МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Харківський національний університет імені В. Н. Каразіна

Факультет: **ННІ Каразінський банківський інститут**
Кафедра: **Інформаційних технологій та математичного моделювання**
Спеціальність: **122 Комп'ютерні науки**
Освітня програма: **Комп'ютерні науки**

Група: АК-21М **денна форма навчання**

КВАЛІФІКАЦІЙНА МАГІСТРЕСЬКА РОБОТА

на тему:

«ДОСЛІДЖЕННЯ СУЧАСНИХ ТЕНДЕНЦІЙ У СФЕРІ ОПТИМІЗАЦІЇ ПРОЦЕСІВ УПРАВЛІННЯ БАЗАМИ ДАНИХ БІЗНЕС-КОМПАНІЙ»
ЗА НАКАЗОМ №4601-5_3262 ВІД 15 вересня 2025 РОКУ

здобувача вищої освіти **Свинаренко Анастасії Анатоліївни**

Робота допущена до захисту в ЕК
протокол кафедри ІТММ №__ від _____2025 р.

Завідувач кафедри ІТММ
к. п. н., доцент
_____ Н. І. Стяглик

Науковий керівник
к. ф.-м. н., доцент
_____ Л. Д. Філатова

м. Харків 2025 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Харківський національний університет імені В. Н. Каразіна

Факультет навчально-науковий інститут "Каразінський банківський інститут"
Кафедра інформаційних технологій та математичного моделювання
Рівень вищої освіти другий (магістерський)
Спеціальність 122 Комп'ютерні науки
Освітня програма Комп'ютерні науки

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ **Н. І. Стяглик**
Підпис ініціали, прізвище

"15" вересня 2025 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ (ПРОЄКТ)

Свинаренко Анастасії Анатоліївни
(прізвище, ім'я, по батькові студента)

1. Тема роботи «Дослідження сучасних тенденцій у сфері оптимізації процесів управління базами даних бізнес-компаній»

керівник роботи к. ф.-м. н., доцент Л. Д. Філатова
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від "15" вересня 2025 року № 4601-5_3262

2. Строк подання студентом роботи 24 листопада 2025 року

3. Перелік питань, які потрібно розробити:
У розділі 1: розглянути теоретичні основи функціонування баз даних і систем управління ними, узагальнити принципи оптимізації процесів керування базами даних і проаналізувати їхню специфіку у бізнес-компаніях.
У розділі 2: проаналізувати ключові аспекти оптимізації керування базами даних, дослідити механізми індексації, оптимізації SQL-запитів, сучасні хмарні рішення, механізми забезпечення захисту даних і застосування технологій штучного інтелекту для підвищення ефективності управління базами даних.
У розділі 3: провести практичний аналіз впливу індексації й оптимізованих SQL-запитів на продуктивність системи, розглянути приклади застосування оптимізаційних рішень у бізнес-компаніях.

4. План роботи

№ з/п	Назви етапів роботи
1	Вибір здобувачем теми кваліфікаційної магістерської роботи
2	Затвердження плану і завдання кваліфікаційної магістерської роботи
3	Здача кваліфікаційної магістерської роботи керівнику
4	Підпис кваліфікаційної магістерської роботи керівника
5	Підпис кваліфікаційної магістерської роботи у нормоконтролера
6	Допуск завідувачем кафедри до захисту кваліфікаційної магістерської роботи
7	Захист кваліфікаційної магістерської роботи

5. Дата видачі завдання 15 вересня 2025 року

Студент _____ А. А. Свинаренко
підпис ініціали, прізвище

Керівник роботи _____ Л. Д. Філатова
підпис ініціали, прізвище

РЕФЕРАТ
НА КВАЛІФІКАЦІЙНУ МАГІСТЕРСЬКУ РОБОТУ
«ДОСЛІДЖЕННЯ СУЧАСНИХ ТЕНДЕНЦІЙ У СФЕРІ ОПТИМІЗАЦІЇ ПРОЦЕСІВ УПРАВЛІННЯ БАЗАМИ ДАНИХ БІЗНЕС-КОМПАНІЙ»
Свинаренко Анастасії Анатоліївни

Кваліфікаційна магістерська робота містить: 70 сторінок, 3 таблиці, 14 рисунків, список літератури з 52 найменувань.

Об'єктом дослідження є корпоративні бази даних і їхні системи керування.

Предмет дослідження – методи і технології оптимізації управління базами даних у бізнес-компаніях.

Мета кваліфікаційної магістерської роботи полягає у дослідженні й аналізі сучасних тенденцій й інструментів оптимізації управління базами даних у бізнес-компаніях, а також наданні актуальних рекомендацій щодо оптимізації процесів обробки та зберігання даних у бізнес-компаніях.

Завданнями кваліфікаційної магістерської роботи є:

− у першому розділі розглянути теоретичні основи функціонування БД і систем управління ними, узагальнити принципи оптимізації процесів керування БД та проаналізувати їхню специфіку у бізнес-компаніях;

− у другому розділі проаналізувати ключові моделі баз даних у контексті їхнього впливу на оптимізацію керування БД, дослідити механізми індексації, розкрити принципи оптимізації SQL-запитів та їх роль у зменшенні витрат обчислювальних ресурсів; розглянути сучасні хмарні рішення і механізми забезпечення захисту даних; проаналізувати можливості застосування технологій штучного інтелекту для підвищення ефективності управління базами даних;

− у третьому розділі провести практичний аналіз впливу індексації та оптимізованих SQL-запитів на продуктивність системи, порівняти підходи до підвищення ефективності роботи БД, розглянути реальні приклади застосування оптимізаційних рішень у бізнес-компаніях і здійснити оцінку сучасних інструментів СУБД та хмарних платформ.

Актуальність дослідження пов’язана зі зростанням обсягів корпоративних даних, адаптації до змін на ринку і необхідністю підвищення ефективності роботи інформаційних систем у бізнес-компаніях, щоб забезпечити конкурентоспроможність підприємства.

За результатами дослідження обґрунтовано важливість оптимізації процесів керування баз даних для бізнес-компаній; визначено ключові підходи та інструменти оптимізації процесів управління БД у бізнес-компаніях, проаналізовано ефективність індексації, оптимізації SQL-запитів, застосування хмарних технологій, розглянуто сучасні механізми забезпечення безперервності та захисту даних методів, поточний стан і перспективи використання штучного інтелекту для підвищення продуктивності керування базами даних.

Практична новизна: виявлення і систематизація ефективних підходів і технологій оптимізації процесів керування базами даних у бізнес-компаніях, що підвищує рівень прийняття рішень керівниками.

Одержані результати можуть бути використані для оптимізації процесів керування баз даних у бізнес-компаніях.

КЛЮЧОВІ СЛОВА: БАЗА ДАНИХ, СИСТЕМА УПРАВЛІННЯ БАЗАМИ ДАНИХ, ОПТИМІЗАЦІЯ, БІЗНЕС-КОМПАНІЯ, ІНДЕКСАЦІЯ, SQL, ХМАРНІ СЕРВІСИ.

ABSTRACT
AT QUALIFICATION MAGISTER WORK
« RESEARCH ON CURRENT TRENDS IN THE OPTIMIZATION OF BUSINESS COMPANY DATABASE MANAGEMENT PROCESSES »
Anastasiia Svynarenko

The master's thesis contains 70 pages, 3 tables, 14 drawings, a list of references of 52 titles.

The object of the study is corporate databases and their management systems..

The subject of the study is the methods and technologies for optimizing database management in business companies.

The purpose of the master's qualification work: to to research and analyze current trends and tools for optimizing database management in business companies, as well as to provide relevant recommendations for optimizing data processing and storage processes in business companies.

The tasks of the master's qualification work are

− in the first section to examine the theoretical foundations of database functioning and database management systems, to summarize the principles of database management process optimization, and to analyze their specifics in business companies;

− in the second section to analyze key database models in the context of their impact on database management optimization, to investigate indexing mechanisms, to reveal the principles of SQL query optimization and their role in reducing computing resource costs; to consider modern cloud solutions and data protection mechanisms; analyze the possibilities of applying artificial intelligence technologies to improve the efficiency of database management;

− in the third section to conduct a practical analysis of the impact of indexing and optimized SQL queries on system performance, compare approaches to improving database efficiency, consider real-world examples of the application of optimization solutions in business companies, and evaluate modern DBMS tools and cloud platforms.

The relevance of the study is related to the rapid growth of corporate data volumes, adaptation to market changes, and the need to improve the efficiency of information systems in business companies to ensure the competitiveness of the enterprise.

According to the results of the research the importance of optimizing database management processes for business companies has been substantiated; key approaches and tools for optimizing database management processes have been identified, the effectiveness of indexing, SQL-query optimization, and the use of cloud technologies has been analyzed, modern mechanisms for ensuring continuity and data protection have been considered, and the current state and prospects for using artificial intelligence to enhance the productivity of database management have been examined.

Practical novelty: identification and systematization of effective approaches and technologies for optimizing database management processes in business companies, which improves the level of decision-making by managers.

The obtained results can be applied to optimize database management processes in business companies.

KEYWORDS: DATABASE, DATABASE MANAGEMENT SYSTEM, OPTIMIZATION, BUSINESS COMPANY, INDEXING, SQL, CLOUD SERVICES.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧОК, СИМВОЛІВ І ТЕРМІНІВ

БД – База даних

ІТ – Інформаційні технології

СУБД – Система управління базами даних

ШІ – Штучний інтелект

DBaaS – Data Base as a Service

DWaaS – Data Warehouse as a Service

ISO – International Organization for Standardization

OLAP – Online Analytical Processing

OLTP – Online Transaction Processing

SQL – Structured Query Language

ВСТУП

У сучасних бізнес-компаніях обсяги даних стрімко зростають, а вимоги до швидкості обробки інформації та безперервності роботи систем щороку посилюються. Бази даних стають ключовим компонентом цифрової інфраструктури підприємств, забезпечуючи підтримку операційних процесів, аналітики, фінансових операцій та стратегічного управління.

Актуальність обраної теми дослідження зумовлена потребою бізнес-компаній адаптації до технологічних змін і підвищенням ефективності управління компанією за допомогою ефективного використання БД. У сучасному ринковому середовищі, який є надзвичайно динамічним, для компаній якість роботи БД безпосередньо впливає на швидкість прийняття рішень, конкурентоспроможність і стійкість до збоїв.

Неефективні запити, невдало обрані моделі даних чи неправильно налаштовані СУБД призводять до значних фінансових втрат, простоїв та зниження конкурентоспроможності компанії. У цих умовах оптимізація процесів управління базами даних набуває особливої важливості, оскільки саме від ефективності індексації, структури запитів, вибору архітектури та використання хмарних інструментів залежить продуктивність і, загалом, роботу бізнесу.

Метою кваліфікаційної роботи є дослідження сучасних підходів та практичних механізмів оптимізації баз даних у бізнес-середовищі; надання рекомендацій щодо підвищення ефективності роботи корпоративних інформаційних систем.

Завданнями даної роботи є:

− вивчити загальні теоретичні аспекти БД, системи управління і їх застосування в бізнесі;

− дослідити теоретичні та прикладні підходи до оптимізації процесів управління БД;

− визначити специфіку роботи з корпоративними даними й оптимізації

процесів управління БД у бізнес-компаніях;

− дослідити вагомість й насущність класичних й сучасних моделей БД, індексації, оптимізації запитів і хмарних механізмів у контексті підвищення продуктивності БД;

− розглянути актуальні аналітичні рейтинги популярних інструментів і сервісів, що сприяють оптимізації процесів керування БД;

− провести практичний аналіз впливу індексації та оптимізованих SQL-запитів на продуктивність системи;

– сформувати рекомендації щодо використання сучасних технологій, включаючи DBaaS і засобів штучного інтелекту для покращення управління базами даних.

Об'єктом дослідження є корпоративні бази даних і їхні системи керування. Предметом дослідження є методи, моделі та інструменти оптимізації продуктивності баз даних у бізнес-компаніях.

Робота складається зі вступу, трьох розділів та висновку.

У вступі представлено актуальність роботи, сформульовано мету та відповідні завдання, об’єкт та предмет дослідження, наведено загальну структуру роботи.

