що усунення будь якого елементу з цього набору вже не дозволить ідентифікувати сутність. Такі ключі розділяють на різні категорії, проте однією з основних є обмеження цілісності даних до якої відносять поняття первинного (англ. primary) та зовнішнього (англ. foreign) ключа.

Первинний слугує для обмеження цілісності у рамках однієї таблиці для однозначної ідентифікації сутності. Первинний ключ не може бути пустим або повторюватись.

Зовнішній ключ - це обмеження цілісності зв'язків декількох таблиць. Що дозволяє будувати цілісні моделі даних, оскільки дочірня таблиця не може мати посилання на неіснуючі дані батьківської таблиці.

3.4 Висновки до розділу

У даному розділі було розглянуто веб-орієнтовану систему комплексного еколого-економіко-енергетичного моніторингу. Оскільки система надає автоматизовані робочі місця (АРМ) експертів, було наведено принципи та порядок роботи системи з описом дій та ролей кожного експерта. Для візуалізації поведінки системи та її подій було створено діаграму прецедентів, де описано можливості користувачів системи.

Для досягнення ефективної роботи веб-системи було детально розглянуто всі переваги обраних інструментів, описано взаємодію їх між собою задля забезпечення швидкої роботи на багатьох платформах. Обрані технології відповідають сучасним вимогам до розробки й дозволяють легко масштабувати та підтримувати програмний продукт.

Оскільки застосунок є веб-орієнтованим, то за основу було взято клієнт-серверну архітектуру, опис якої наведений на схемі інформаційних потоків.

Серверна частина напряму взаємодіє з базою даних, тому для розуміння побудови зв'язку між ними було описано такі структурні елементи, як контролери та сервіси.

Підключення до бази даних виконується за допомогою конфігурації з мінімально необхідним набором параметрів, що дозволяє швидко перемикатися між різними базами даних (наприклад для порівняння результатів оптимізації між локальною та віддаленою базою).

Конструкції задання параметрів у сервісах надають зручний спосіб побудови SQL-запитів, тому було детально описано перетворення, які вони виконують над різними типами вхідних даних.

Реляційна база даних КЕЕЕМ складається з 35 таблиць, кожна з яких має первинний ключ та з'єднана з іншими таблицями, що допомагає робити вибірку даних з застосуванням відношень. У системі використовуються всі три типи відношень: один до одного, один до багатьох та багато до багатьох. Детально всю структуру та таблиці бази даних зображено на ER-діаграмі з позначенням первинних ключів та зв'язків між іншими таблицями.

4 ОПИС РОБОТИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Веб-орієнтована система не потребує окремого встановлення на обладнання кінцевого користувача, достатньо лише мати доступ до мережі Інтернет та один із браузерів, наприклад Google Chrome, Opera чи Mozilla Firefox, які підтримуються значною більшістю сучасних гаджетів. Взаємодія з базою даних відбувається шляхом надсилання HTTP запитів до API серверу, що виникають внаслідок дій користувача з системою.

4.1 Інсталяція програмного забезпечення

Для використання веб-орієнтованої системи потрібно встановити браузер. У випадку з браузером Google Chrome для цього потрібно зайти на офіційну сторінку веб-переглядача за посиланням <https://www.google.com/intl/ru/chrome/> та завантажити останню версію. Після успішного завантаження необхідно запустити виконаний файл інсталятора та виконати кроки згідно інструкцій, які надає майстер встановлення.

Якщо всі кроки виконано правильно, у автоматично відкритому після встановлення вікні браузера необхідно в адресній стрічці вказати адресу на веб-орієнтовану систему KEEEM <https://keem.com.ua/> та перейти за посиланням.

4.2 Вимоги до обчислювальної техніки

Оскільки для відкриття сторінки з веб-орієнтованою системою необхідний браузер, то вимоги до обчислювальної техніки обмежуються наявністю доступу до мережі Інтернет та вимогами браузера до обчислювальної техніки користувача.

Для комп'ютерів, що працюють на сімействі операційних систем Windows:

- ОС Windows 7, Windows 8, Windows 8.1, Windows 10 чи більш пізніші версії;
- центральний процесор Intel Pentium 4 чи більш пізнішої версії з підтримкою SSE3.

Для комп'ютерів, працюючих на операційній системі MacOS необхідно мати як

мінімум MacOS High Sierra 10.13 чи більш пізнішу її версію.

Для комп'ютерів з операційними системами на базі ядра Linux необхідно мати:

— ОС Ubuntu 18.04 (64-бітна версія) чи більш пізніші версії, Debian 10 чи більш пізніші версії, openSUSE 15.2 чи більш пізніші версії, Fedora Linux 32 чи більш пізніші версії;

— центральний процесор Intel Pentium 4 чи більш пізнішої версії з підтримкою SSE3.

Також Google Chrome підтримується сучасними мобільними телефонами, так званими смартфонами, тому з системою KEEEM можна працювати, якщо смартфон користувача має операційну систему Android 7.0 Nougat чи більш пізніші версії.

4.3 Демонстрація та представлення сценаріїв програмної системи

Веб-орієнтована система KEEEM складається з набору веб-сторінок, між якими користувач може вільно перемикається.

4.3.1 Головна сторінка

Структура сторінок складається з двох частин, навігаційне меню (рисунок 4.1) та змістовна частина, яка в свою чергу може поділятися на під частини. Навігаційне меню складається з розміщених на ньому посилань: посилання на головну сторінку, довідник (доступний лише Адміністратору), карта викидів та кнопка входу до системи, де міститься призначена для експертів інформація, що міститься в базі даних.

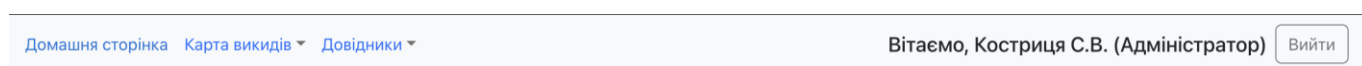


Рисунок 4.1 — Навігаційне меню веб-орієнтованої системи KEEEM

На головній сторінці змістовна частина (рисунок 4.2) складається з каруселі зображень та опису основних принципів еколого-економіко-енергетичного моніторингу, які описують головне призначення системи.

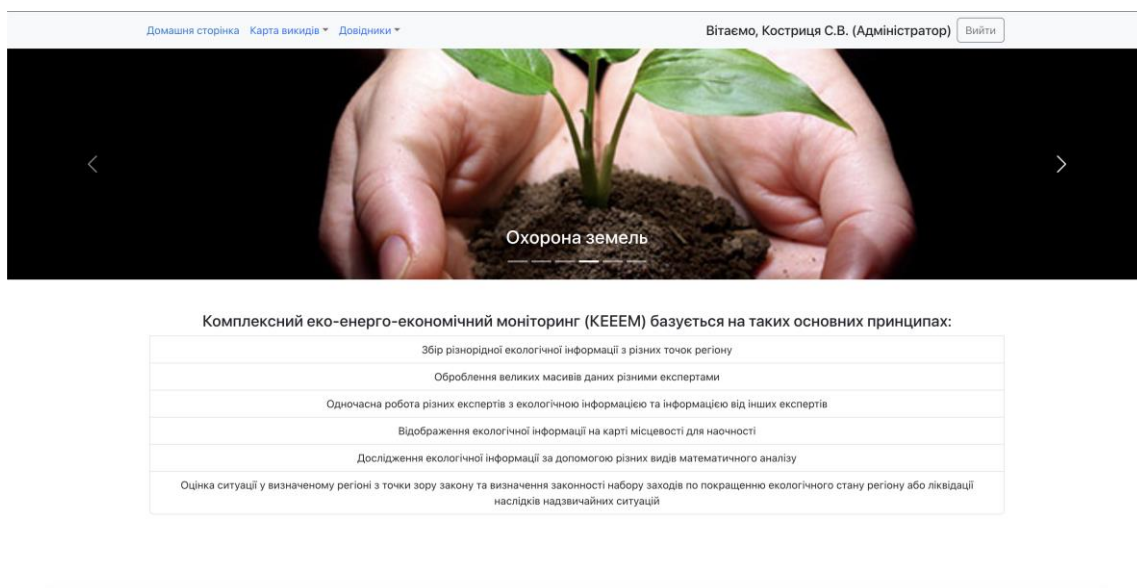


Рисунок 4.2 — Змістовна частина головної сторінки

При натисканні на пункт навігаційного меню «Карта викидів» користувачеві відобразиться список середовищ (рисунок 4.3) за якими він може переглядати інформацію на інтерактивній мапі.

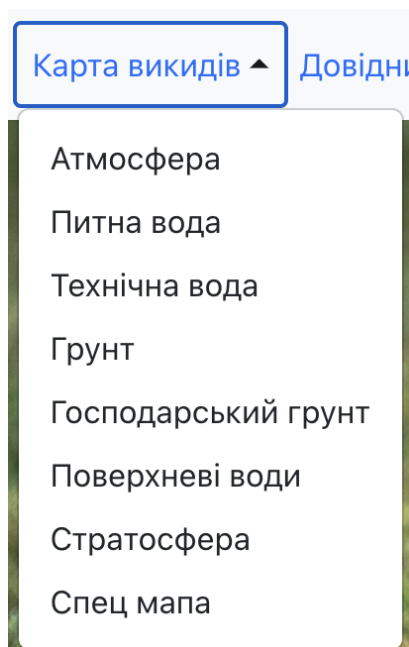
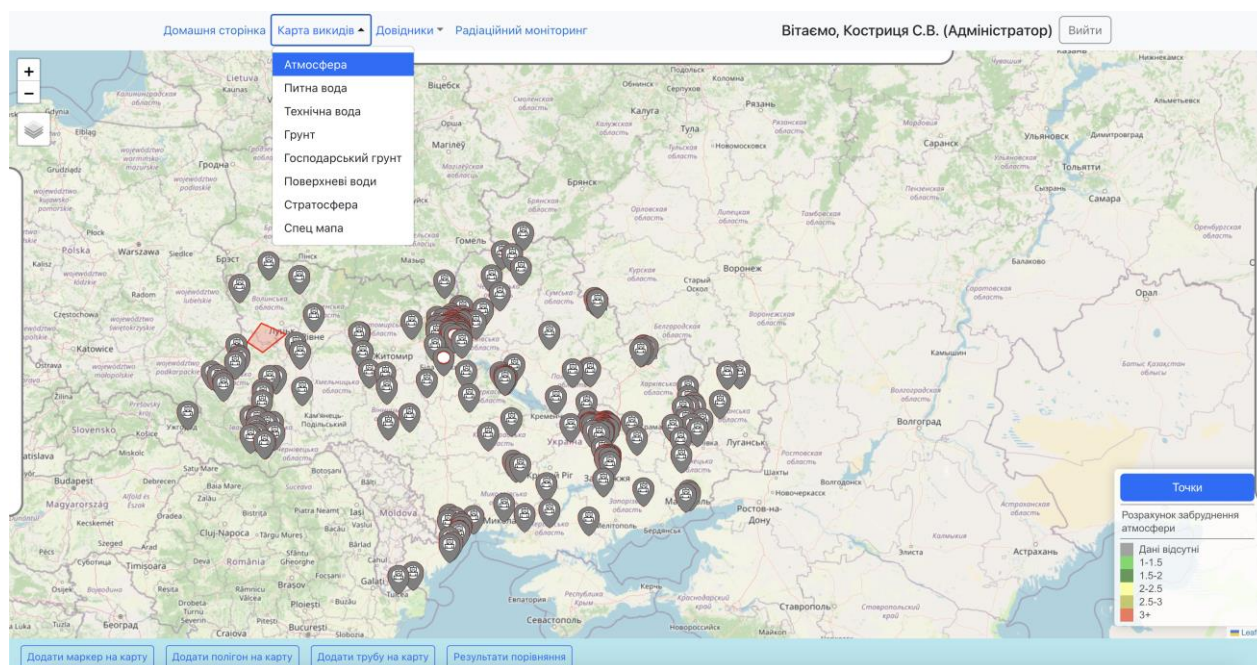


Рисунок 4.3 — Доступні для вибору екологічні середовища

Натиснувши на один із елементів списку система перенаправить користувача на сторінку з інтерактивною мапою, де йому буде надана інформація згідно обраного середовища.

4.3.2 Карта викидів

Змістовна частина сторінки з картою викидів складається з інтерактивної мапи та панелі керування внизу, додатково присутні бокові панелі, які відкриваються при тільки натисканні і містять окремий функціонал (рисунк 4.4).



Рисунк 4.4 — Змістовна частина сторінки карти викидів

На мапі користувачеві відображається велика кількість компонентів з якими він може взаємодіяти, серед них:

- інтерактивна мапа та її компоненти;
- меню навігації;
- меню фільтрації;
- панель інструментів.

Під час роботи з мапою можна побачити такі компоненти:

- POI маркери;
- полігони;
- трубопроводи;
- області.

Кожен з цих компонентів є інтерактивним і нього можна натиснути, де буде відображена загальна інформація по об'єкт у вигляді вікна підказки (рисунк 4.5).

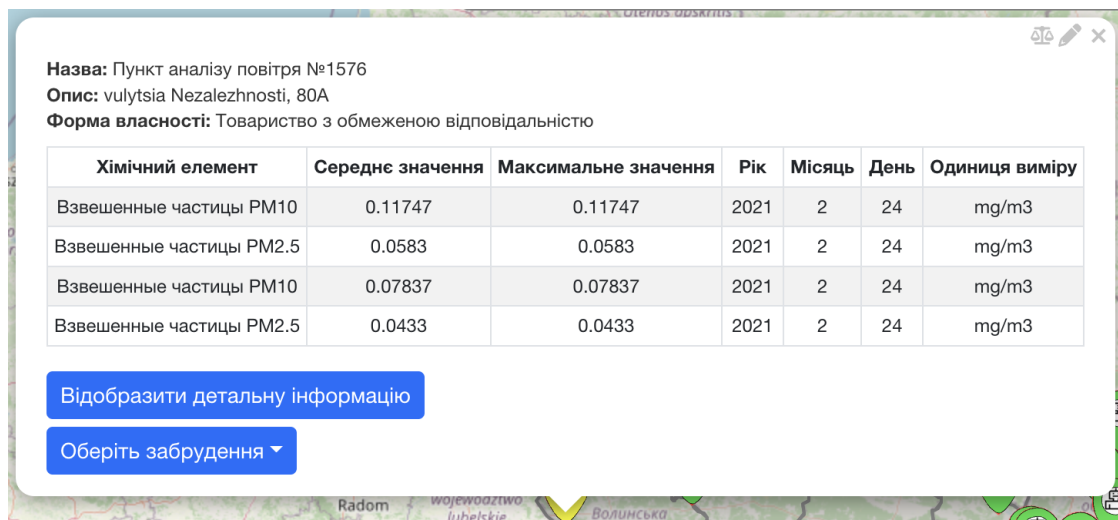


Рисунок 4.5 — Вікно підказки POI-маркера

У цьому вікні користувач може вибрати даних об'єкт для порівняння з іншими, відобразити його детальну інформацію та переглянути забруднення конкретного типу.

4.3.3 Довідники

Сторінка довідки доступна лише Адміністратору і використовується для редагування даних в базі. Оскільки зміни в базі можуть вплинути на всю систему – ні гість, ні інший експерт який не має прав Адміністратора не зможе виконати цих дій, оскільки даний пункт у нього не буде відображений. Після натискання на пункт «Довідники» навігаційного меню, відобразиться список наявних для редагування довідників (рисунок 4.6).

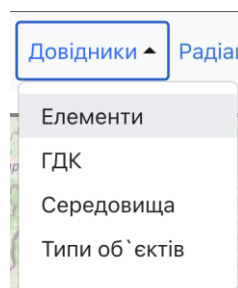


Рисунок 4.6 — Список довідників для редагування

Довідники використовуються Адміністратором для перегляду інформації в базі даних, її додавання, редагування та видалення через інтерфейс (рисунок 4.7).

4.4 Експериментальні дослідження параметрів функціонування бази даних KEEEM

Перш ніж виконувати аналіз та оптимізацію запитів, необхідно перевірити налаштування серверу, зокрема впевнитись, що увімкнене кешування запитів.

Для перевірки скористаємось командою `SHOW VARIABLES LIKE '%query_cache%'`.

У результаті бачимо, що значення змінної `have_query_cache` – YES, тобто кешування увімкнене (рисунок 4.9).

Variable_name	Value
have_query_cache	YES
query_cache_limit	262144
query_cache_min_res_unit	4096
query_cache_size	33554432
query_cache_type	ON
query_cache_wlock_invalidate	OFF

Рисунок 4.9 – Результат команди `SHOW VARIABLES LIKE '%query_cache%'`

Звернемо також увагу на значення змінної `query_cache_type`, що описує тип кешування, що використовується. У даному випадку значення ON показує, що кешування увімкнено для усіх запитів, за виключенням операторів SELECT з опцією SQL_NO_CACHE, DEMAND (дозволяє активувати кешування запитів за вимогою у разі використання операторів SELECT з опцією SQL_CACHE).

Для даної бази даних такі налаштування за замовчуванням влаштовують, тому вносити зміни у конфігурацію не потрібно. Розмір кешу влаштовує, його надмірне збільшення може створити зайве навантаження.

Виконаємо експериментальне дослідження сформованого набору рекомендацій з оптимізації бази даних на прикладі оптимізації запитів різними шляхами.

4.4.1 Запит 1. Відображення маркерів на інтерактивній мапі

Для відображення точок інтересу певного середовища на інтерактивній мапі необхідно реалізувати запит, що дозволить оглянути інформацію про точку (опис, ім'я об'єкту, тип) та інформацію про користувача, який її виставив.