У першому розділі «Теоретичні засади керування й оптимізації баз даних у бізнес-компаніях» розглянуто загальні теоретичні аспекти баз даних, описано їх основні характеристики, надано знання про СУБД. Наведено теоретичні засади загальної оптимізації процесів управління базами даних, що включають ключові методи підвищення продуктивності та ефективного використання ресурсів. Проаналізовано теоретичні основи бізнес-компаній та їхні ознаки. Окремо увагу приділено специфіці і принципам оптимізації процесів керування базами даних у бізнес-компаніях де вимоги до масштабованості, доступності та швидкості обробки даних є особливо критичними.

Другий розділ роботи присвячено аналітичному огляду актуальних моделей баз даних, механізмів індексації, оптимізації SQL-запитів і сучасних

хмарних сервісів, що впливають на підвищення ефективності обробки корпоративних даних. У ньому також розглядаються інструменти забезпечення безперервності роботи, захисту даних і можливості застосування технологій штучного інтелекту в управлінні БД, що формують комплексний підхід до підвищення продуктивності інформаційних систем підприємств.

У третьому розділі наведено практичні експерименти з індексацією та SQL-оптимізацією, порівняно різні підходи до підвищення продуктивності БД та оцінено ефективність застосування хмарних сервісів у бізнес-середовищі. Розглянуто реальні приклади компаній, зокрема стосовно застосування індексів у різних СУБД і впровадження хмарних рішень. Для аналізу інструментів СУБД і хмарних платформ використано дані авторитетних джерел, що дозволило порівняти сучасні рішення й інструменти роботи з БД.

Висновок висвітлює інформацію щодо підсумків дослідження, його наукової та практичної значущості, можливі перспективи подальшого розвитку.

РОЗДІЛ 1
ТЕОРЕТИЧНІ ЗАСАДИ КЕРУВАННЯ Й ОПТИМІЗАЦІЇ БАЗ ДАНИХ У БІЗНЕС-КОМПАНІЯХ

1.1. Загальна теорія баз даних і систем керування базами даних

Інформація – це всілякі відомості про абиякий предмет, факт, властивість, сутність, процес, явище, ситуацію, стан, взаємодію тощо, які виступають як об’єкт. Функціонал призначений для різноманітних операцій, наприклад, таких як передача, зберігання, перетворення, використання цих відомостей.

Інформація зростає у своїй цінності у той момент, коли вона стає зафіксованою у певній формі. Ця унікальність відрізняє інформацію від фундаментального для БД терміну – даних. Над формулюванням найбільш комплексного тлумачення цього терміну впродовж багатьох років працювали й продовжують працювати представники різних галузей.

Стандарт ISO/IEC 2382:2015, яким послуговуються усі професіонали, пропонує наступне тлумачення: дані – інтерпретоване представлення інформації у формалізований спосіб, придатний для передачі, інтерпретації або обробки.

Іншими словами, дані являють собою специфічну форму подання інформації, організовану за певними правилами, що визначаються контекстом ситуації, потребами системи чи умовами її функціонування. Ці правила можуть ґрунтуватися на міжнародних або національних стандартах, нормативних актах чи внутрішніх корпоративних угодах, що регламентують формат і спосіб опису даних.

Залежно від вимог, дані можуть бути подані у вигляді таблиць, графіків, схем, списків або деревоподібних структур. Варіативність інтерпретацій дає змогу по-різному сприймати одні й ті самі дані, змінюючи контекст їхнього змістового наповнення та напрям подальшого аналізу.

Із появою комп’ютерів і СУБД процеси зберігання, опрацювання й аналізу даних зазнали докорінної трансформації. Поняття БД набуло ключового значення в розвитку сучасних інформаційних технологій, оскільки відображає потребу в систематизації та ефективному управлінні інформацією. Використання БД дало змогу замінити ручне внесення відомостей стандартизованими електронними процедурами.

У стандарт ISO/IEC 2382:2015 БД має наступне визначення: база даних – це сукупність даних, які структуровані згідно з концептуальною моделлю, що визначає їхні характеристики і взаємозв’язки між відповідними сутностями і підтримує одну або декілька галузей використання.

БД завжди передбачає, що воно відповідає таким вимогам:

- БД має містити необхідну і достатню кількість даних для розв’язання конкретних задач існуючого або потенційного користувача;
- елементи БД повинні бути структуровані, організовані, пов’язані між собою;
- форма даних має бути придатна для їхнього швидкого використання на практиці.

Створення ефективної БД – це складний, багатоетапний процес, який не закінчується на етапі вибору й налаштування програмного забезпечення. Він вимагає ретельного планування та аналізу, щоб забезпечити, що кінцевий продукт буде відповідати бізнес-вимогам, бути масштабованим і легко керованим. Цей процес зазвичай складається кількох етапів, одні з ключових які це аналіз вимог, проектування та реалізація.

Внесення інформації до БД може бути здійснене різноманітними шляхами: від мануального до автоматичного. Залежно від потреб користувачів, етапів роботи й ресурсів системи – це може бути як метод, який відомий з початків роботи з базами, а саме ручне введення безпосередньо у форму чи таблицю, так і більш сучасний підхід – імпорт із зовнішніх джерел, наприклад файлів чи інших баз.

Утім ще більш прогресивним і професійним методом вважається

завантаження даних через програмні інтерфейси (API), адже використання такого підходу забезпечує швидкість, високу точність і гарантує безперервність обміну інформацією.

На фазі після надходження даних до БД відбувається валідація, ціль якої нею є перевірка цілісності, достовірності та відповідність визначеній структурі. Насамперед, такий механізм унеможливлює внесення некоректної чи випадкової інформації, що на практиці дає чіткий план мати справу з даних у належному форматі, оскільки дані проходять перевірку завчасно встановлених параметрів, наприклад, типу даних, обмеження довжини, цілісності, вимога унікальності поля тощо. Перевірка даних на регламентований порядок відповідності бізнес-правилам мінімізує ризики помилок у подальших операціях.

Для терміну БД, якщо спростити, але при цьому залишити найважливішу суть, то трактування може бути наступним: база даних – це великий обсяг даних, взаємозв'язок між якими побудована за спеціальними правилами. За їх дотримання відповідає СУБД – система (яка базується на програмному та апаратному забезпеченні) для опису, створення, використання, контролю та управління базами даних. [23]

Спроби створити СУБД почалися у 1960-х роках американською електронної компанією International Business Machines Corporation (IBM). У 1968 році IBM представила Information Management System (IMS) – систему управління інформацією. Це впровадження стало важливою віхою у всесвітній технології управління даними.

СУБД – це спеціалізоване ПЗ, призначене для створення, ведення та спільного використання баз даних у середовищі електронно-обчислювальних машин. Вона забезпечує:

- створення, модифікацію та підтримку структури БД;
- введення, редагування, вилучення, пошук і вибірку даних з використанням мов запитів;
- керування транзакціями, забезпечення цілісності даних та

відновлення після збоїв;

− зберігання даних на зовнішніх носіях, їхню оптимізовану організацію для швидкого доступу;

− захист і розмежування прав доступу, підтримку багатокористувацького режиму з одночасним доступом до БД;

− представлення результатів обробки у зручному для користувача вигляді.

У 1970-их роках відбувся прорив у сфері СУБД – запропонували модель, яка відмежовувала логічне представлення даних від фізичного зберігання. Назвою цієї моделі є реляційна модель, яка зробила СУБД більш гнучкими. Перший опис був у новаторській статті "A Relational Model of Data for Large Shared Data Banks" автора, британського інформатика з IBM, Едгара Ф. Кодда. IBM на основі ідей Кодда запустила дослідницький проєкт System R, метою якого було створення прототипу реляційної СУБД. У цьому проєкті був розроблений перший варіант мови запитів SQL (Structured Query Language), яка згодом стала стандартом для всіх реляційних СУБД.

З огляду на те, що саме IBM запровадила структуровану мову запитів, то вважаю за необхідне надати трактування визначенню від імені цієї компанії: структурована мова запитів (SQL) – це предметно-орієнтована стандартизована мова програмування, яка використовується для взаємодії з реляційними системами керування базами даних, такими як MySQL, SQL Server, IBM Db2, PostgreSQL та Oracle Database. [50]

Реляційні БД відповідають набору властивостей ACID:

− Atomicity (атомарність) забезпечує, що всі кроки кожної транзакції в базі даних будуть повністю виконані, або відновлені до початкового стану.

− Consistency (узгодженість) – відповідність даних заздалегідь заданим обмеженням щодо цілісності та бізнес-правил. Навіть якщо кілька користувачів одночасно виконують подібні операції, вони отримують однакові дані.

− Isolation (ізольованість) – ні одна проміжна зміна не стане

видимою за межами транзакції до її завершення, тобто кожна нова транзакція, яка отримує доступ до певного запису, повинна чекати завершення попередньої транзакції, перш ніж розпочати свою роботу. Результат незалежний від процесів, що відбуваються паралельно.

- Durability (довговічність) : усі здійснені транзакції зберігаються в базі даних у разі збою системи. Це означає, що всі зміни, внесені в транзакції ACID, завжди зберігаються і на них не впливають наступні збої системи.

Фундаментальним принципом теорії розподілених систем є CAP-теорема, або теорема Брюера. Вона стверджує, що в будь-якій розподіленій системі можна реалізувати лише дві властивості одночасно, і неможливо одночасно забезпечити три ключові властивості:

- Consistency (узгодженість): усі вузли системи одночасно бачать однакові дані.
- Availability (доступність): система завжди відповідає на запити користувачів, навіть якщо деякі сервери недоступні.
- Partition Tolerance (стійкість до розділення): система може продовжувати функціонувати попри втрату зв’язку між деякими її вузлами.

У зв’язку з обмеженнями, які визначені CAP-теоремою, стало зрозумілим, що звичайні реляційні БД, які функціонують за принципами ACID, не завжди добре будуть проявляти себе у розподілених системах при великому залученні даних. Тому з’явилися нові способи зберігання та обробки інформації, де важливість надійності зменшилася, а доступність та можливість розширення стали основними. Ідея нереляційних БД розглядається як альтернатива традиційним реляційним системам.

Нереляційна БД (NoSQL) – це вид бази даних, де дані зберігаються без суворої структури й явних зв’язків між ними. Різниця від SQL полягає в тому, що інформація може відображатися у вигляді різних документів, включаючи текст, зображення, відео та інше, і не потребує строгого визначення структури даних. [29]

Нереляційні БД функціонують за принципами BASE (Basically Available,

Soft State, Eventual Consistency), які ставлять у пріоритет на доступність та масштабованість у розподілених системах:

- Basic Availability: навіть якщо деякі компоненти недоступні, дані все одно доступні, але вони можуть бути тимчасово застарілими.
- Soft-state: сховища не обов'язково повинні бути узгодженими для запису, а різні репліки не обов'язково повинні бути взаємно узгодженими весь час.
- Eventual consistency: зрештою, узгодженість буде досягнута, але вона не гарантується для окремих транзакцій.

На відміну від цих локальних рішень, розподілені СУБД орієнтовані на складніші завдання з обробки великих обсягів даних. Такі системи дозволяють розміщувати частини БД на кількох серверах, які можуть фізично знаходитися у різних регіонах або навіть країнах. Це забезпечує підвищену відмовостійкість, балансування навантаження та швидший доступ до даних для користувачів у різних місцях.

До потужних представників розподілених СУБД належать Oracle Database, Sybase, Informix, а також сучасні рішення, як-от Microsoft SQL Server і PostgreSQL. Oracle вважається одним із найпотужніших і наймасштабованіших продуктів, що підтримує складні транзакції, реплікацію та високий рівень безпеки даних. Sybase історично використовувалася у банківській сфері та фінансових установах завдяки стабільності та швидкості обробки транзакцій. Informix, у свою чергу, отримала популярність через простоту адміністрування та оптимізовану роботу з об’єктно-реляційними структурами.

Сьогодні до категорії розподілених і високопродуктивних СУБД також відносять MySQL Cluster, MongoDB, Cassandra, які активно застосовуються у вебдодатках, системах аналітики великих даних і хмарних інфраструктурах. Вони поєднують масштабованість, гнучкість структури даних і високу доступність. Таким чином, вибір конкретної СУБД залежить від масштабів підприємства, обсягу даних, технічних вимог і рівня підготовки персоналу.

1.2. Теоретичні основи оптимізації процесів управління базами даних

У сучасній парадигмі бізнес-систем спостерігається інтенсивне збільшення занесення та обробки даних. Динаміка обмежується не лише кількісним зростанням записів, але й характеризується підвищенням структурної складності, поліморфізмом форматів, такими як, до прикладу, реляційними, документо-орієнтованими, часовими рядами й іншими неструктурованими або напівструктурованими сутностями, а також акселерацією темпів оновлення й доступу.

Як наслідок, традиційні архітектурні підходи до СУБД демонструють зниження ефективності та наражаються на кризу масштабованості: фіксується редукція швидкості відгуку, надмірне навантаження на обчислювальні ресурси, і відповідно до цього відбувається зниження рівня якості обслуговування кінцевих користувачів.

Відтак, виникає імперативна потреба у визначенні й систематизації фундаментальних концептів і принципів, які становлять базис для комплексної оптимізації функціонування цих високонавантажених інформаційних систем.

Оптимізація процесів управління БД є комплексним підходом, який охоплює покращення продуктивності, скорочення витрат на обладнання, логічний і структурний дизайн БД, алгоритмічні рішення, адміністрування, використання апаратних ресурсів та здатність системи адаптуватися до змін рівня навантаження. Теоретичні аспекти цієї оптимізації дають змогу визначити межі можливостей, аналізувати різні моделі й підходи, а також формувати критерії для оцінки ефективності.

У даному контексті оптимізація – це сукупність методів і стратегій, які скеровані на покращення рівня продуктивності, ефективності та надійності БД. Ключові завдання:

- зменшення часу обробки запитів,

- підвищення швидкості доступу до даних,
- використання найбільш доцільного способу експлуатації ресурсів,
- забезпечення стабільної, безперебійної й швидкої роботи системи,
- ефективне масштабування системи без зниження продуктивності,
- гнучкість у швидкій адаптації на зміни обсягів даних і кількості користувачів,
- сумісність з гібридними або мультихмарними архітектурними підходами,
- підвищення рівня якості зберігання даних.

Під час оптимізації проводиться глибокий аналіз структури БД, запитів, що виконуються, та налаштувань, щоб знайти проблемні області та виправити їх. Така стратегічність передбачає систематичну ревізію та вдосконалення концептуальної та логічної моделі сховища даних з метою максимізації операційної ефективності його експлуатації. Головна ціль – забезпечити максимально можливу ефективність при мінімальному використанні обчислювальних ресурсів.

Фундаментальний етап оптимізації передбачає визначення ключових сутностей, перевірку коректності зв’язків між ними та впровадження обмежень, спрямованих на забезпечення інформаційної цілісності. Окрім цього, він включає аналіз структури, вибір оптимальних типів даних і методів індексації для прискорення операцій пошуку й сортування. Важливою складовою є також оцінка потенційних вузьких місць продуктивності, розробка механізмів балансування навантаження, налаштування політик кешування, транзакцій і резервного копіювання.

Нормалізація виступає одним з методів оптимізації процесів, сутність якої полягає у послідовній декомпозиції таблиць для досягнення такої структури, де мінімізується інформаційна надлишковість і, як наслідок, попереджаються потенційні аномалії модифікації даних, таких як вставки, оновлення, видалення.

Одночасно, за певних умов, застосовується діаметрально протилежна

стратегія – денормалізація. Вона передбачає свідоме внесення контрольованої надмірності шляхом дублювання даних, що має на меті кардинальне прискорення операцій вибірки і спрощення аналітичних запитів за рахунок зменшення кількості необхідних з’єднань. Саме тому стратегічний вибір між цими двома парадигмами не є довільним, а визначається операційним профілем системи.

Для систем онлайн-обробки транзакцій, де критичною є цілісність й уникнення аномалій запису, пріоритетом є високий ступінь нормалізації. Натомість, в аналітичних системах та сховищах даних, орієнтованих на швидке виконання складних запитів, виправданим є застосування денормалізації задля підвищення продуктивності операцій читання.

Також інструментарієм оптимізації є індексація, яка забезпечує прискорення доступу до даних за допомогою формування спеціалізованих структур пошуку. Раціональна інтеграція індексів дозволяє мінімізувати кількість читання з диску, що критично впливає на латентність виконання запитів. Додатково застосовується розбиття – partitioning, яке розбиває великі таблиці на менші логічні або фізичні компоненти для полегшення виконання запитів та управління обсягами даних.

Впорядкування даних є одним із найважливіших аспектів роботи з БД. Індекси дозволяють СУБД знаходити потрібні записи за частки секунди, не переглядаючи всю таблицю. Це значно прискорює операції, особливо на великих масивах даних. Крім того, БД підтримують зв'язки між таблицями, що дозволяє уникнути дублювання інформації та забезпечити її узгодженість. Наприклад, інформація про клієнта зберігається в одній таблиці, а його замовлення – в іншій, і вони пов'язані унікальним ідентифікатором.

Одним з методів оптимізації управління процесами БД є кешування, що передбачає зменшення навантаження на сервер через зменшення звертань до БД за рахунок використання оперативної пам’яті для тимчасового зберігання даних.

Для оптимізації також важлива реплікація – процес автоматичного

копіювання та підтримки об’єктів БД з одного вузла на один чи більше вузлів-реплік з метою забезпечення доступності, масштабованості та відмовостійкості системи управління БД. Завдяки цьому відбувається зменшення навантаження на основний, підвищується продуктивність і масштабованість через обробку паралельних запитів на різних вузлах, високу доступність даних і мінімізація простою у випадку збою.

Оптимізація процесів управління БД охоплює не лише вдосконалення запитів і структури даних, а й вибір сучасних технологічних підходів до їх зберігання та обробки. У контексті стрімкого зростання обсягів інформації традиційні серверні підходи все частіше поступаються місцем більш адаптивним і динамічним моделям. Одним із перспективних напрямів у цьому аспекті стало запровадження хмарних технологій, які забезпечують високий рівень масштабованості, доступності та продуктивності інформаційних систем.

Використання хмарної інфраструктури для розгортання БД визначається як один із найефективніших способів оптимізації процесів їх управління. DBaaS – це база даних як послуга, що означає хмарну версію бази даних, суть якої полягає в тому, що клієнти отримують доступ до бази даних без необхідності самостійно розгортати та керувати інфраструктурою. DBaaS презентується як керована служба бази даних, тобто постачальник послуг бере на себе відповідальність за налаштування, масштабованість, оновлення і резервне копіювання бази даних. [49]

Очищення даних є необхідним процесом підтримання ефективності та стабільності БД. З часом у системі накопичується значна кількість застарілої, дубльованої або непотрібної інформації, що уповільнює виконання запитів. Регулярний моніторинг і видалення таких записів дозволяє зменшити обсяг таблиць, покращити швидкодію операцій пошуку. Крім того, очищення підвищує точність аналітичних результатів, адже з бази видаляються некоректні або неповні дані. Для цього використовуються спеціальні процедури або скрипти, що автоматично перевіряють і фільтрують

інформацію за заданими критеріями. Важливим аспектом є створення резервних копій перед очищенням, щоб уникнути втрати важливих відомостей.

На сьогодні тему штучного інтелекту важко оминути оскільки все більше інтегрують її у різні сфери, для СУБД це не є виключенням. Сучасні рішення використовують алгоритми машинного навчання для автоматичного налаштування індексів, оптимізації запитів, прогнозування навантаження та виявлення аномалій у даних. Такий підхід сприяє підвищенню продуктивності, адаптивності та надійності БД, особливо в умовах великих і динамічних бізнес-середовищ.

Оптимізація процесів управління БД є багатогранним напрямом, який охоплює як алгоритмічні, так і технологічні підходи до підвищення ефективності роботи систем. Використання хмарних інфраструктур, вдосконалення запитів, балансування навантаження та модернізація апаратного забезпечення формують комплексну основу для стабільного функціонування баз даних. Застосування таких рішень сприяє скороченню часу обробки, зниженню витрат ресурсів і забезпеченню

1.3. Особливості роботи з корпоративними даними й оптимізації процесів управління базами даних у бізнес-компаніях

Для термінів «бізнес», «компанія», «підприємство» існують численні трактування, в залежності від контексту використання. Декілька визначень:

Бізнес – 1. справа, рід заняття, підприємство, яке націлене на отримування прибутку; 2. виробництво, що спеціалізується на виготовленні й випуску конкретних продуктів для подальшого споживача; 3. угода або торгова операція.

Бізнес – це діяльність, що здійснюється суб’єктами-підприємцями та юридичними особами з метою отримання прибутку та не суперечить закону.

Бізнес – ініціативна економічна діяльність, виконувана за рахунок

власних чи позикових засобів на свій ризик і під свою відповідальність, що має на меті одержання прибутку і розвиток власної справи.

Комерційна діяльність – ініціативна, самостійна, виконувана від свого імені, на свій ризик, під свою майнову відповідальність діяльність громадян, фізичних і юридичних осіб, спрямована на одержання доходу, прибутку від користування майном, продажу товарів, виконаних робіт, надання послуг. [23]

Українське законодавство надає наступне визначення терміну підприємство: самостійний суб'єкт господарювання, створений компетентним органом державної влади або органом місцевого самоврядування, або іншими суб'єктами для задоволення суспільних та особистих потреб шляхом систематичного здійснення виробничої, науково-дослідної, торговельної, іншої господарської діяльності в порядку, передбаченому цим Кодексом та іншими законами. [6] Синонімами до цього терміну у зарубіжному комерційному праві є поняття компанія або корпорація.

Компанія – це самостійна юридична особа, що є корпоративним підприємством, яке сформоване фізичним особами. Залежно від того, які цілі хоче досягати компанія, то вона може бути організована різними способами і носити різні найменування, які будуть вказувати на її діапазон юридичної відповідальності. Основною метою компанії є виробництво товарів чи послуг для продажу на ринку за ціною, яка покриває витрати на виробництво, або обслуговування таких компаній.

У сучасному бізнес-ландшафті, що керується даними, наявність високоякісної БД є однією з найважливіших умов успіху в бізнес-секторі. Однією з запорук успішного маркетингу, високого рівня прибутку й ефективного управління взаємовідносин з клієнтами є добре організована і точна БД. Створення якісної бази може значно вплинути на результати бізнесу. БД, створена та налаштована відповідно до конкретних потреб компанії, дозволяє максимізувати цінність бізнесу завдяки оперативній інтеграції й миттєвого доступу до даних. Управління даними настільки важливий бізнес-чинник, що застосовується не лише для забезпечення

отримання, перевірки, зберігання, а й для захисту даних.

Ефективне керування БД залежить від сфери застосування і відповідних вимог до неї. Кожне середовище має власні специфікації порівняно з іншими сферами. Керування БД у комерційному секторі, на відміну від академічних, науково-дослідних чи особистих проєктів, має свої особливості.

По-перше, переважна більшість бізнес-систем, наприклад, таких як систем обробки замовлень, банківські транзакції, CRM, функціонує за моделлю OLTP(Online Transaction Processing). Це характеризується високою інтенсивністю коротких, одночасних транзакцій, активними та паралельними операціями читання та запису, критичною вимогою до паралелізму і цілісності даних. Це кардинально відрізняється від OLAP (Online Analytical Processing) – систем, типових для науки чи аналітики, де домінують рідкісні, але складні та тривалі запити на читання для агрегації великих масивів історичних даних. Оптимізація для OLTP вимагає акценту на мінімізації блокувань, ефективному індексуванні для швидкого запису та пошуку окремих рядків.

По-друге, у комерційному середовищі навіть невеликий період недоступності сервісу безпосередньо конвертується у вимірювані фінансові збитки: втрачені продажі, штрафи за порушення угод про рівень обслуговування, зниження продуктивності персоналу. Це висуває надвисокі вимоги до показників доступності (часто 99.999% і вище), що досягається через складні оптимізовані механізми відмовостійкості та реплікації.