Для цього напишемо такий запит:

```
SELECT DISTINCT
```

```
    poi.Id,
```

```
    poi.id_of_user,
```

```
    poi.Coord_Lat,
```

```
    poi.Coord_Lng,
```

```
    poi.Description,
```

```
    poi.Name_object,
```

```
    poi.Type,
```

```
    type_of_object.id as Object_Type_Id,
```

```
    type_of_object.Name as Object_Type_Name,
```

```
    owner_type as owner_type_id,
```

```
    owner_types.type as owner_type_name,
```

```
    user.id_of_expert
```

```
FROM poi
```

```
INNER JOIN type_of_object ON poi.Type = type_of_object.id
```

```
INNER JOIN owner_types ON poi.owner_type = owner_types.id
```

```
INNER JOIN user ON poi.id_of_user = user.id_of_user
```

```
INNER JOIN emissions_on_map ON poi.Id = emissions_on_map.idPoi
```

```
INNER JOIN environment ON (environment.id =
```

```
emissions_on_map.idEnvironment OR environment.AttachEnv =
```

```
emissions_on_map.idEnvironment)
```

```
WHERE environment.id = 1 OR environment.AttachEnv = 1;
```

Фрагмент результату виконання запиту (рисунок 4.10):

Id	id_of_user	Coord_Lat	Coord_Lng	Description	Name_object	Type	Object_Type_Id	Object_Type_Name	owner_type_...	owner_type_name	id_of_expert
1061	2	48.408944	35.072505	Geroev Avenue, 40	SAVEDNIPRO_001	273	273	Точка збору інформації	4	Товариство з обмеженою відповідальністю	0
1067	2	48.46179	35.022893	Ulyanova Street, 26	SAVEDNIPRO_003	273	273	Точка збору інформації	4	Товариство з обмеженою відповідальністю	0
1064	2	48.405856	35.145451	Pryberezha Street, 49	SAVEDNIPRO_006	273	273	Точка збору інформації	4	Товариство з обмеженою відповідальністю	0
1068	2	48.418962	35.009254	Pershogo Travnaya Street, 7	SAVEDNIPRO_002	273	273	Точка збору інформації	4	Товариство з обмеженою відповідальністю	0
1066	2	48.428476	35.010238	Novokrymskaya Street, 1	SAVEDNIPRO_004	273	273	Точка збору інформації	4	Товариство з обмеженою відповідальністю	0
1070	2	48.467467	35.041472	Dmytra Yavornytskoho Avenue, 60	SAVEDNIPRO_007	273	273	Точка збору інформації	4	Товариство з обмеженою відповідальністю	0
1062	2	48.475959	35.016979	Vokzalna Square, 2	SAVEDNIPRO_014	273	273	Точка збору інформації	4	Товариство з обмеженою відповідальністю	0
1069	2	48.44803	35.058606	Haharina Avenue, 43	SAVEDNIPRO_008	273	273	Точка збору інформації	4	Товариство з обмеженою відповідальністю	0
1065	2	50.3873111	30.4713752	Mixalina Ломоносова вулиця, 73	SAVEDNIPRO_010	273	273	Точка збору інформації	4	Товариство з обмеженою відповідальністю	0
1063	2	48.513536	35.062787	Kalynova Street, 49	SAVEDNIPRO_009	273	273	Точка збору інформації	4	Товариство з обмеженою відповідальністю	0
1071	2	48.467926	35.055617	Pavla Nirinberha Street, 4-6	SAVEDNIPRO_015	273	273	Точка збору інформації	4	Товариство з обмеженою відповідальністю	0
1075	2	48.484697	35.070445	Liubarskoho Street, 36	SAVEDNIPRO_017	273	273	Точка збору інформації	4	Товариство з обмеженою відповідальністю	0

Рисунок 4.10 – Результат виконання запиту 1

У цьому запиті присутні об'єднання INNER JOIN. Запит SELECT буде виконуватись шляхом розкриття всіх таблиць, що беруть участь у запиті та їх подальшому з'єднанню по JOIN та умові WHERE [14]. Під час з'єднання будуть створюватися нові тимчасові таблиці, що будуть задовольняти усім умовам. За перерахованими у запиті умовами буде створено проміжну таблицю довжиною у 306 записів та шириною у 8 (для таблиці poi) + 4 (для таблиці type_of_object) + 2 (для таблиці owner_types) + 5 (для таблиці user) + 11 (для таблиці emissions_on_map) + 3 (для таблиці environment) = 33 стовпці, після чого з неї будуть обиратися необхідні 12 стовпців: *poi.Id*, *poi.id_of_user*, *poi.Coord_Lat*, *poi.Coord_Lng*, *poi.Description*, *poi.Name_object*, *poi.Type*, *type_of_object.id*, *type_of_object.Name*, *owner_types.id*, *owner_types.type*, *user.id_of_expert*.

Якщо розмір створеної під час виконання тимчасової таблиці буде меншим, аніж значення змінної *tmp_table_size* (тобто, максимального розміру внутрішніх тимчасових таблиць в пам'яті [14]), то запит спрацює достатньо швидко. У протилежному випадку створення цієї таблиці відбуватиметься на диску, що є досить повільною операцією і може завантажити дискову систему [14].

Вирішенням цієї проблеми може бути виділення потрібних для вибірки таблиць і винесення їх в окрему таблицю, після чого виконати об'єднання таблиць. Для початку визначимо початкову швидкість виконання запиту (рисунок 4.11).

Timing (as measured at client side):
Execution time: 0:00:0.25545692

Timing (as measured by the server):
Execution time: 0:00:0.25554000
Table lock wait time: 0:00:0.00002800

Рисунок 4.11 – Швидкість виконання Запиту 1

Отримали результат у 0.255 сек. для 306 записів, 0.170 сек. для 300 записів, 0.0015 сек. для 200 записів та 0.0011 сек. для 100 записів.

План виконання запиту чітко описує всі виконані операції та зазначений нижче (рисунок 4.12).

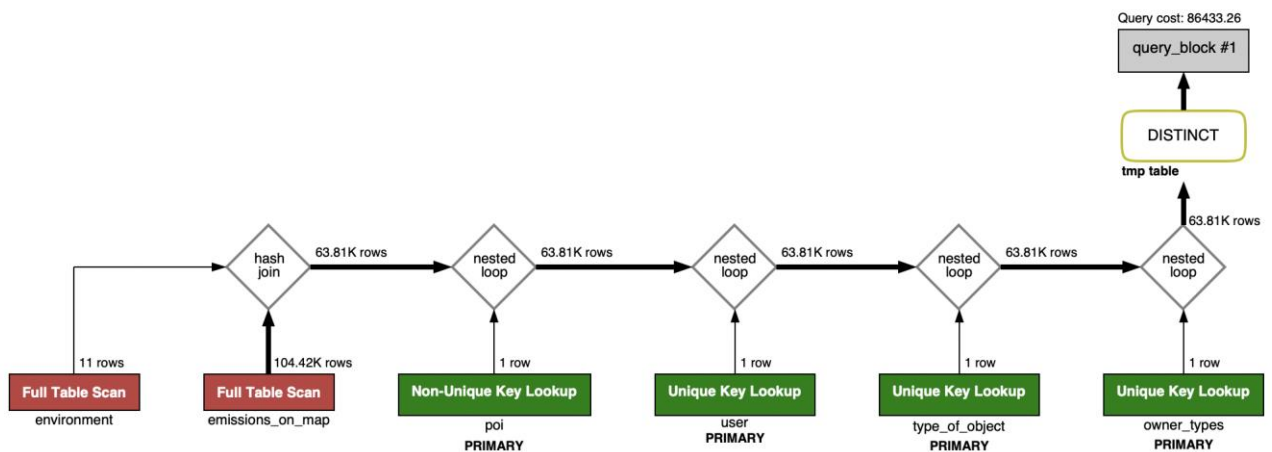


Рисунок 4.12 – План виконання Запиту 1

Виконаємо аналіз швидкості даного запиту, а також передбачення для більших об'ємів даних у таблиці (таблиця 4.1).

Таблиця 4.1 Аналіз продуктивності за часом виконання запиту

Об'єм вибірки	100	200	300
Час виконання, сек	0.0011	0.0015	0.170

Тепер виконаємо передбачення швидкості виконання запиту для більших об'ємів таблиці. Для початку необхідно виконати інтерполяцію, щоб, знайшовши проміжні значення за отриманими експериментальним шляхом дискретними значеннями, зрозуміти характер зміни наших даних (рисунок 4.13).

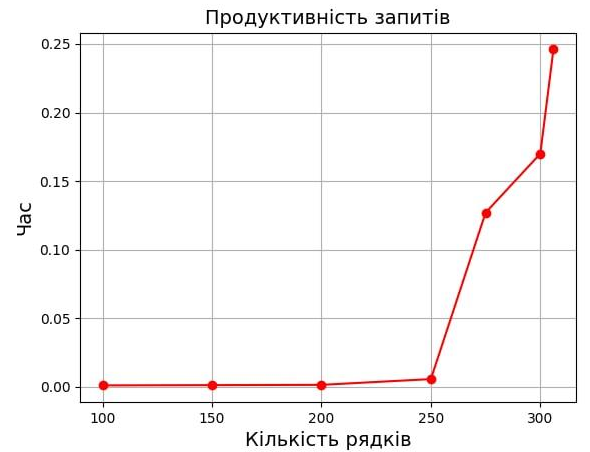


Рисунок 4.13 – Графік інтерполяції за даними з табл. 4.1

Далі необхідно виконати лінійну екстраполяцію для знаходження майбутніх значень (рисунок 4.14).

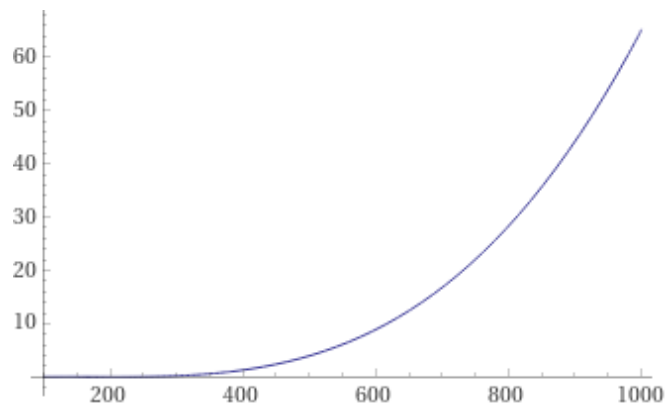


Рисунок 4.14 – Графік лінійної екстраполяції

Виходячи з графіку на рисунку 4.14, а також з того, що передбачення буде будуватись для великих значень, проаналізуємо продуктивність за часом виконання (таблиця 4.2).