По-третє, бізнес-компанії оперують чутливими даними: персональними даними клієнтів, фінансовою інформацією, комерційною чи банківською таємницею. Це є причиною, чому процеси керування і оптимізації БД повинні суворо відповідати міжнародним та локальним регуляторним нормам, наприклад, GDPR, PCI DSS. Некоректне налаштування безпеки чи процесу резервного копіювання може призвести до витоку даних, що тягне за собою багатомільйонні штрафи та невідворотні репутаційні втрати.

Сучасні бізнес-компанії функціонують у середовищі постійного зростання обсягів даних, що зумовлює потребу у впровадженні ефективних

механізмів управління корпоративною інформацією. Корпоративні БД виступають центральним елементом інформаційної інфраструктури підприємства, забезпечуючи зберігання, узгодженість і доступність критично важливих даних. З огляду на це, для ефективного управління інформаційними даними значна увага приділяється оптимізації процесів роботи з БД. У цьому контексті оптимізація виступає ключовим механізмом підтримання стабільності, безперервності та ефективності обробки великих масивів інформації.

Оптимізація процесів керування БД набуває критичної важливості, безпосередньо впливаючи на операційну ефективність, фінансові показники та конкурентоспроможність компанії. Основна ціль оптимізації керування БД для бізнес-компаній виходить за рамки лише підвищення рівня продуктивності.

Для бізнесу завдання оптимізації також полягає у забезпеченні архітектури, що відповідає вимогам корпорації, і надавані того ступеню функціонування БД, який відповідає візії компанії й її стратегічним бізнес-цілям.

Важливим нюансом щодо оптимізації БД для компанії є те, що необхідно обрати найбільш оптимальний варіант і водночас зважати на обмеженість у ціновій і ресурсній політиці, на який розраховує підприємець. Ефективна стратегія оптимізації має гарантувати безперебійну найкращу БД, що допомагатиме компаніям ухвалювати швидші та точніші рішення.

Відповідно для бізнес-компаній є важливим, щоб оптимізація пройшла за мінімальних витрат, але при цьому гарантувала надійний рівень продуктивності, безпеки й зручності взаємодії.

Важливість впровадження і підтримання оптимізації процесів керування БД для бізнес-компаній демонструють такі пункти/аспекти:

− Швидкість відгуку клієнт-орієнтованих систем (e-commerce, онлайн-банкінг) має пряму кореляцію з рівнем утримання клієнтів та конверсією. Оптимізація запитів та структури БД знижує латентність – час

відгуку, що є критичним для користувацького досвіду.

- Своєчасний доступ до актуальних даних є базисом для роботи внутрішніх систем та аналітичних платформ. Неоптимізовані БД призводять до затримок у формуванні звітності, що нівелює гнучкість компанії в реагуванні на ринкові зміни.
- Ефективне керування БД дозволяє раціонально використовувати апаратні та хмарні ресурси. Оптимізація індексів, запитів та конфігурацій СУБД знижує вимоги до обчислювальних потужностей та дискового простору, що веде до прямої економії капітальних та операційних витрат.
- Оптимізація включає розробку та впровадження ефективних стратегій резервного копіювання, реплікації та аварійного відновлення. Це гарантує високі показники доступності та мінімізує фінансові й репутаційні втрати від потенційних збоїв.

У підприємствах відповідальність за управління БД, моніторинг потреб і процесів оптимізації покладена на позицію адміністратора БД. Мануальний аудит логів виконання є необхідним для своєчасної ідентифікації критичних ділянок коду та превенції надмірного ресурсного навантаження на архітектуру системи навіть за наявності вбудованих оптимізаторів в СУБД.

Однією з ключових особливостей роботи з корпоративними базами даних є застосування механізмів індексації для підвищення швидкодії обробки запитів. Індекси дозволяють мінімізувати час пошуку інформації у великих наборах даних, забезпечуючи ефективний доступ до потрібних записів. У бізнес-середовищі, де кожна секунда відповіді системи впливає на прийняття рішень або клієнтське задоволення, ефективна індексація стає одним із ключових чинників конкурентоспроможності.

Розміщення БД у хмарних середовищах стає не лише технічним, а й стратегічним інструментом підвищення конкурентоспроможності бізнесу. Це дозволяють зменшити час виведення продуктів на ринок, швидко розгортати аналітичні рішення та інтегрувати дані з різних джерел. Гнучкість хмарної архітектури також забезпечує стабільність бізнес-процесів під час

масштабування або реструктуризації організації.

DBaaS відіграє суттєву роль для компаній, що працюють у сфері фінансів, торгівлі чи виробництва, оскільки забезпечує централізований і стандартизований підхід до доступу внутрішніх даних компанії. За допомогою цього інструмента компанія уникає дублювання інформації та починає зберігати її в єдиному середовищі, що означає можливість оперативного прийняття рішень на основі актуальної інформації Ба більше, DBaaS легко інтегрується з різними аналітичними платформами, які вже використовуються на практиці. Така інтеграція допомагає автоматизувати процеси аналізу і звітності.

Для бізнесу впровадження сучасних методів оптимізації є комплексним, високопріоритетним завданням, яке може забезпечити безперервний доступ до даних, швидке прийняття рішень і конкурентну перевагу на ринку. Це виходить за рамки суто технічного адміністрування і стає критичним елементом стратегічного управління, спрямованим на забезпечення стабільності, ефективності та безпеки ключових бізнес-процесів компанії. Саме тому оптимізація процесів адміністрування БД набуває особливого значення для забезпечення стійкості продуктивності корпоративних систем. Вона скерована на раціональне застосування ресурсів, скорочення часу обробки запитів і підвищення дієвості взаємодії між компонентами системи.

РОЗДІЛ 2
АНАЛІЗ ПІДХОДІВ ДО ОПТИМІЗАЦІЇ ПРОЦЕСІВ УПРАВЛІННЯ БАЗАМИ ДАНИХ У БІЗНЕСІ

2.1. Аналіз ключових моделей БД у розрізі оптимізації процесів у компаніях

Серед моделей БД архаїчними вважаються ієрархічна і мережева, які були на піку популярності в період домінування мейнфреймів. Хоча з появою реляційних БД вони втратили провідні позиції, їхні ідеї частково втілилися в сучасних нереляційних підходах.

Графові БД, наприклад, можна розглядати як еволюційне продовження мережевої моделі, що знову набула актуальності для аналізу соціальних мереж і складних систем взаємозв'язків. Завдяки орієнтації на зберігання сутностей та їхніх зв'язків вони забезпечують надзвичайно ефективну обробку операцій, пов'язаних із пошуком та дослідженням шляхів між вузлами. Зростаюча кількість сценаріїв використання, зокрема в рекомендаційних системах та кібербезпеці, робить графові СУБД одним із найбільш перспективних напрямів у сучасній обробці даних.

Ієрархічна структура, у свою чергу, стала основою для формування документо-орієнтованої моделі – нереляційного типу БД, який зберігає інформацію у форматі, подібному до документів, наприклад, JSON або BSON. Такий підхід забезпечує високу гнучкість, дозволяючи додавати нові поля без зміни загальної схеми, що є доцільним для обробки динамічно змінюваних структур даних.

Серед новітніх категорій БД у бізнес-середовищі виділяють real-time databases – БД реального часу, що забезпечують обробку потоків БД затримок. Такі системи орієнтовані на забезпечення негайної доступності оновленої інформації, що є критично важливим у середовищах з високою динамікою змін та потребою в оперативному прийнятті рішень.

У сучасному бізнесі використання таких СУБД стає ключовим для компаній, орієнтованих на оперативну аналітику, фінансовий моніторинг та персоналізовану взаємодію з клієнтами. Зокрема, у FinTech БД реального часу застосовуються для моніторингу транзакцій, у e-commerce для персоналізованих рекомендацій, управлінь запасами; у логістиці для відстеження відправлень у реальному часі.

Сфера зберігання даних переживає період значних змін через зростаючі обсяги, швидкість генерування та різноманітність даних, особливо у хмарних середовищах та при роботі з Big Data. Традиційні реляційні БД й надалі залишаються одним з ключових інструментів, адже вони надають строгість схеми, транзакційну цілісність і типові запити між таблицями. Перевага їхня полягає в надійній транзакційності і притаманній їй ACID-властивості.

Галузь аналізу даних та управління БД значною мірою залежить від SQL, фундаментальної мови програмування, яка використовується для обробки та маніпулювання структурованими даними . SQL є важливим для таких завдань, як отримання, оновлення та керування даними, що зберігаються в базах даних.

На відміну від мов програмування загального призначення, SQL спеціально розроблений для реляційних БД, а реляційні БД, у свою чергу, оптимізовані для SQL. Цей симбіоз робить SQL високоефективним інструментом, який надає допомогу ефективно організовувати, керувати, швидко взаємодіяти з великими наборами даних.

Розуміння SQL має вирішальне значення для кожного, хто працює з базами даних, оскільки воно дозволяє користувачам ефективно взаємодіяти з ними. Декларативний характер SQL дозволяє користувачам зосередитися на логіці запиту, а не на деталях реалізації, оскільки сама СУБД визначає найефективніший спосіб доступу та вибірки даних.

Аналітики даних, фахівці з обробки даних та адміністратори БД регулярно використовують SQL, оскільки він чудово справляється з такими завданнями, як обробка даних, визначення даних, контроль доступу, обмін

даними , інтеграція даних та аналітика великих даних.

На відміну від SQL-систем, NoSQL – група БД, які відходять від традиційної реляційної моделі та не використовують стандартний механізм структурованих запитів SQL. NoSQL орієнтований на обробку неструктурованих та напівструктурованих даних, що робить його гнучким інструментом для сучасних високонавантажених застосунків.

Розуміння NoSQL стає все більш важливим для фахівців у сфері даних, оскільки сучасні застосунки часто поєднують різні типи даних і вимагають високої гнучкості. Аналітики та інженери з даних застосовують NoSQL для зберігання веблогів, обробки великих масивів даних, побудови персоналізованих рекомендаційних систем, роботи зі складними графовими структурами та побудови масштабованих розподілених архітектур (табл. 2.1).

Таблиця 2. 1

Порівняльна таблиця SQL і NoSQL

Аспект	SQL	NoSQL
1	2	3
Модель даних	Структурована, таблична з фіксованими схемами та зв’язками	Гнучкі моделі даних (документ, ключ-значення, стовпчиково-орієнтована, графова)
Схема	Жорстка; повинна бути визначена заздалегідь зі строгою структурою	Безсхемна або гнучка; дозволяє динамічні та неструктуровані дані
Масштабованість	Вертикальна	Горизонтальна
Консистентність	Висока консистентність завдяки транзакціям ACID	Кінцева узгодженість, але може налаштовуватись у деяких системах
Підтримка транзакцій	Повна підтримка ACID	Модель BASE, інколи частково ACID

Закінчення таблиці 2.1

1	2	3
Нормалізація	Підтримує	Підтримує денормалізацію
Додавання даних	Вимагає змін у схемі	Дозволяє без змін у схемі
Вартість	Дорожчий масштаб	Дешевший масштаб
Продуктивність	Оптимізовані для складних запитів і зв’язків	Оптимізовані для високої пропускної здатності читання/запису, особливо у розподілених системах
Виклики масштабування	Важко виконати шардинг або розподіл між серверами	Побудовані для легкого шардингу та розподілу даних
Цілісність даних	Висока; забезпечується жорсткими обмеженнями та зв’язками	Варіюється; часто забезпечується на рівні застосунку
Зв’язки та об’єднання	Підтримує складні об’єднання між таблицями та зовнішні ключові зв’язки	Зв’язки існують або вбудовані, або реалізовані, як наприклад, у графових БД
Використання	Для структурованих даних і застосунків, що потребують сильної консистентності (наприклад, фінансові системи, ERP, CRM)	Для неструктурованих або напівструктурованих даних і великомасштабних розподілених систем (наприклад, Big Data, аналітика в реальному часі, соціальні мережі)
Приклади СУБД	MySQL, PostgreSQL, Oracle, SQL Server	MongoDB, Cassandra, DynamoDB, Redis, Neo4j

Саме на межі підходів SQL і NoSQL у 2011 році був запропонований термін NewSQL, аби позначити нове покоління реляційних СУБД, які намагаються об’єднати транзакційну цілісність і строгі гарантії ACID традиційних SQL-систем із високою продуктивністю та горизонтальним масштабуванням, притаманним NoSQL-рішенню.