Таблиця 4.2 Аналіз продуктивності за часом виконання запиту

Об'єм вибірки	10000	30000	60000	1000000
Час виконання, сек	2.3402	4.4739	9.6559	17.9339

Виконаємо створення тимчасових та перевіримо роботу запиту:

```
CREATE TEMPORARY TABLE tmp_type_of_object (PRIMARY KEY (id)) SELECT id,
Name FROM type_of_object;
```

```
CREATE TEMPORARY TABLE tmp_user (PRIMARY KEY (id_of_user)) SELECT
id_of_user, id_of_expert FROM user;
```

```
CREATE TEMPORARY TABLE tmp_emissions_on_map (PRIMARY KEY (id)) SELECT
emissions_on_map.id as id, idPoi FROM emissions_on_map WHERE idEnvironment = 1;
```

```
SELECT DISTINCT
```

```
poi.Id, poi.id_of_user, poi.Coord_Lat, poi.Coord_Lng, poi.Description, poi.Name_object,
poi.Type, type_of_object.id as Object_Type_Id, type_of_object.Name as
Object_Type_Name, owner_types.id as owner_type_id, owner_types.type as
owner_type_name, user.id_of_expert
```

```
FROM poi
```

```
INNER JOIN owner_types ON poi.owner_type = owner_types.id
```

```
INNER JOIN tmp_type_of_object as type_of_object ON poi.Type = type_of_object.id
```

```
INNER JOIN tmp_user as user ON poi.id_of_user = user.id_of_user
```

```
INNER JOIN tmp_emissions_on_map as emissions_on_map ON poi.Id =
emissions_on_map.idPoi;
```

Тепер проміжні таблиці налічують лише $6 + 6 = 12$ стовбців у сумі, що майже втричі меншим, ніж при попередньому виконанні запиту. Отримуємо, що виконання запиту пришвидшується до 0.233 сек. (рисунок 4.15).

Timing (as measured at client side):

Execution time: 0:00:0.23294497

Timing (as measured by the server):

Execution time: 0:00:0.23299700

Table lock wait time: 0:00:0.00000400

Рисунок 4.15 – Виконання оптимізованого Запиту 1

Бачимо, що оптимізована версія запиту виконується дещо швидше, тому даний метод можна використовувати при написанні великої кількості об'єднань з іншими таблицями.

4.4.2 Запит 2. Отримання детальної інформації про місто

Для одержання інформації про міста, для яких використовується система моніторингу, необхідно вилучити із таблиці *cities* такі дані: *Name, Longitude, Latitude*.

Прикладом такого запиту слугує запит 2:

```
SELECT Name, Longitude, Latitude FROM cities;
```

Виконання цього запиту займає 0,012 сек. (рисунок 4.16)

Timing (as measured at client side):

Execution time: 0:00:0.00126982

Timing (as measured by the server):

Execution time: 0:00:0.00117400

Table lock wait time: 0:00:0.00001000

Рисунок 4.16 – Час виконання запиту 2

Проаналізуємо даний запит за допомогою команди *Explain*, результат виконання *Explain* для запиту 2 зображений на рисунку 4.17.

	id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
►	1	SIMPLE	cities	NULL	ALL	NULL	NULL	NULL	NULL	37	100.00	NULL

Рисунок 4.17 – Виконання Explain для неоптимізованого запиту 2

Після додавання складеного індексу по полях *longitude*, *latitude*, які було використано в умові запиту, швидкість запиту зменшилась до 0,00050 сек. Таке скорочення часу виконання для таблиці об'ємом у 37 записів є незначним, тому необхідність створення цих індексів викликає сумніви, оскільки індекси передбачають додаткові операції запису на диск, і під час кожного додавання даних у таблицю відбувається також запис та оновлення даних в індексі.

Новий результат виконання Explain для запиту 2 зображений на рисунку 4.18.

	id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
►	1	SIMPLE	cities	NULL	index	NULL	idx_cities_Longitude_Latitude	16	NULL	37	100.00	Using index

Рисунок 4.18 – Виконання Explain для запиту 2 (оптимізованого)

Виконаємо аналіз швидкості запиту 2 з передбаченням більших об'ємів даних у таблиці. Спочатку розглянемо випадки, коли індекси за полями, що використовуються у пошуку, не було створено (таблиця 4.3).

Таблиця 4.3 Аналіз продуктивності за часом виконання запиту без індексів

Об'єм вибірки	100	200	300	400	500	600	700	800	900	1000
Час виконання, сек	0.0014	0.0018	0.0029	0.0034	0.0045	0.0058	0.0067	0.0072	0.0084	0.0092

Виконаємо передбачення для більших об'ємів даних за допомогою екстраполяції. Використаємо лінійну екстраполяцію (таблиця 4.4).

Таблиця 4.4 Аналіз продуктивності за часом виконання запиту

Об'єм вибірки	10000	30000	60000	1000000
Час виконання, сек	0.7399	2.2799	4.7799	8.0011

Бачимо, що для 1000000 записів у таблиці *cities* швидкість запиту складає 8 секунд, оскільки відбуватиметься повне сканування таблиці. Тому у цьому випадку створення індексів є необхідним, оскільки повне сканування таблиці буде значно вповільнювати роботу запиту. Необхідно додати індекси за полями *longitude*, *latitude*. Перевіримо, як вплине додавання індексів на швидкість запиту (таблиця 4.5).

Таблиця 4.5 Аналіз продуктивності за часом виконання запиту з індексами

Об'єм вибірки	100	200	300	400	500	600	700	800	900	1000
Час виконання, сек	0.0005	0.0006	0.0007	0.00075	0.00089	0.001	0.0011	0.0012	0.0014	0.0018

Виконаємо екстраполяцію для передбачення на більших об'ємах даних (таблиця 4.6).

Таблиця 4.6 Аналіз продуктивності за часом виконання запиту

Об'єм вибірки	10000	30000	60000	1000000
Час виконання, сек	0.1	0.293	0.606	1.005

Помічаємо, що додавання індексів пришвидшує виконання запиту у 8 разів для таблиці розміром у 1000000 записів. Це суттєва різниця у продуктивності, тому починаючи з такого розміру таблиць створення індексів є необхідним.

4.4.3 Запит 3. Додання нового полігону

Після того, як користувач авторизується, він може додавати / редагувати трубопроводи на мапі. Трубопроводи представляються в системі як полігони та задаються точками на мапі. До параметрів полігону входять дані про лінію (колір,

прозорість, товщину), ім'я користувача, що його додав та опис. При додаванні нового полігону виконується пошук ідентифікатора останнього створеного полігону, значення якого потім збільшується на одиницю та використовується для нової вставки в таблицю. Очевидно, що це виконується у дві дії:

- 1) Пошук ідентифікатора останнього полігону.
- 2) Вставка нового полігону зі знайденим ідентифікатором + 1.

Для спрощення та пришвидшення такої операції вирішено було створити збережену процедуру, яка буде одразу виконувати обидві дії.

Процедура матиме вигляд:

BEGIN

```
INSERT INTO poligon(Id_of_poligon, line_collor_r, line_color_g, line_color_b,  
line_alfa, line_thickness, name, id_of_user, type, description)  
values((select max(Id_of_poligon) from poligon)+1, lineColorR, lineColorG,  
lineColorB, lineAlfa, lineThickness, Name, UserId, Type, Description);
```

END //

Після першого виклику цієї процедури її буде збережено на сервері у скомпільованому вигляді, і для наступних вставок користувачів вміст процедури буде швидше оброблятися сервером. Таким чином, замість послідовного виконання двох окремих запитів, клієнт буде посилатись на існуючу збережену процедуру, передаючи у неї лише необхідні дані, що полегшить виконання необхідних операцій та забезпечить більш безпечний доступ до таблиці бази даних.

4.5 Висновки до розділу

У даному розділі було детально розглянуто інсталяцію програмного забезпечення, наведено вимоги до обчислювальної техніки, продемонстровано сценарії роботи, з розподілом по модулям, які має веб-орієнтована система KEEEM.

Проведено детальне експериментальне дослідження параметрів функціонування бази даних KEEEM на прикладі оптимізації трьох запитів різними оптимальними шляхами, з наведеними результатами до та після оптимізації.

5 СТАРТАП-ПРОЄКТ

Розробка стартап-проєкту є гарною можливістю розповсюдити та покращити продукт за рахунок інвестування з боку зацікавлених користувачів. Внаслідок цього оптимізація баз даних стане значно легшим для виконання процесом і набуде статусу необхідності на етапі проєктування та розробки програмних систем з використанням баз даних великих обсягів.

5.1 Опис ідеї стартапу

У межах даного підпункту буде наведено аналіз та подано у вигляді таблиць наступне:

- зміст ідеї;
- можливі напрямки застосування;
- вигоди, які може отримати користувач товару;
- відмінні риси від наявних аналогів та напрямків.

Зміст ідеї, потенційні напрямки застосування та вигоди, що може отримати користувач товару представлені у вигляді таблиці 5.1.

Таблиця 5.1 Опис ідеї стартапу.