Ідея створення NewSQL полягала в тому, щоб об’єднати переваги обох підходів і розробити архітектуру, яка могла б працювати в кластерах із підтримкою реплікації та фрагментацією, зберігаючи знайомі та стандартизовані інтерфейси SQL, схеми та транзакції.

NewSQL – це клас сучасних реляційних СУБД, які прагнуть забезпечити масштабованість, характерну для NoSQL-систем, для навантажень онлайн-транзакційної обробки, зберігаючи при цьому гарантії ACID традиційних БД. NewSQL є еволюційним містком, який дозволяє застосовувати реляційні моделі даних у середовищах з високими вимогами до пропускної здатності та доступності.

Огляд сучасних моделей БД та їхніх гібридних рішень демонструє, що вибір підходу й інструментів стає багатокритеріальною аналітичною задачею для бізнес-компаній, яка вимагає глибокого розуміння вектору розвитку й стану поточної діяльності компанії, характеру навантажень, вимог до консистентності, обсягів даних тощо.

2.2. Індексація як механізм оптимізації продуктивності СУБД

Зі зростанням даних, швидкість доступу до них стає ключовим фактором конкурентоспроможності, підтримки щоденної діяльності компаній ф прийняття рішень. Важливим ключем керування БД, який оптимізує процеси управління, є індексація БД, що прискорює виконання запитів.

Індекси – це спеціалізовані структури даних, які зберігають посилання на записи, тим самим дає змогу знаходити необхідну інформацію за порівняно короткий проміжок часу.

Суть індексація БД полягає у мінімізації необхідності повного сканування таблиці, яке є дорогим за обчислювальними ресурсами та часом.

Без індексів СУБД перевірятиме записи таблиці при повному перегляді по черзі, що триватиме відчутний проміжок часу у випадку великих обсягів даних.

Натомість використання індексів дозволяє звернутися лише до тих рядків, які відповідають критеріям пошуку, що забезпечує значну оптимізацію продуктивності.

До цілей застосування індексів можна віднести:

- Прискорення операцій вибірки і пошуку. Головною метою є зменшити час виконання запитів. Індекс надає можливість швидкому пошуку завдяки тому, що СУБД виконує пару читань таблиці замість повного сканування. Особливо це відчутно при використанні умови WHERE, яка надає чітку умову фільтрації, що особливо важливо для оптимізації процесів роботи з великою кількістю рядків у таблиці.
- Пришвидшення сортування і групування. Якщо зазначати умову сортування ORDER BY чи групування GROUP BY, то СУБД уникне ресурсоємних операцій і зекономить час і простір.
- Забезпечення цілісності даних. Гарантія унікальності кожного рядка, яка реалізується за допомогою первинного ключа, забезпечує швидкий доступ до рядків, які запитуються. Також індексування стовпців, які не є первинними ключами, за частого звертання до них має значно прискорити операції, які пов'язані з перевіркою цілісності, як наприклад, видалення або оновлення рядка таблиці.
- Прискорення операцій об'єднання. Виконання операції JOIN для двох таблиць, які, наприклад, об'єднуються за спільним стовпцем: краще індекс на стовпці, який розташований на меншій з двох таблиць або на зовнішньому ключі, дозволяє СУБД швидко знаходити відповідні рядки, що значно підвищує швидкість виконання запиту. [2525]

Індекси доцільно створювати:

- для унікальних стовпців (unique): створюються автоматично;
- на атрибутах, що часто фігурують у фільтраційних умовах запитів (where);
- на стовпцях, що застосовуються для сортування (order by);
- для об’єднання таблиць (join);
- на стовпцях, що використовуються у фільтрації, групуванні або сортуванні (where, group by, order by);
- для таблиць з великим обсягом даних: у системах із понад 1 млн записів створення індексу на ключових стовпцях може прискорити виконання запитів до 300% [4646];
- на стовпцях із високою кардинальністю: стовпців містять унікальні або майже унікальні значення.

Попри очевидні переваги, індексація даних має низку ризиків та обмежень, які важливо враховувати при управлінні корпоративними системами. По-перше, надмірне використання індексів може призвести до зростання витрат на зберігання, уповільнення операцій вставки, оновлення та видалення, так як кожна зміна даних вимагає оновлення відповідних структур індексів. По-друге, неоптимальне проектування індексів здатне спотворити план виконання запитів, що створить додаткове навантаження на сервер. По-третє, адміністративні витрати можуть збільшитися у зв’язку з регулярним моніторинг і реорганізації індексів.

Для забезпечення високої продуктивності та адаптивності СУБД застосовуються різноманітні підходи до організації індексів. Кожен з них має специфічні переваги й обмеження, які визначаються характером даних та способом їх використання в запитах. Результативність оптимізації безпосередньо залежить від усвідомлення того, як типи індексів взаємодіють із фізичною схемою зберігання даних. Саме цей аспект ґрунтовно впливає на важливість для спеціалістів розрізняти типи індексів.

Кластерний індекс – це тип індексу, який визначає порядок зберігання даних у таблиці, і сортує рядки даних у таблиці лише одним способом: за

їхніми значеннями ключа.

Некластерний індекс – це окрема, фізично відокремлена структура даних, яка створюється для прискорення доступу до даних, що зберігаються у таблиці. Головна особливість полягає в тому, що він не визначає фізичний порядок зберігання рядків у самій таблиці (табл. 2.2).

Таблиця 2. 2

Порівняння кластерного і некластерного індексу

Критерій	Кластерний індекс	Некластерний індекс
Роль	Визначає фізичний порядок зберігання даних у таблиці	Окрема структура даних, що містить ключ й посилання на місце розташування даних
Фізична організація даних	Дані фізично впорядковуються відповідно до індексу	Логічний порядок не відповідає фізичному порядку рядків
Кількість в таблиці	Лише один	Може бути кілька (зазвичай до 999, залежно від СУБД).
Розмір у дисковому просторі	Невеликий, оскільки і є таблицею, за ключем	Вимагає додаткового дискового простору, оскільки є копією ключових стовпців
Оптимальне призначення	Для запитів за діапазоном і при отриманні великої кількості послідовних даних	Для точного пошуку за стовпцем
Механізм доступу до даних	Безпосередньо через структуру індексу, оскільки вона містить самі рядки таблиці, впорядковані за ключем	Спочатку виконується пошук у структурі індексу, потім додаткове звернення до таблиці за вказівником на відповідний рядок

Кластерний індекс у більшості СУБД за замовчуванням –первинний ключ, некластерний може бути створений для будь-яких стовпців, які часто використовуються.

Після розгляду індексів, які класифіковані за способом фізичного розміщення даних, а саме кластерних і некластерних індексів, доцільно перейти до іншого типу класифікації індексів – за структурою організації. Структура індексу визначає механізм алгоритму обробки даних. Це один із визначальних елементів, який забезпечує логічну впорядкованість пошуку, стабільну ефективність запитів та оптимальний доступ до інформації незалежно від обсягів її зберігання.

Кожна сфера вимагає своїх специфікацій, кожна БД є своєрідною, тому серед різноманіття також варто підбирати необхідний для конкретної робити тип. Для ефективного проєктування і оптимізації продуктивності БД потрібно знати кожний тип індексу, їхні переваги і недоліки для коректного вибору, адже це буде впливати на подальшу роботу з БД.

Серед основних типів індексів, що застосовуються для оптимізації запитів у сучасних СУБД:

- B-tree-індекс є найпоширенішим, його структура базується на збалансованому багатодітному дереві, де кожен вузол містить відсортовану множину ключів і посилання на дочірні елементи. Така організація забезпечує логарифмічний час доступу до даних O(logn), що робить пошук, вставку та видалення високоефективними операціями. Перевагами B-tree є впорядкованість ключів, мінімальна кількість звернень до пам’яті та стабільна продуктивність під час вставки й видалення елементів. Водночас цей тип індексу має недолік у вигляді додаткових витрат на оновлення при частих модифікаціях даних і можливого збільшення висоти дерева при нерівномірному розподілі значень.
- B+Tree-індекси є модифікацією попереднього типу, оптимізованою для діапазонних запитів. Усі ключі розташовані у листових вузлах, що полегшує послідовне сканування значень. Внутрішні вузли містять

лише ключі без даних, тому індекс споживає дещо більше пам’яті, але забезпечує підвищену ефективність при пошуку за інтервалами.

- Hash-індекси ґрунтуються на використанні хеш-функцій, які перетворюють ключі у хеш-таблиці для забезпечення прямого доступу до даних. Основною перевагою цього підходу є постійний час пошуку при виконанні точкових запитів. Проте такі індекси не підтримують діапазонні запити та сортування, а також можуть страждати від колізій, що знижує ефективність у випадках нерівномірного розподілу значень.
- GIN (Generalized Inverted Index) використовується переважно для повнотекстового пошуку та роботи з даними типу JSONB у PostgreSQL. Його архітектура побудована на принципі інверсного індексування, коли зберігається відповідність між елементом і множиною записів, що його містять. GIN забезпечує високу швидкість пошуку та підтримує широкий набір операторів, наприклад, @>, <@, ?, ?|, ?&, що робить його ефективним для напівструктурованих і складних типів даних.
- GiST (Generalized Search Tree) – універсальний механізм, який дозволяє створювати користувацькі алгоритми індексації. Його структура представлена у вигляді дерева, де кожен вузол містить межі, що групують схожі значення. Такий підхід забезпечує ефективну підтримку діапазонних, наближених і просторових запитів, зокрема для геоінформаційних систем. Недоліком GiST є відносно повільне оновлення й підвищене споживання пам’яті, що обмежує його використання при великій кількості точкових операцій.
- BRIN (Block Range Index) є компактним типом індексу, який зберігає агреговану інформацію про блоки даних – зокрема мінімальні та максимальні значення в межах сторінок таблиці. Такий підхід суттєво зменшує розмір індексу і дозволяє ефективно працювати з великими послідовними таблицями, де дані впорядковані або мають природну кореляцію з їхнім фізичним розташуванням. Недоліками BRIN є менша точність пошуку та необхідність додаткового сканування блоків при

нерівномірному розподілі даних.

− Bitmap-індекси реалізують представлення даних у вигляді бітових карт, де кожен біт відповідає рядку таблиці. Вони забезпечують високу продуктивність у запитах до полів із низькою кардинальністю (наприклад, статі, категорії, статуси) і широко використовуються в аналітичних системах (OLAP). Перевагою є зменшений час обробки складних логічних запитів і компактність при стисненні. Проте при високій кардинальності або частих оновленнях даних ефективність Bitmap-індексів помітно знижується.

Використання індексації має бути збалансованим процесом, який передбачає ретельний аналіз запитів, частоти оновлень даних і доступних ресурсів. Оптимальний підхід полягає у створенні лише тих індексів, які забезпечують реальне скорочення часу виконання критичних бізнес-запитів, не перевантажуючи систему надлишковими обчисленнями.

2.3. Оптимізація запитів як основа підвищення продуктивності БД

Удосконалення написання SQL-запитів, а саме влучно підібраний варіант видобування даних, є однією з найбільш вирішальних і дієвих складових у загальній схемі налагодження процесів керування даними. Ця робота охоплює модифікацію SQL-коду, конфігурацію СУБД, розробку структур даних з метою зменшення часу відповіді щодо виконання запиту, зниження потреби у використанні системних ресурсах, таких як CPU, I/O, пам'яті.

Якість SQL-запитів має значну вагу, яка безпосередньо впливає на показник латентності відповідей з боку БД. Дієвість переважної частини бізнес-застосунків, особливо які оперують у режимі обробки транзакцій у реальному часі, напряму залежить від затримки відповідей бази даних.