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Набір рекомендацій для оптимізації запитів до баз даних великих обсягів	Дослідження проблемних місць у роботі бази даних	Продукт є безкоштовним для використання
	Пришвидшення виконання запитів до бази даних	Немає потреби інсталиувати програму
	Збільшення продуктивності системи, залежної від БД	Незалежність від операційної системи та СКБД
		Продукт допомагає оптимізувати всі основні компоненти: сервер, структуру та запити до БД
		Мінімальні системні вимоги: доступ до бази даних та мережі Інтернет

На ринку існують готові рішення з оптимізації баз даних, проте більшість з них обмежені певною СКБД, розповсюджуються на платній основі та потребують інсталяції. Основною відмінністю розробленої системи є оптимізація всіх ланок роботи з БД, що дозволяє проводити аналіз всієї БД та виправлення конкретних проблемних місць у її роботі, у той час коли інші системи дозволяють оптимізувати запити до БД. У таблиці 5.2 наведено порівняльний аналіз визначення сильних, слабких та нейтральних характеристик ідеї проєкту.

Таблиця 5.2 Визначення сильних, слабких та нейтральних характеристик проєкту.

№ п/п	Техніко-економічні характеристики ідеї	Товари / концепції конкурентів			W	N	S
		Мій проєкт	EverSQL Query Optimizer	DbForge Studio			
1.	Автономність	Може працювати автономно чи з віддаленою базою	Після встановлення може працювати тільки автономно	Після встановлення може працювати тільки автономно			+
2.	Призначення	Оптимізація всіх аспектів роботи БД	Тільки оптимізація запитів до БД	Тільки оптимізація запитів до БД			+
3.	Багатоплатформність	Повністю не залежить від платформи та СКБД	ОС Windows, СКБД MySQL, MariaDB та PerconaDB	ОС Microsoft Windows, СКБД MySQL			+

5.2 Технологічний аудит ідеї проєкту

У цьому підрозділі буде наведено аналіз аудиту технологій, за допомогою яких можна реалізувати ідеї проєкту. Для визначення технологічної здійсненності ідеї проєкту потрібно проаналізувати такі складові:

- за якою технологією буде виготовлено товар згідно з ідеєю проєкту?
- чи існують такі технології, чи їх потрібно розробити/добробити?
- чи доступні такі технології авторам проєкту?

У таблиці 5.3 наведено технологічну здійсненність ідеї проєкту.

Таблиця 5.3 Технологічна здійсненність ідеї проєкту

№ п/п	Ідея проєкту	Технології її реалізації	Наявність технологій	Доступність технологій
1.	Еталонне тестування бази даних	Sysbench, Percona's TPC-C-MySQL Tool	Наявна	Доступні для вільного користування
2.	База даних	СКБД MySQL	Наявна	Умовно безкоштовна
3.	Алгоритми та скрипти оптимізації	Perl, Lua	Наявна	Доступні для вільного користування
Висновок: проєкт реалізувати можливо				

Обрані технології є доступними для вільного користування та надають можливості для реалізації поставленої задачі.

5.3 Аналіз ринкових можливостей запуску стартап-проєкту

Передбачення і визначення ринкових можливостей, які можна використати під час ринкової імплементації проєкту, та ринкових загроз, які можуть перешкодити реалізації проєкту, дає змогу спланувати напрями розвитку проєкту із урахуванням стану ринку, аналогічних пропозицій проєктів конкурентів.

Спочатку проводиться аналіз попиту: наявність попиту, обсяг, динаміка розвитку ринку (таблиця 5.4).

Таблиця 5.4 Характеристика потенційного ринку стартап-проєкту

№ п/п	Показники стану ринку (найменування)	Характеристика
1	Кількість головних гравців, од	3
2	Загальний обсяг продажів, грн/ум.од	20000 грн
3	Динаміка ринку (якісна оцінка)	Зростає
4	Наявність обмежень для входу (вказати характер обмежень)	Немає
5	Специфічні вимоги до стандартизації та сертифікації	Немає
6	Середня норма рентабельності в галузі (або по ринку), %	50

Середня норма рентабельності в галузі (або по ринку) порівнюється із банківським відсотком на вкладення. За умови, що останній є вищим, можливо, має сенс вкласти кошти в інший проєкт.

За результатами аналізу таблиці робиться висновок щодо того, чи є ринок привабливим для входження за попереднім оцінюванням.

Надалі визначаються потенційні групи клієнтів, їх характеристики, та формується орієнтовний перелік вимог до товару для кожної групи (таблиця 5.5).

Таблиця 5.5 Характеристика потенційних клієнтів стартап-проекту

№ п/п	Потреба, що формує ринок	Цільова аудиторія	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
1	Збільшення продуктивності роботи бази даних та залежних від неї систем	Розробники та користувачі програмних систем з використанням баз даних	Розробники повинні мати досвід роботи з базою даних, для більш тонкого налаштування системи, коли користувачі можуть взяти готову запропоновану рекомендацію	Розуміння принципів роботи та легкості у впровадженні. Можливість працювати без інсталяції

Аналіз ринкового середовища складається з таблиць факторів, що перешкоджають ринковому впровадженню проекту, а також тих, що сприяють. Таблиці 5.6 та 5.7 представляють результати.

Таблиця 5.6 Фактори загроз

№ п/п	Фактор	Зміст загрози	Можлива реакція компанії
1.	Велика клієнтська база конкурентів	Конкуренти з досвідом продукту мають сильну базу клієнтів	Розвиток маркетингової кампанії та акційних пропозицій
2	Доступу до БД	Потребує прав доступу та модифікування бази даних	Надання варіантів оптимізації без обов'язкового доступу до даних

Таблиця 5.7 Фактори можливостей

№ п/п	Фактор	Зміст загрози	Можлива реакція компанії
1.	Незалежність від платформи	Система може працювати на будь-якій операційній системі за допомогою терміналу та локальною чи віддаленою базою даних, доступ до якої здійснюється через мережу Інтернет	Вихід на ринок мобільних додатків
2	Фокус продукту на можливостях, яких немає у конкурентів	Реалізація нових функціональних можливостей для користувачів	Планування задач та розподілення між розробниками

Продовження таблиці 5.7

3	Відсутність повноцінних альтернатив	Наявні альтернативи не надають такий набір можливостей	Розширення набору можливостей
---	-------------------------------------	--	-------------------------------

Надалі проводиться аналіз пропозиції: визначаються загальні риси конкуренції на ринку (таблиця 5.8).

Таблиця 5.8 Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства
1. Тип конкуренції: чиста	Існують фірми конкуренти на ринку	Презентація продукту на виставках, надати ширші можливості
2. За рівнем конкурентної боротьби: національний	Закордонні конкуренти	Публікація статей на міжнародних сайтах, можливість вибору мови ПЗ
3. За рівнем конкурентної боротьби: міжгалузева	Спостерігається в пропозиціях на покупку ПЗ, якості функцій.	Розробка програмного забезпечення загального напрямку
4. Конкуренція за видами товарів: товарно-видова	Види товарів однакові – програмне забезпечення	Розроблене ПЗ повинно враховувати недоліки конкурентів

Продовження таблиці 5.8

5. За характером конкурентних переваг: нецінова	Функціональні можливості	Надавати такі функціональні можливості, що не мають конкуренти
6. За інтенсивністю – не марочна	Конкуренція за надання послуг	Популяризація власного проєкту

Результати аналізу умов конкуренції в галузі подано у таблиці 5.9

Таблиця 5.9 Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
	1. EverSQL Query Optimizer 2. DbForge Studio 3. AppOptics APM	Наявність товарних знаків, доступ до ресурсів	Основним постачальником є інтернет ресурси	Система-інформації	немає
Висновки	Конкурентна боротьба неінтенсивна, конкуренти більше спеціалізуються на інших функціях	Є можливість виходу на ринок, бо конкуренти	Постачальники не диктують умов праці	Умови клієнтів змінюються у залежності від ситуації	немає

Аналіз показує, що робота на ІТ арені України можлива, бо конкурентна боротьба неінтенсивна і прямі конкуренти орієнтовані на інші функції.

Обґрунтування факторів конкурентоспроможності наведено у таблиці 5.10.

Таблиця 5.10 Обґрунтування факторів конкурентоспроможності

№ п/п	Фактор конкурентоспроможності	Обґрунтування
1	Функціональна перевага над конкурентами	Система покриває усі можливі шляхи оптимізації бази даних
2	Простота та інтуїтивність використання	Користувач у покроковому режимі виконує рекомендації

Аналіз сильних та слабких сторін стартап-проєкту наведений у таблиці 5.11.

Таблиця 5.11 Порівняльний аналіз сильних та слабких сторін

№ п/п	Фактор конкурентоспроможності	Бали 1-20	Рейтинг програм-конкурентів у порівнянні з DbForge Studio						
			-3	-2	-1	0	+1	+2	+3
1	Споживчі потреби	12				+			
2	Основна результативність	15			+				
3	Технічне обслуговування	12					+		
4	Маркетинговий потенціал	8		+					

Аналіз проєкту за схемою SWOT наведено у таблиці 5.12.

Таблиця 5.12 SWOT-аналіз

Сильні сторони: Якість продукту. Функціональні можливості Доступність	Слабкі сторони: «Невідомість» програмного забезпечення Низька репутація
Можливості: Вихід на міжнародний ринок	Загрози: Блокування інтернет-ресурсу програмного забезпечення

На основі SWOT-аналізу були розроблені альтернативи ринкової поведінки, що представлені у таблиці 5.13.