Підґрунтям для апаратної оптимізації є елемент СУБД, відомий як «query optimizer» або «оптимізатор». Це програмне забезпечення аналізує вхідний SQL-запит і генерує множину потенційних «планів виконання»

(execution plans) – послідовностей операцій (сканування таблиць, з'єднання, упорядкування) для отримання результату. Оптимізатор використовує статистичні відомості про розподіл даних у таблицях та індексах для оцінки вартості кожного плану, обираючи той, що подається як найменш витратним з позиції очікуваного використання ресурсів.

Сучасні реляційні СУБД переважно застосовують Cost-Based Optimizer (CBO) – вартісно-спрямований оптимізатор, який прийшов на зміну застарілому оптимізатору на основі правил – Rule-Based Optimizer. На противагу RBO, що дотримувався суворо визначеного набору евристик, CBO формує рішення на базі свіжої статистики: кількість рядків, кардинальність, гістограми розподілу даних. Це дозволяє системі гнучко реагувати на зміни в обсягах та структурі даних, проте водночас накладає великі вимоги щодо своєчасності й точності зібраної статистики, які є ключовим ресурсом для оптимізатора.

Незважаючи на високий ступінь автоматизації, процес вдосконалення SQL-запитів залишається дуалістичним і вимагає залучення адміністратора баз даних.

На першому, логічному рівні оптимізації, інженери зосереджуються на рефакторингу самого SQL-коду, застосовуючи набір фундаментальних принципів, що покращують взаємодію запиту з оптимізатором СУБД, незалежно від фізичної структури даних. Метою є мінімізація обсягу зайвої роботи, що виконує БД.

Одним зі способів змусити надавати лише необхідні дані є відмова від невибіркових запитів на користь явного переліку лише необхідних для бізнес-логіки стовпців, тобто якщо потрібно отримати конкретні критерії, то замість використання «SELECT *» варто вказувати конкретні колонки. Також варто застосовувати фільтрацію якомога раніше, тобто використання секції WHERE для відсіювання рядків до агрегації, замість HAVING, що фільтрує вже агрегований набір.

Критично важливим для предикатів є забезпечення SARG-здатності –

Searchable Argument. Цього можна досягти завдяки уникненню застосування будь-яких функцій до індексованих стовпців у секції WHERE, наприклад, WHERE YEAR(Birth) = 1998.

Також варто відмовитися від використання початкових узагальнюючих символів, таких як LIKE '%value'. Подібні практики унеможливлюють використання індексів оптимізатором.

Логічна оптимізація запитів також передбачає добір найбільш ефективних операторів відповідно до специфіки виконуваних обчислювальних завдань, наприклад, заміну UNION на UNION ALL, коли унікальність результату не є вимогою, що дозволяє уникнути дорогої операції сортування та дедуплікації, або використання EXISTS замість IN при роботі з великими наборами даних у підзапитах.

Для високотранзакційних OLTP-систем параметризація запитів є одним із ключових підходів до підвищення ефективності роботи СУБД. Вона передбачає заміну конкретних значень у SQL-запитах параметрами, що дає змогу системі розпізнавати запити як ідентичні та повторно використовувати вже згенеровані плани виконання. Завдяки цьому зменшується кількість операцій компіляції запитів і, відповідно, навантаження на процесор, що підвищує продуктивність системи в умовах великої кількості транзакцій.

Другий рівень – фізична оптимізація – стосується створення та модифікації структур зберігання даних, насамперед індексів, наприклад, B-tree, Bitmap. Аналіз планів виконання показує, де запит не може використати індекс, наприклад, через функцію в секції WHERE, що робить предикат non-sargable, або де індекс взагалі відсутній для ключових полів фільтрації. Створення правильного індексу дозволяє змінити план виконання з повного сканування таблиці на значно ефективніший пошук за індексом.

Обґрунтування необхідності застосування вищезазначених методів оптимізації у бізнес-компаніях полягає у їхньому прямому та вимірюваному впливі на фінансові й операційні показники.

По-перше, це пряме скорочення операційних витрат. У хмарних

середовищах, де ресурси тарифікуються за споживанням, кожна неефективна операція входу-виходу або цикл центрального процесору конвертується у реальні витрати. Оптимізація запитів після міграції скорочує витрати на хмарну інфраструктур. Це доводить, що тюнінг SQL є не лише технічним, але й фінансовим інструментом оптимізації.

По-друге, це критичний вплив на якість обслуговування клієнтів і внутрішню операційну ефективність. Впровадження структурованої методології SQL-тюнінгу дозволяє значно скорочувати середній час відгуку запитів.

Оптимізація безпосередньо впливає на здатність бізнесу до масштабування та швидкість прийняття управлінських рішень. Завдяки скороченню часу обробки пакетів даних (batch-процесів) компанії отримують можливість частіше оновлювати дані в аналітичних системах, забезпечуючи менеджмент більш актуальною інформацією для прийняття рішень і підвищуючи загальну гнучкість бізнес-процесів.

2.4. Огляд особливостей хмарних рішень в оптимізації роботи з даними

Сучасний розвиток інформаційних технологій значно змінює підходи до зберігання та обробки даних у бізнесі та державних структурах. Однією з ключових тенденцій є впровадження хмарних технологій, які дозволяють забезпечити гнучкість, масштабованість та економічну ефективність інформаційних систем.

Використання хмарних сервісів дозволяє організаціям делегувати завдання, пов'язані з підтримкою та модернізацією апаратної інфраструктури, зовнішньому провайдеру. Це, у свою чергу, дає змогу вивільнити внутрішні ресурси та сконцентрувати зусилля на досягненні профільних бізнес-цілей, а не на обслуговуванні допоміжних технічних систем.

У контексті даної роботи, особливу значущість набувають ті хмарні

рішення, що безпосередньо спрощують процеси роботи з даними. Їх використання дає змогу мінімізувати часові та фінансові витрати, пов'язані з адмініструванням складних систем баз даних.

База даних як послуга, або DBaaS – це хмарна модель обслуговування, яка надає доступ користувачам до програмного забезпечення для БД і використовувати його без придбання та налаштування обладнання, встановлення програмного забезпечення чи самостійного керування системою. [49]

Концептуально, головною метою цієї парадигми хмарних обчислень є повне делегування відповідальності за життєвий цикл бази даних від кінцевого користувача (організації) до хмарного провайдера. Постачальник хмарних послуг бере на себе всі завдання, пов’язані з розгортанням, конфігурацією, моніторингом, резервним копіюванням і відновленням баз даних.

Користувач отримує готове середовище, до якого можна підключитися через вебінтерфейс або API. Тобто замість того, щоб компанія самостійно купувала сервери, встановлювала СУБД та наймала штат адміністраторів, вона отримує базу даних як готову послугу.

Перевагами DBaaS є:

- зниження витрат на IT-інфраструктуру;
- можливість автоматичного або ручного масштабування ресурсів (пам’яті, процесорної потужності, обсягу сховища);
- автоматизація обслуговування: постачальник відповідає за оновлення, моніторинг, резервне копіювання та відновлення даних;
- висока доступність і відмовостійкість;
- прискорене розгортання нових екземплярів БД за лічені хвилини через вебінтерфейс або API.

До недоліків можна віднести:

- залежність від постачальника послуг;
- обмежений контроль над конфігурацією: користувач не має повного доступу до внутрішніх параметрів системи;

- ризики безпеки та конфіденційності: дані зберігаються у зовнішньому середовищі, що потребує додаткового контролю доступу;
- залежність продуктивності від швидкості мережевого з’єднання;
- додаткові витрати при масштабуванні.

Попри наявність певних обмежень, розглянуті переваги хмарних БД свідчать про їхню стратегічну доцільність для сучасних підприємств. Зростання попиту на масштабовані, безпечні та економічно ефективні рішення стимулювало активний розвиток моделі DBaaS.

2.5. Сучасні механізми забезпечення безперервності та захисту даних

Сучасні бізнес-компанії накопичують величезні обсяги даних, що охоплюють фінансові, персональні, комерційні та стратегічні відомості. Захист цих даних стає критично важливим не лише для збереження конфіденційності, а й для забезпечення безперебійної роботи підприємства. Втрата або компрометація може призвести до фінансових збитків, зниження довіри клієнтів та порушення правових норм, зокрема законодавства про захист персональних даних.

Серед найбільш поширених загроз виділяють кіберзлочинність, включно з фішингом, шкідливим програмним забезпеченням та ransomware-атаками, а також внутрішні ризики, пов’язані з несанкціонованим доступом співробітників. Крім того, бізнес стикається з ризиком витоку даних через сторонні сервіси, слабкі паролі, некоректне управління правами доступу та застарілі системи безпеки. Одними з найбільш критичні для СУБД є SQL-ін'єкції , які дозволяють зловмисникам маніпулювати запитами до БД.

Захист даних нового покоління фокусується на кіберстійкості – здатності підприємства не лише запобігати атакам, а й швидко відновлюватися з мінімальними втратами. Цей підхід є кіберорієнтованим, пріоритетом якого є відновлення в першу чергу.

Незмінні знімки – автоматичні створенні копії даних, які не можуть бути

змінені чи видалені зловмисниками протягом заданого період; створення логічно чи фізично відокремлених середовищ, де зберігаються критичні копії даних; автоматизоване відновлення – це ключові елементи кіберстійкості в контексті СУБД.

Одним з найбільш традиційних, але необхідних механізмів забезпечення високої доступності й аварійного відновлення є реплікація. Вона передбачає створення та підтримку актуальних копій даних на різних серверах. Реплікація дозволяє миттєво перемикатися на резервну копію у випадку відмови основного сервера, що мінімізує час простою забезпечує безперервність бізнес-процесів, навіть якщо основна система зазнала кібератаки

Бізнес-компанії впроваджують комплексні політики управління ризиками, що включають оцінку загроз, класифікацію даних та визначення критичних активів. Дотримання міжнародних стандартів безпеки, таких як ISO/IEC 27001, дозволяє систематизувати процеси захисту даних та підвищити рівень довіри партнерів і клієнтів.

За даними Всесвітнього економічного форуму, 91% бізнес-лідерів та експертів з кібербезпеки вважають безперервність бізнесу пріоритетом, що перевищує всі інші кіберризики, і саме тому нове покоління захисту даних орієнтується на швидке та гарантоване відновлення. [36]

2.6. Використання технологій штучного інтелекту в оптимізації управління базами даних

У сучасних інформаційних системах впровадження методів штучного інтелекту викликає інтерес як метод оптимізації процесів БД.

Переваги використання технологій ШІ в управлінні БД полягають насамперед у підвищенні ефективності та точності процесів обробки інформації. ШІ-системи дозволяють автоматизувати завдання, що раніше потребували значних людських ресурсів, наприклад, оптимізація запитів, прогнозування навантаження на сервери та виявлення аномалій у даних.

Використання алгоритмів машинного навчання сприяє адаптивному налаштуванню індексів, підвищенню швидкодії системи та зменшенню кількості помилок у процесі адміністрування. У свою чергу, при правильному налаштуванні це забезпечує більш стабільну роботу інформаційних систем і знижує витрати на технічну підтримку. [44,48]

Актуальним вектором розвитку є інтеграція великих мовних моделей та технологій Generative AI у процеси взаємодії з базами даних. Сучасні підходи реалізують концепцію Text-to-SQL, де завдяки обробці природної мови система здатна трансформувати запит користувача, сформульований звичайною мовою, у синтаксично коректний SQL-код. Це значно знижує поріг входження для бізнес-аналітиків та менеджерів, дозволяючи їм отримувати вибірки даних без залучення технічних фахівців.

Паралельно з цим набуває поширення використання ШІ-асистентів для адміністраторів баз даних та розробників. Інструменти, інтегровані в середовища розробки, наприклад, плагіни для DBeaver, DataGrip або GitHub Copilot, допомагають не лише у написанні рутинного коду, але й у створенні складних збережених процедур, тригерів та функцій. ШІ може пропонувати рефакторинг застарілого коду, виявляти потенційні логічні помилки ще на етапі написання та автоматично генерувати тестові дані для перевірки навантаження, що суттєво прискорює цикл розробки та підтримки БД.

Найвищим рівнем автоматизації на сьогоднішній день є так звані самокеровані БД – це системи, які використовують алгоритми ШІ та машинного навчання для автоматизації процесів, що повністю усуває людський фактор з процесів обслуговування. Система самостійно проводить оновлення, налаштування безпеки, бекапування та масштабування ресурсів залежно від поточного навантаження. Такий підхід змінює роль адміністратора БД з оператора, який виконує рутинні дії, на архітектора даних, який фокусується на моделюванні даних та політиках доступу, довіряючи технічну оптимізацію алгоритмам штучного інтелекту.

Незважаючи на значні переваги впровадження ШІ-підходів для

оптимізації управління БД існують суттєві ризики і обмеження. Насамперед це висока вартість впровадження та потреба у кваліфікованих фахівцях, здатних інтегрувати ШІ-рішення з наявними СУБД. Крім того, алгоритми машинного навчання потребують великих високоякісних обсягів даних для навчання.

Додатковим викликом є невизначеність в обґрунтуванні алгоритмів, а також нормативно-правові, етичні й безпекові виклики, складність пояснення рішень, прийнятих моделлю, що створює ризики для прозорості управління даними. Також надмірна автоматизація може призвести до зниження контролю з боку адміністраторів, особливо у випадках некоректного функціонування алгоритмів. [27]

З огляду на вище перелічені факти, варто наголосити, що хоча використання ШІ в бізнес-процесах стає дедалі популярнішим, його застосування саме в контексті оптимізації процесів управління БД ще не є масовим стандартом і першочерговою потребою. Таку пропозицію компаніям наразі краще розглядати як стратегічний етап розвитку бізнесу.

РОЗДІЛ 3
ПРИКЛАДНІ ДОСЛІДЖЕННЯ Й ІНСТРУМЕНТИ ОПТИМІЗАЦІЇ БАЗ ДАНИХ У КОРПОРАТИВНОМУ КОНТЕКСТІ

3.1. Вибір методів і платформ для керування БД у контексті інформаційної системи підприємства

У різних сферах, особливо пов’язаних з наявністю достатньо великої кількості напівструктурованих чи неструктурованих даних, помітним є збільшення популярності використання NoSQL-технологій. Гнучкість у зберіганні інформації стає альтернативою для обробки великих об'ємів даних, що мають різноманітну структуру, і для сценаріїв, в яких швидкість та горизонтальне масштабування важливіші за строгість схеми.

Згідно з результатами досліджень, які розміщені організацією Global Growth Insight, впровадження NoSQL-систем призвело до покращення продуктивності запитів приблизно на 47 % і зниження загальних витрат на інфраструктуру на 25 %. [41]

NoSQL часто має кращі показники продуктивності для великих наборів даних та у горизонтальному масштабуванні. Більше ніж 70 % глобальних підприємств переходять на NoSQL-системи, щоб підвищити масштабованість, гнучкість даних та продуктивність у мультихмарних інфраструктурах. [41]

Проте SQL-системи забезпечують високу стабільність у традиційних прикладних сферах. Тому для систем із строгими вимогами до цілісності та формальної структури даних SQL лишається стандартом.

Також слід мати на увазі, що вибір між SQL і NoSQL залежить від конкретного випадку: характер даних, обсяг, швидкість змін, вимоги до транзакцій і зв’язків, бюджет, навички команди. Наприклад, у проєктах, де дані змінюються часто або мають складну чи гнучку форму, NoSQL часто є кращим вибором.

Також важлива тенденція: багато організацій використовують гібридні

підходи, поєднуючи SQL та NoSQL у своїй інфраструктурі. Це дозволяє використовувати переваги кожного типу БД там, де вони найбільш доцільні: SQL для критично важливих транзакцій та звітності, NoSQL – для частин системи з великими обсягами або з менш структурованими даними.

Згідно з дослідженням RavenDB: 49% розробників використовують комбінацію реляційних БД і NoSQL, 43,9% – виключно реляційний підхід, і лише 7,2 – нереляційний. Незважаючи на зростаючу популярність баз даних NoSQL і NewSQL, традиційні системи реляційні СУБД, такі як MySQL, PostgreSQL і Microsoft SQL Server, продовжують користуватися стабільним попитом і сильною підтримкою. [43]

NoSQL використовуватиметься для спеціалізованих завдань: БД NoSQL, такі як Redis і MongoDB, мають високий рівень взаємодії з реляційними БД (50 - 70 %), що вказує на те, що вони доповнюють, а не замінюють реляційні БД. Хоча деякі бази даних NoSQL, можуть прийти на заміну традиційним підходам, дані опитування RavenDB чітко показують, що більшість рішень NoSQL не використовуються ізольовано – вони виконують спеціалізовану роль. [43]

Близько 64 % організацій впроваджують документо-орієнтовані і графові БД. що полегшує інтеграцію гібридних хмар та обробку аналітики в режимі реального часу. [41]

Вибір СУБД для бізнес-проєктів засновується з огляду на тип оброблюваних даних, навантаження на систему, вимоги до безпеки, наявність підтримки хмарної інфраструктури, вартість ліцензії та доступність фахівців.

Для оцінки актуальних тенденцій розвитку та використання СУБД доцільно звернутися до аналітичного ресурсу DB-Engines, який здійснює щомісячне оновлення рейтингу популярності СУБД. [31]

Комплексний показник «score» будується на основі частоти пошукових запитів, кількості згадувань у публічних технічних обговореннях, кількості пропозицій роботи і кількості профілів у професійних мережах, у яких згадується система. Найпопулярнішими на листопад 2025 рік є Oracle, MySQL,

Microsoft SQL Server, PostgreSQL, MongoDB, Snowflake, Redis, Databricks, IBM Db2, Elasticsearch (Рис. 3. 1).

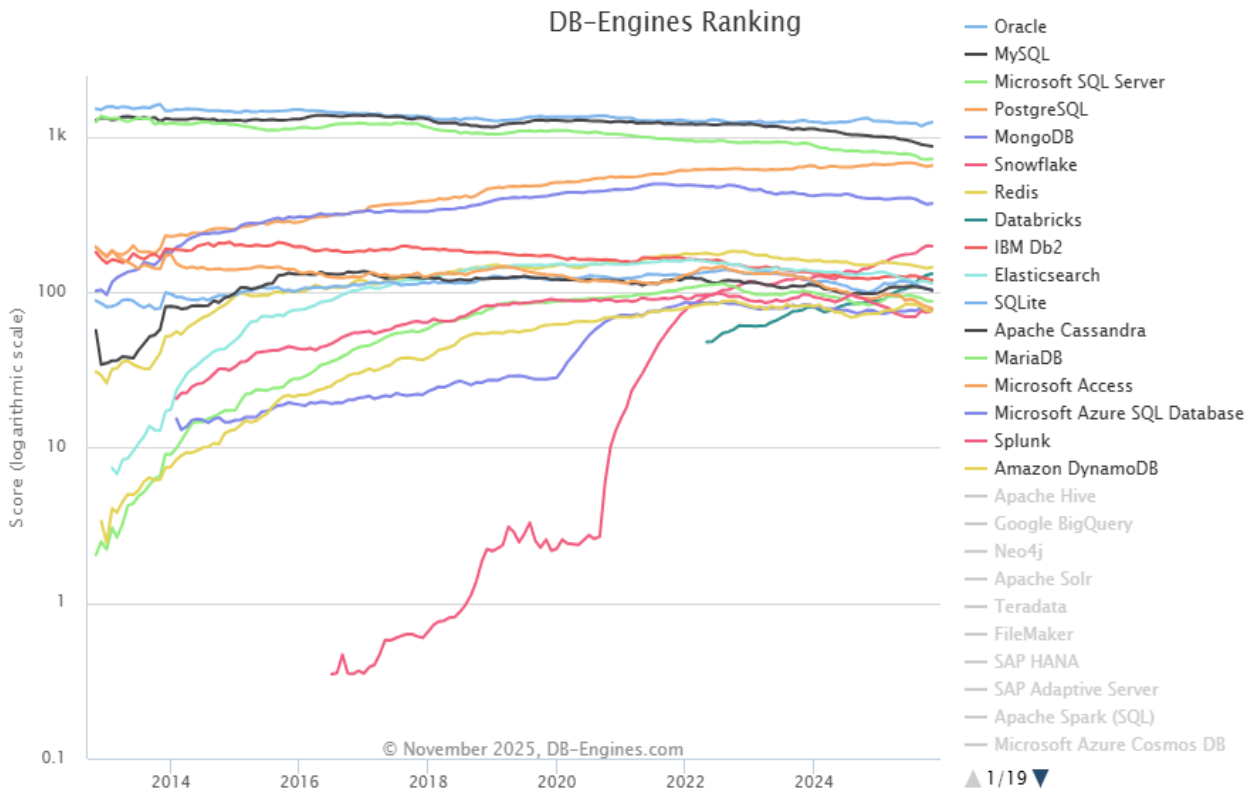


Рис. 3. 1. – Рейтинг популярності СУБД від DB-Engines

За результатами Stack Overflow Developer Survey 2025 (Рис. 3. 2.) можна зробити висновок про поточні тенденції серед професійних розробників. Дослідження демонструє, що найбільш поширеними є реляційні системи: PostgreSQL – 55,6 % і MySQL – 40,5 %. Серед нереляційних рішень суттєве поширення мають Redis і MongoDB. Наведені показники відображають не ринкову частку, а частку розробників, що працювали з відповідною технологією протягом останнього року, що дозволяє оцінити популярність інструментів у професійній спільноті.

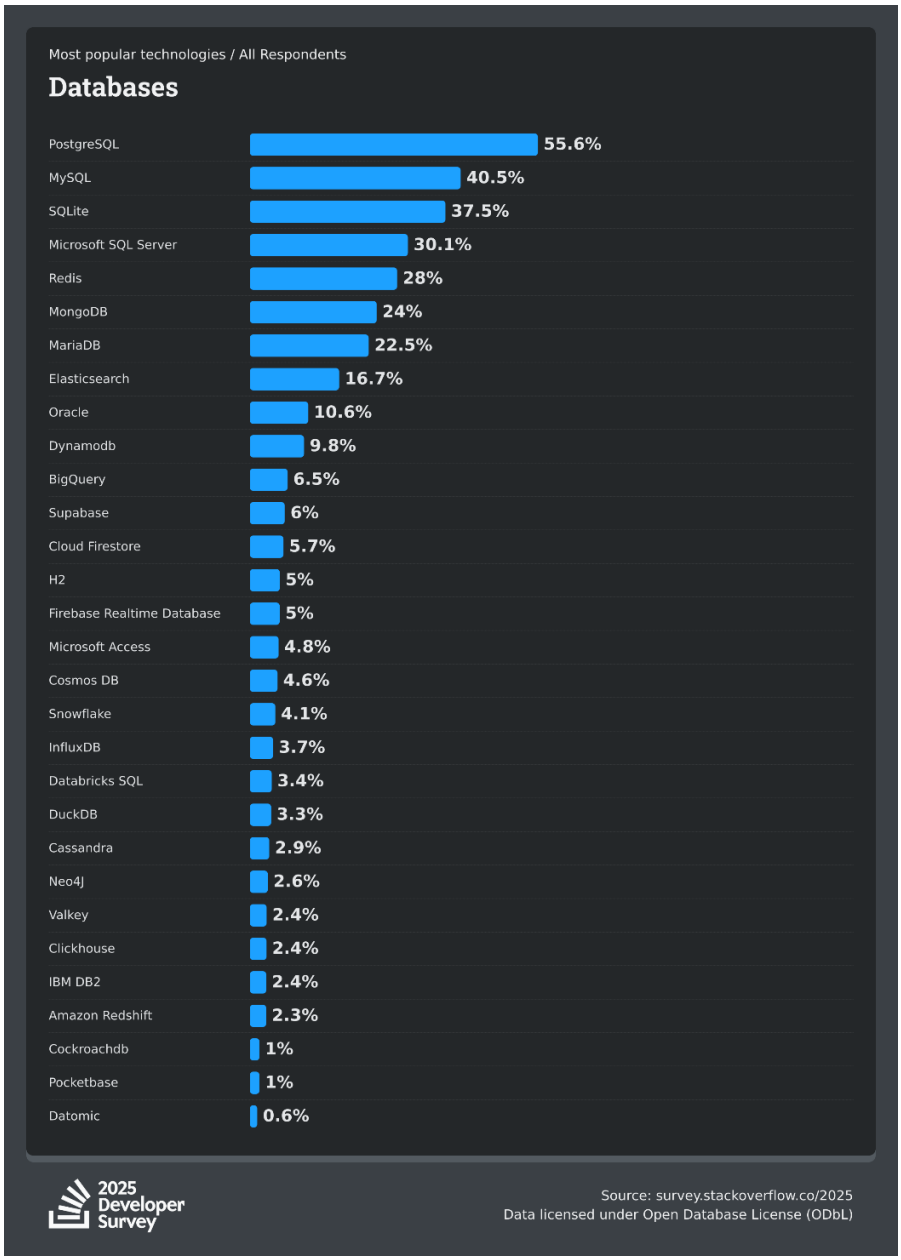


Рис. 3. 2. – Рейтинг СУБД від Stack Overflow Developer Survey 2025

3.2. Практичний аналіз впливу індексації на продуктивність запитів у системах управління базами даних

Індексація є одним із базових механізмів, який дозволяє значно скоротити час доступу до даних і зменшити навантаження на обчислювальні ресурси. У сучасних інформаційних системах індекси розглядають як невід’ємний компонент структури даних, який впливає на загальну продуктивність, стабільність та масштабованість бази даних.