Таблиця 5.13 Альтернативи ринкового впровадження стартап-проєкту

№ п/п	Альтернатива	Ймовірність отримання ресурсів	Строки реалізації
1	Розширення функціональності	25%	18 міс
2	Проведення конференції-демо для закордонних користувачів	50%	6 міс

Спочатку потрібно вивести розроблену систему на основний ринок, а вже потім шукати можливості розширення програмного функціоналу для користувачів.

5.4 Розроблення ринкової стратегії проєкту

Розроблення ринкової стратегії першим кроком передбачає визначення стратегії охоплення ринку: було проведено опис цільових груп потенційних споживачів (таблиця 5.14)

Таблиця 5.14 Вибір цільових груп потенційних споживачів

№ п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1	Фізичні особи	Потребують недовгих переговорів	Середній	Висока	Складно
2	Підприємства	Готові	Високий	Висока	Просто
3	Державні установи	Готові	Високий	Висока	Складно
4	Стартапери	Готові	Середній	Високий	Просто
Висновок: цільова група - підприємства					

Базова стратегія розвитку представлена у таблиці 5.15.

Таблиця 5.15 Визначення базової стратегії розвитку

№ п/п	Обрана альтернатива розвитку проєкту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
1	Розширення виробничої лінії	Стратегія диференціації	Розширення функціоналу	Стратегія диференціації

Продовження таблиці 5.15.

2	Надання товару важливих з точки зору споживача відмітних властивостей	Стратегія диференціації	Багатоплатформність забезпечує швидкість розробки, що необхідна для стартап-діячів	Стратегія диференціації
---	---	-------------------------	--	-------------------------

Вибір стратегії конкурентної поведінки представлено на таблиці 5.16.

Таблиця 5.16 Визначення базової стратегії конкурентної поведінки.

№ п/п	Чи є проєкт «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати наявних у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки
1	Ні	Шукати нових споживачів	Ні	Стратегія позиціонування

Визначення стратегії позиціонування наведено у таблиці 5.17

Таблиця 5.17 Визначення стратегії позиціонування

№ п/п	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап проєкту	Вибір асоціацій, які мають сформувати комплексну позицію власного проєкту
-------	-------------------------------------	---------------------------	--	---

Продовження таблиці 5.17.

1	Ціна, якість	Стратегія спеціалізації (спирається на диференціацію)	Потребують швидкості розробки, яку надає підтримка багатьох платформ даним продуктом	Якість, надійність, багатоплатформність
---	--------------	---	--	---

5.5 Розроблення маркетингової програми стартап-проекту

Початковим етапом є формування маркетингової концепції товару. У таблиці 5.18 наведені ключові концепції потенційного товару.

Таблиця 5.18 Визначення ключових переваг концепції потенційного товару

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами
1	Широкі можливості оптимізації БД	Система пропонує всі доступні шляхи оптимізації БД, а саме: оптимізація серверу БД, структури та запитів.	«Покриття» усіх можливих шляхів оптимізації
2	Відкритий доступ	Система є відкритою для користування	Користувачі можуть вільно та необмежено користуватися системою

Надалі розробляється трирівнева маркетингова модель товару: уточнюється ідея продукту та/або послуги, його фізичні складові, особливості процесу його надання. Опис трьох рівнів моделі товару наведено у таблиці 5.19.

Таблиця 5.19 Опис трьох рівнів моделі товару

Рівні товару	Сутність та складові		
I. Товар за задумом	Програмний засіб для оптимізації бази даних великих обсягів даних		
II. Товар у реальному виконанні	Властивості/характеристики	М/Нм	Вр/Тх /Тл/Е/Ор
	1. Зручність	1. М	4. Тл
	2. Інтуїтивність	2. М	5. Тл
	3. Швидкодія	3. М	6. Тх
	Якість: тестування та регресія		
	Пакування відсутнє		
	Марка: КЕЕЕМ Оптимізація БД		
III. Товар із підкріпленням	До продажу: Безкоштовна версія		
	Після продажу: 24/7 підтримка		
За рахунок чого потенційний товар буде захищено від копіювання: патент на продукт			

М/Нм – монотонні або немонотонні; Вр/Тх/Тл/Е/Ор – вартісні, технічні, технологічні, ергономічні або органолептичні (останній – для харчових продуктів).

Таблиця 5.20 представляє межі встановлення ціни.

Таблиця 5.20 Визначення меж встановлення ціни

№ п/п	Рівень цін на товари- замінники	Рівень цін на товари- аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
1	3000-5000 грн.	25000 грн.	Всі	3000-25000

Наступним кроком є визначення оптимальної системи збуту, в межах якого приймається рішення (таблиця 5.21).

Таблиця 5.21 Формування системи збуту

№ п/п	Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
1	Отримати більше за менші гроші	Клієнтська база	Тільки виробник	Вертикальна маркетингова система
2	Набір рекомендацій надається у повній версії	Простота доступу	Розробник, веб-сайт, користувач	Збут силами посередника

Концепція маркетингових комунікацій подана у таблиці 5.22.

Таблиця 5.22 Концепція маркетингових комунікацій

№ п/п	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
1	Клієнти дізнаються про продукт з реклами, соціальних мереж	Інтернет, соціальні мережі	Довіра до бренду, ціна	Розповсюдження інформації про продукт	Висвітлення ключових особливих даних про продукт

5.6 Висновки до розділу

Розроблений програмний продукт є конкурентним на ринку та має свої переваги над конкурентами. Визначені маркетингові стратегії та шляхи збуту забезпечать подальший розвиток продукту. Підприємства – основна цільова аудиторія, яка може використовувати систему оптимізації роботи бази даних.

ВИСНОВКИ

Під час роботи над магістерською дисертацією було досліджено питання оптимізації комплексу еколого-економіко-енергетичного моніторингу, у якому використовується база даних із 35-ти таблиць та сформовано набір рекомендацій з оптимізації бази даних, який було застосовано до веб-орієнтованої системи KEEEM.

Під час виконання отримано наступні результати:

1. Проведено аналіз існуючих технік оптимізації систем реляційних баз даних, який показав необхідність проведення попереднього еталонного тестування системи та визначив основні напрями оптимізації: оптимізація налаштувань серверу, оптимізація структури БД та оптимізація запитів.

2. Виконано аналіз веб-орієнтованої системи KEEEM та її бази даних на доцільність використання методів оптимізації. Аналіз виконувався за допомогою штучного навантаження системи виконання еталонних тестів, що виявило потенційні проблемні місця в системі для їх подальшого виправлення.

3. Проведено оптимізацію веб-орієнтованої системи KEEEM за рахунок виправлення проблемних місць, а саме: удосконалення структури, пошукових запитів та налаштування параметрів серверу бази даних. Це пришвидшило роботу всієї системи більш ніж у 7 разів і експериментально довело якість обраних методів оптимізації.

4. Сформовано набір рекомендацій по оптимізації бази даних та продемонстровано його роботу на прикладі бази даних веб-орієнтованої системи KEEEM.

Створений набір рекомендацій надає наступні можливості:

— оптимізація серверу БД: налаштування використання пам'яті, налаштування вводу / виводу, налаштування конкурентного доступу;

- оптимізація структури БД: вибір оптимальних типів даних, створення індексів, нормалізація та денормалізація, кешування і використання зведених таблиць;
- оптимізація запитів до БД: реструктуризація запитів, кеш запитів, підказки оптимізатору запитів.

Для розробки програмного продукту було обрано клієнт-серверну архітектуру. Серверна частина була створена на платформі NodeJS з використанням мінімалістичного фреймворку Express. Основою для клієнтської частини стала веб-бібліотека React з додатковим використанням пакетів react-leaflet (інтерактивна мапа), react-ag-grid (робота з таблицями), read-excel-file (імпорт даних з Excel) та react-bootstrap (візуальне оформлення).

Користувачами системи в першу чергу експерти системи, яким швидкий доступ до даних є незамінним у їх роботі. Додатково комплекс надає відкритий доступ до даних моніторингу для неавторизованих користувачів, тому будь-хто може швидко переглянути дані про проведені / заплановані заходи і відслідкувати прогрес вирішення екологічних проблем.

Практичне значення отриманих результатів полягає у тому, що розроблений та застосований до системи комплексного еколого-економіко-енергетичного моніторингу набір рекомендацій дасть змогу користувачам системи отримати зручний доступ до необхідних даних, що пришвидшить вирішення проблем, які безпосередньо впливають на здоров'я населення.

Набір рекомендацій орієнтований базу даних системи KEEEM, але розробники можуть брати його в увагу при оптимізації власних баз з великими обсягами даних.

Таким чином, у рамках веб-системи реалізовано функціонал оптимізації бази даних з використанням різних методів та технік оптимізації, налагоджено структуру та використання запитів. Отримані результати показали, що наведені для кожного запиту варіанти оптимізації можуть пришвидшити час виконання запитів у декілька разів. Тому сформований набір рекомендацій можна використовувати як основу для подальших розробок у сфері оптимізації продуктивності баз даних великих обсягів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Костриця С. В., Левицький І. В., Полягушко Л. Г. Еталонне тестування та аналіз реляційних баз даних моніторингових систем з великими обсягами даних. *Розвиток суспільства та науки в умовах цифрової трансформації*: матеріали III Міжнародної студентської наукової конференції, м. Луцьк, 16 груд. 2022 р.
2. Запорожець А. О. Analysis of means for monitoring air pollution in the environment. *Science-based technologies*. 2017. Т. 35, № 3. URL: <https://doi.org/10.18372/2310-5461.35.11844> (дата звернення: 10.10.2022).
3. Красовський Г. Я. Космічний моніторинг безпеки водних екосистем із застосуванням геоінформаційних технологій: монографія. Київ: Інтертехнологія, 2008. 480 с.
4. Кучерявий В. П. Урбоекотологія. Львів: Світ, 1999. 360 с.
5. Модели прогнозирования: общая классификация. *Математическое бюро | Прогнозирование на ОПЭМ*. URL: <https://www.mbbureau.ru/blog/modeli-prognozirovaniya-obshchaya-klassifikaciya> (дата звернення: 10.10.2022).
6. The modern javascript tutorial. *The Modern JavaScript Tutorial*. URL: <https://javascript.info/> (дата звернення: 10.10.2022).
7. West M. HTML5 foundations. Wiley & Sons, Incorporated, John, 2012. 384 с.
8. Sikora P. Professional CSS3. Packt Publishing, 2016. 362 с.
9. React-Bootstrap. *React-Bootstrap · React-Bootstrap Documentation*. URL: <https://react-bootstrap.github.io/> (дата звернення: 10.10.2022).
10. React – A JavaScript library for building user interfaces. *React – A JavaScript library for building user interfaces*. URL: <https://reactjs.org/> (дата звернення: 10.10.2022).
11. Learning UML 2.0: A pragmatic introduction to UML. O'Reilly, 2008.
12. Express - Node.js web application framework. *Express - Node.js web application framework*. URL: <https://expressjs.com/> (дата звернення: 10.10.2022).

13. MySQL :: MySQL Documentation. *MySQL :: Developer Zone*. URL: <https://dev.mysql.com/doc/> (дата звернення: 10.10.2022).
14. Оптимізація JOIN з допомогою тимчасових таблиць. *Практика і філософія програміста* / my2i.ru. URL: <https://my2i.ru/article/optimizaciya-join-s-pomoshchyu-vremennyh-tablic> (дата звернення: 10.10.2022).
15. Stephens J., Russell C. Beginning mysql database design and optimization. Berkeley, CA : Apress, 2004. URL: <https://doi.org/10.1007/978-1-4302-0729-0> (дата звернення: 10.10.2022).
16. EverSQL | your database, just faster. *EverSQL | Automatic SQL Query Optimization for MySQL & PostgreSQL*. URL: <https://www.eversql.com/> (дата звернення: 10.10.2022).
17. Тейер Т. Надійність програмного забезпечення. М., 1981. 325 с.
18. Schwarts B., Zaitsev P., Tkachenko V. Benchmarking MySQL. *High Performance MySQL*. 3rd ed. USA, 2012. P. 35–67.
19. Using sysbench for OLTP workload performance benchmark. *FlamingBytes*. URL: <https://www.flamingbytes.com/posts/sysbench/> (дата звернення: 10.10.2022).
20. Белоус Р., Крилов Є., Анікін В. Методи оптимізації запитів розподілених БД. *Адаптивні системи автоматичного управління*. 2021. Т. 2, № 39. С. 3–11. URL: <https://doi.org/10.20535/1560-8956.39.2021.247364> (дата звернення: 10.10.2022).
21. Ab M. MySQL administrator's guide. MySQL Press, 2004. 448 с.
22. Nodejs in action. Manning Publications, 2012.
23. Javeed A. Performance optimization techniques for reactjs. *2019 IEEE international conference on electrical, computer and communication technologies (ICECCT)*, м. Coimbatore, India, 20–22 лют. 2019 р. 2019. URL: <https://doi.org/10.1109/icecct.2019.8869134> (дата звернення: 10.10.2022).
24. Каленіченко Ю. Б., Калениченко Ю. Б., Kalenichenko Y. B. Екологічний моніторинг. Суспільний екологічний моніторинг : Theses. 2005. URL: <http://essuir.sumdu.edu.ua/handle/123456789/21834> (дата звернення: 10.10.2022).

25. Попов О. О. Методи аналізу ризиків в екології. *Збірник наукових праць Інституту проблем моделювання в енергетиці ім. Г.Є. Пухова*. 2013. Вип. 69. С. 19–29.
26. Веселовський О. В., Гальберг Т. В. Екологічний моніторинг територій, які контролюються та прогноз катастрофічних явищ. *Geoinformatics 2011*, м. Київ, Ukraine, 10–13 трав. 2011 р. Netherlands, 2011. URL: <https://doi.org/10.3997/2214-4609.20145050> (дата звернення: 10.10.2022).
27. Когнитивное усиление системы принятия и поддержки решений / Е. А. Яковлева та ін. *Leadership and management*. 2021. Т. 8, № 1. С. 127–144. URL: <https://doi.org/10.18334/lim.8.1.111421> (дата звернення: 10.10.2022).
28. Косс В. А. Інформаційні моделі для комплексної інтелектуальної підтримки прийняття рішень в системах ситуаційного управління : автореф. дис. ... канд. техн. наук. Київ, 2007. 28 с.
29. Казмірчук С. В. Методологія оцінювання ризиків безпеки ресурсів інформаційних систем : автореф. дис. ... д-ра техн. наук. Київ, 2017. 36 с.
30. Яциковська У. О. Методи та засоби моделювання безпечної комунікації в системі клієнт-сервер : автореф. дис. ... канд. техн. наук. Львів, 2013. 20 с.
31. Шарапова С. В. Екологічний моніторинг у правовідносинах використання природних ресурсів. *Проблеми законності*. 2008. Вип. 95. С. 185–188.
32. Сліпченко В. Г., Полягушко Л. Г., Круш О. Є. Система комплексного еко-енерго-економічного моніторингу для оптимізації управлінських рішень (області, району та міста). *ВІСНИК СХІДНОУКРАЇНСЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ імені Володимира Даля*. 2021. № 4(268). С. 13–20. URL: <https://doi.org/10.33216/1998-7927-2021-268-4-13-20> (дата звернення: 10.10.2022).
33. Куценко И. Л. Оптимизация запросов в реляционных базах данных. *Естественные и Технические Науки*. 2021. № 03. С. 129–134. URL: <https://doi.org/10.37882/2223-2966.2021.03.22> (дата звернення: 10.10.2022).
34. Инструментальные средства построения баз данных и знаний на персональных ЭВМ. Киев, 1989. 83 с.

35. Троянський О. І. Моніторинг якості води. Житомир : Волинь, 2004. 192 с.
36. Gao G., Li J., Wen Y. DeepComfort: energy-efficient thermal comfort control in buildings via reinforcement learning. *IEEE internet of things journal*. 2020. Т. 7, № 9. С. 8472–8484. URL: <https://doi.org/10.1109/jiot.2020.2992117> (дата звернення: 10.10.2022).
37. Enabling efficient process mining on large data sets: realizing an in-database process mining operator / R. Dijkman та ін. *Distributed and parallel databases*. 2019. Т. 38, № 1. С. 227–253. URL: <https://doi.org/10.1007/s10619-019-07270-1> (дата звернення: 10.10.2022).
38. Baumer B. S. A grammar for reproducible and painless extract-transform-load operations on medium data. *Journal of computational and graphical statistics*. 2018. Т. 28, № 2. С. 256–264. URL: <https://doi.org/10.1080/10618600.2018.1512867> (дата звернення: 10.10.2022).
39. Ellenberg S. S. Monitoring data on data monitoring. *Clinical trials: journal of the society for clinical trials*. 2004. Т. 1, № 1. С. 6–8. URL: <https://doi.org/10.1191/1740774504cn001ed> (дата звернення: 10.10.2022).
40. Lavaee R., Criswell J., Ding C. Codestitcher: inter-procedural basic block layout optimization. *The 28th international conference*, м. Washington, DC, USA, 16–17 лют. 2019 р. New York, New York, USA, 2019. URL: <https://doi.org/10.1145/3302516.3307358> (дата звернення: 10.10.2022).
41. Семькина Д., Русаков А. Современные средства разработки серверной части web-сайтов. *SWorldJournal*. 2018. № 08-01. С. 61–74. URL: <https://doi.org/10.30888/2663-5712.2021-08-01-026> (дата звернення: 10.10.2022).
42. Берзтисс А. Т. Структуры данных. М., 1974. 408 с.
43. Классификация структур данных : Theses / М. М. Соколов та ін. 2011. URL: <http://essuir.sumdu.edu.ua/handle/123456789/9892> (дата звернення: 10.10.2022).
44. 13 best SQL query optimization tools | phoenixnap KB. *Knowledge Base by phoenixNAP*. URL: <https://phoenixnap.com/kb/sql-query-optimization-tool> (дата звернення: 10.10.2022).

звернення: 10.10.2022).

45. Rubalcava R. Introducing ArcGIS API 4 for JavaScript: Turn Awesome Maps into Awesome Apps. Apress, 2017. 135 с.
46. Никитна Е. В. Микробиологический мониторинг наземных экосистем. Новосибирск, 1991. 218 с.
47. Тухтарьев А. С. Екологічний моніторинг забруднених учбових приміщень формальдегідом : Магістерська робота. 2020.
URL: <https://dspace.znu.edu.ua/jspui/handle/12345/1494> (дата звернення: 10.10.2022).
48. Лавриненко С. Управління сталим розвитком: екологічний аспект. *Education and science of today: intersectoral issues and development of sciences*. 2021.
URL: <https://doi.org/10.36074/logos-19.03.2021.v1.25> (дата звернення: 10.10.2022).
49. Попов О. О. Методи аналізу ризиків в екології. *Збірник наукових праць Інституту проблем моделювання в енергетиці ім. Г.Є. Пухова*. 2013. Вип. 69. С. 19–29..
50. Мартиненко В. О., Мартыненко В. А., Martynenko V. О. Еколого-економічний моніторинг як засіб управління охороною довкілля на муніципальному рівні : Theses. 2004. URL: <http://essuir.sumdu.edu.ua/handle/123456789/60632> (дата звернення: 10.10.2022).
51. Мацібора О. В. Веб-орієнтовані геоінформаційні системи та їх використання для обробки палеогеографічних даних. *Український географічний журнал*. 2019. № 1 (105). С. 51–58.

ДОДАТОК А

Оптимізація запитів до великих обсягів даних еколого-економіко-
енергетичного моніторингу

Апробації

УКР.НТУУ“КПІ ім. Ігоря Сікорського” ТР-1188мп_22М

Аркушів 6

2022

ЕТАЛОННЕ ТЕСТУВАННЯ ТА АНАЛІЗ РЕЛЯЦІЙНИХ БАЗ ДАНИХ МОНІТОРИНГОВИХ СИСТЕМ З ВЕЛИКИМИ ОБСЯГАМИ ДАНИХ

Костриця Сергій Володимирович

здобувач вищої освіти інституту атомної та теплової енергетики
*Національний технічний університет «Київський політехнічний інститут ім. Ігоря
Сікорського», Україна*

Левицький Ілля В'ячеславович

здобувач вищої освіти інституту атомної та теплової енергетики
*Національний технічний університет «Київський політехнічний інститут ім. Ігоря
Сікорського», Україна*

Науковий керівник: Полягушко Любов Григорівна

канд. тех. наук, доцент кафедри цифрових технологій в енергетиці ІАТЕ
*Національний технічний університет «Київський політехнічний інститут ім. Ігоря
Сікорського», Україна*

Моніторингові системи завжди повинні працювати з актуальними даними, тому задля забезпечення системи новою інформацією використовується велика кількість датчиків і пристроїв, що отримують сирі (необроблені) дані з різних точок земного шару та неперервно виконують операції вставки до бази даних моніторингових систем. З часом база набирає великі обсяги даних і у випадку поганого проектування схеми та некоректних налаштувань – робота з базою даних сильно уповільнюється.

Прикладом погано побудованої системи з великими обсягами даних є система комплексного еколого-економіко-енергетичного моніторингу (KEEEM), тому для дослідження конкретних проблемних місць у її роботі було застосовано еталонне тестування до бази даних системи.

Еталонне тестування – це створення робочого навантаження, призначеного для того, щоб перевести систему у стресовий режим. Основною ціллю еталонного тестування є вивчення поведінки системи, однак не менш важливими причинами для

його виконання є відтворення бажаного стану системи та випробування нового обладнання на стійкість [1].

Існують дві основні стратегії еталонного тестування:

- *повнотекстове* – тестування всього додатку, включаючи веб-сервер, програмний код додатку, мережу та базу даних. При повнотекстовому тестуванні можна зрозуміти чи є СКБД вузьким місцем та як веде себе кеш кожної частини додатку;

- *компонентне* (модульне) – тестування лише СКБД, що допомагає порівнювати різні схеми та запити; протестувати конкретну проблему, яка виникла в додатку; уникнути довготривалого еталонного тестування, обмежуючись коротким тестом, який дозволить швидко внести зміни та виміряти їх.

Гарним інструментом компонентного тестування є *sysbench*, завдяки його багатопотоковій природі він дозволяє отримати представлення про швидкодію системи з точки зору таких важливих для роботи факторів як файловий ввід / вивід, планувальник операційної системи, розподіл пам'яті і швидкості передачі даних, потоків POSIX та самого сервера бази даних.

Для інтерактивних системи розрахованих на велику кількість користувачів краще всього підходить еталонний тест, який виміряє швидкодію оперативної обробки транзакцій (OLTP). Загальноприйнятою одиницею виміру є кількість транзакцій на секунду [2].

Було проведено таке тестування до бази даних KEEEM за допомогою інструменту *sysbench* для таблиці, що містить 1 мільйон записів показників з датчиків. Тест виконувався 60 секунд в режимі читання даних із використанням восьми конкурентних потоків (рис. 1).

```

kostr@MacBook-Pro ~ % sysbench /opt/homebrew/share/sysbench/tests/include/oltp_legacy/oltp.lua \
--oltp-table-size=1000000 --mysql-host=keem.com.ua --mysql-db=KEEM_Magistr \
[--mysql-user=Keem_Magistr --mysql-password=KeEm_MaGiStR2022 \
[> --time=60 --oltp-read-only=on --max-requests=0 --threads=8 run
sysbench 1.0.20 (using system LuaJIT 2.1.0-beta3)

Running the test with following options:
Number of threads: 8
Initializing random number generator from current time

Initializing worker threads...

Threads started!

SQL statistics:
  queries performed:
    read:                65632
    write:               0
    other:              9376
    total:             75008
  transactions:        4688 (78.02 per sec.)
  queries:             75008 (1248.24 per sec.)
  ignored errors:      0 (0.00 per sec.)
  reconnects:         0 (0.00 per sec.)

General statistics:
  total time:          60.0903s
  total number of events: 4688

Latency (ms):
  min:                 80.90
  avg:                 102.46
  max:                 232.77
  95th percentile:    0.00
  sum:                480320.22

Threads fairness:
  events (avg/stddev): 586.0000/3.57
  execution time (avg/stddev): 60.0400/0.03

```

Рис. 1. Результат виконання першого тесту OLTP

Отримані результати показали, що потоки завантажені нерівномірно і це вплинуло на швидкість виконання транзакцій. Тому в першу чергу було виконано огляд параметрів серверу бази даних, внаслідок чого було виявлено некоректні налаштування конкурентного доступу, які не були достатніми для системи з таким великим обсягом даних.

Після виправлень було виконано повторне тестування (рис. 2).

```

kostr@MacBook-Pro ~ % sysbench /opt/homebrew/share/sysbench/tests/include/oltp_legacy/oltp.lua \
--oltp-table-size=1000000 --mysql-db=KEEEM_Magistr_Optimized \
--mysql-user=root --mysql-password=password \
[--time=60 --oltp-read-only=on --max-requests=0 --threads=8 run
sysbench 1.0.20 (using system LuaJIT 2.1.0-beta3)

Running the test with following options:
Number of threads: 8
Initializing random number generator from current time

Initializing worker threads...

Threads started!

SQL statistics:
  queries performed:
    read:                3892182
    write:                0
    other:                556026
    total:                4448208
  transactions:         278013 (4633.27 per sec.)
  queries:              4448208 (74132.29 per sec.)
  ignored errors:       0      (0.00 per sec.)
  reconnects:           0      (0.00 per sec.)

General statistics:
  total time:            60.0033s
  total number of events: 278013

Latency (ms):
  min:                   0.59
  avg:                   1.73
  max:                   29.62
  95th percentile:      0.00
  sum:                   479837.36

Threads fairness:
  events (avg/stddev):   34751.6250/95.25
  execution time (avg/stddev): 59.9797/0.00

```

Рис. 2. Результат виконання другого тесту OLTP

Отримані результати повторного тестування бази даних KEEEM показали ефективність виконаних налаштувань, які пришвидшили роботу системи приблизно в 60 разів.

Еталонні тести OLTP дуже зручні для швидкого порівняння різних систем і диску є неоціненними для виправлення несправностей та ізолювання помилкових компонентів при виникненні проблем зі швидкодією системи.

У результаті проведення еталонних тестів OLTP до бази даних системи KEEEM було виявлено некоректні налаштування конкурентного доступу і виправлено згідно з вимогами до баз даних великих обсягів.

Список використаних джерел:

1. Schwartz B., Zaitsev P., Tkachenko V. Benchmarking MySQL. *High Performance MySQL*. 3rd ed. USA, 2012. P. 35–67.
2. Using sysbench for OLTP workload performance benchmark. *FlamingBytes*. URL: <https://www.flamingbytes.com/posts/sysbench/> (дата звернення: 07.12.2022).

СЕРТИФІКАТ УЧАСНИКА

КОСТРИЦЯ СЕРГІЙ ВОЛОДИМИРОВИЧ

ВЗЯВ(-ЛА) УЧАСТЬ У ІІІ МІЖНАРОДНІЙ СТУДЕНТСЬКІЙ НАУКОВІЙ КОНФЕРЕНЦІЇ

**РОЗВИТОК СУСПІЛЬСТВА ТА НАУКИ
В УМОВАХ ЦИФРОВОЇ ТРАНСФОРМАЦІЇ**

16 ГРУДНЯ 2022 | М. ЛУЦЬК, УКРАЇНА

Конференцію схвалено УКРІНТЕІ (Посвідчення №456 від 05.10.2022)



Матеріали учасника конференції опубліковані та знаходяться у відкритому доступі на умовах ліцензії CC BY 4.0 за посиланням: <https://ojs.ukriogos.in.ua/index.php/liga/issue/view/16.12.2022>



ДИРЕКТОР МОЛОДІЖНОЇ НАУКОВОЇ ЛІГИ
ГОЛОВА ОРГКОМІТЕТУ КОНФЕРЕНЦІЇ
ІГОР КОРЕНЮК