Наприклад, у PostgreSQL для таблиці contragent з близько 80000 записів додано B-tree індекс для поля identifycode. Час виконання точкового запиту з вказаним полем становив приблизно 0,098 мс (Рис. 3. 3.). Значення cost (0.42..8.44) показує очікувану вартість виконання Index Scan, де нижча межа відповідає ініціалізації оператора, а верхня – загальному прогнозованому навантаженню.

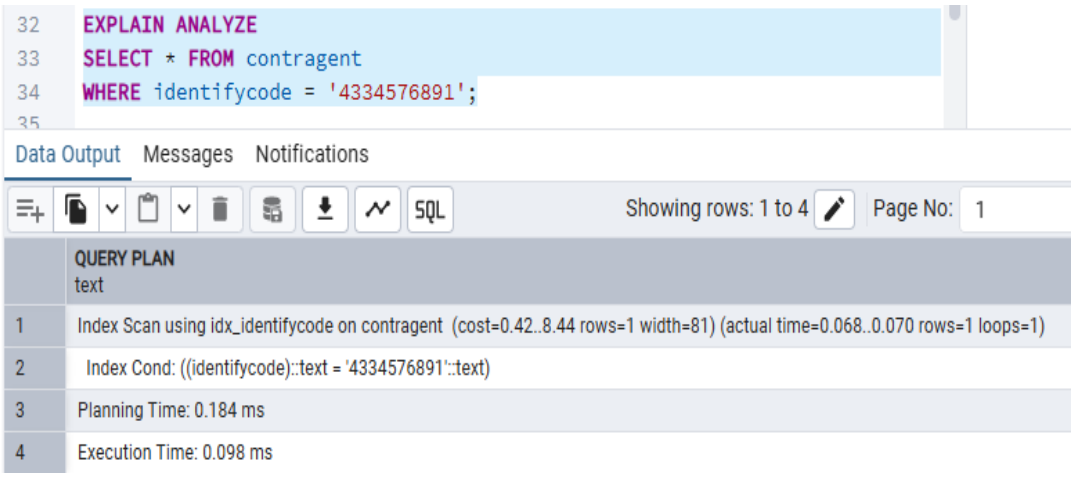


Рис. 3. 3. – План виконання запиту з застосуванням індексу для таблиці обсягом приблизно 80000 записів

Після збільшення таблиці до 900 000 рядків Index Scan залишився швидким: час виконання дорівнював 0.102 мс (Рис. 3. 4.).

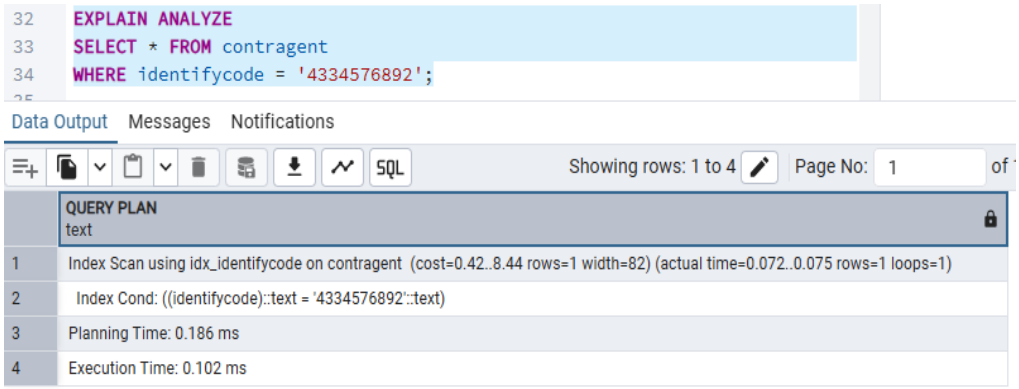


Рис. 3. 4 – План виконання запиту з застосуванням індексу для таблиці обсягом 900000 записів

Це підтверджує властивість B-tree індексу: логічний порядок значень підтримується навіть при значному збільшенні обсягу даних, а фізичне розташування рядків (Рис. 3. 5.) у таблиці не впливає на швидкість точкового пошуку.

```
39  SELECT identifycode, ctid
40  FROM contragent
41  WHERE identifycode IN ('4334576891','4334576892')
42  ORDER BY identifycode;
```

Data Output | Messages | Notifications

Showi

	identifycode character varying (5000)	ctid tid
1	4334576891	(829,10)
2	4334576892	(835,106)

Рис. 3. 5. – Розташування запитуваних даних у таблиці contragent

Після видалення індексу запит виконувався як паралельне послідовне сканування таблиці. У результаті час виконання показав 539.082 мс (Рис. 3. 6.). Це означає, що у PostgreSQL здійснилася перевірка кожного рядка у таблиці. Навіть за використання двох паралельних робітників час виконання збільшився у понад 5000 разів, ніж при використанні індексу.

```
32  EXPLAIN ANALYZE
33  SELECT * FROM contragent
34  WHERE identifycode = '4334576892';
```

Data Output | Messages | Notifications

Showing rows: 1 to 8 | Page No: 1 of 1

	QUERY PLAN text
1	Gather (cost=1000.00..16213.89 rows=1 width=82) (actual time=39.998..539.049 rows=1 loops=1)
2	Workers Planned: 2
3	Workers Launched: 2
4	-> Parallel Seq Scan on contragent (cost=0.00..15213.79 rows=1 width=82) (actual time=135.347..289.658 rows=0 loops=3)
5	Filter: ((identifycode)::text = '4334576892'::text)
6	Rows Removed by Filter: 326835
7	Planning Time: 1.070 ms
8	Execution Time: 539.082 ms

Рис. 3. 6. – План виконання запиту без використання індексу для таблиці обсягом 900000 записів

Даний експеримент демонструє, наскільки індексація критично важлива для швидкого доступу до даних у великих таблицях, і наскільки індекси є ефективним інструментом оптимізації запитів, де важливі швидкість і продуктивність пошуку.

Індекси суттєво збільшують швидкість SELECT-операцій, однак супроводжуються зростанням витрат на INSERT, UPDATE та DELETE.

У системах із високою частотою читання індекси є критично необхідними, тоді як у системах з інтенсивним оновленням їх кількість має бути мінімальною.

Архітектурні нюанси реалізації індексів можуть стати вирішальним фактором при виборі СУБД. Практичним підтвердженням цього є приклад компанії Uber, яка здійснила міграцію зі PostgreSQL до MySQL.

Причиною цього рішення були особливості архітектури індексів і механіка запису змін. У PostgreSQL всі індекси вказують на фізичне розташування рядків (ctid), і під час оновлення рядка ctid змінюється, що призводить до того, що всі індекси повинні бути оновлені, навіть якщо змінюється лише один стовпець.

Для Uber оновлення рядків є частою практикою, тому це стало серйозною проблемою. Через зростаючий розмір індексів збільшувались затрати на зберігання та підтримку, а також страждала продуктивність запитів.

Перехід на MySQL дозволив зменшити ці витрати, оскільки у MySQL вторинні індекси зберігають ключ первинного ключа, а не вказівник на фізичне розташування рядка. При оновленні у цьому середовищі змінюєтьсяж2 тільки той індекс, який індексує змінений стовпець, що знижує накладні витрати на оновлення індексів.[51]

З огляду на цей відомий кейс, можна зауважити, наскільки важливо для забезпечення високої продуктивності звертати увагу не лише на вибір типів індексів і їх кількість, а й на такі тонкощі як логіку реалізації індексних структур в обраній СУБД.

3.3. Порівняльний аналіз підходів до оптимізації SQL-запитів у реальних сценаріях

Оптимізація запитів є одним із ключових етапів підвищення ефективності роботи БД. Вона спрямована на зменшення часу виконання запитів, зниження навантаження на процесор та мінімізацію кількості операцій введення-виведення. Завдяки використанню методів логічного та фізичного тюнінгу забезпечується раціональне використання системних ресурсів і підвищується продуктивність СУБД у цілому. У процесі оптимізації застосовуються інструменти аналізу планів виконання, оцінки вартості операцій і вибору найефективніших шляхів доступу до даних.

Для перевірки навантаження, яке спричиняє вибір усіх стовпців замість вказування конкретних, було проведено дослідження двох SQL-запитів, що звертаються до однієї таблиці DEAL у середовищі Oracle Database. Для обох варіантів було згенеровано плани виконання за допомогою команди. Перший запит вибирав усі атрибути, використовуючи «SELECT *» (Рис. 3. 7.), тоді як другий – лише необхідні колонки – «SELECT id, inputdate» (Рис. 3. 8).

```
explain plan for select * from deal
select * from table(DBMS_XPLAN.DISPLAY);
```

	PLAN_TABLE_OUTPUT
1	Plan hash value: 2996231276
2	
3	----------
4	| Id | Operation | Name | Rows | Bytes | Cost (%CPU)| Time |
5	----------
6	| 0 | SELECT STATEMENT | | 29837 | 4370K| 141 (1)| 00:00:01 |
7	| 1 | TABLE ACCESS FULL| DEAL | 29837 | 4370K| 141 (1)| 00:00:01 |
8	----------

Рис. 3. 7 – План виконання запиту з використанням SELECT *

```
explain plan for select id, inputdate from deal
select * from table(DBMS_XPLAN.DISPLAY);
```

	PLAN_TABLE_OUTPUT
1	Plan hash value: 2996231276
2	
3	----------
4	| Id | Operation | Name | Rows | Bytes | Cost (%CPU)| Time |
5	----------
6	| 0 | SELECT STATEMENT | | 29837 | 378K| 140 (0)| 00:00:01 |
7	| 1 | TABLE ACCESS FULL| DEAL | 29837 | 378K| 140 (0)| 00:00:01 |
8	----------

Рис. 3. 8. – План виконання запиту з використанням SELECT з конкретними колонками

Обидва запити виконуються за допомогою TABLE ACCESS FULL – повного сканування таблиці, тобто система зчитує усі рядки без використання індексів. Проте спостерігається суттєва різниця у показнику Bytes – обсязі даних, що зчитуються. У запиті з використанням SELECT * цей показник становить 4,37 МБ, тоді як при виборі лише двох колонок – лише 378 КБ.

Це підтверджує, що вибір лише необхідних полів у запиті є ефективнішою практикою з точки зору споживання ресурсів введення-виведення і навантаження на підсистему пам’яті. Незважаючи на те, що загальний Cost та орієнтовний час виконання залишаються близькими, при масштабуванні на великі обсяги даних або при багатокористувацькому доступі така оптимізація може знизити загальне навантаження на СУБД.

Уникнення символів-замінників на початку шаблонів LIKE дозволяє використовувати індекси та покращити продуктивність відповідних запитів у середньому на 67%.

Застосування EXISTS замість IN для підзапитів у великих наборах даних, понад 500000 рядків, дозволяє скоротити час виконання у середньому до 30%. [39] Схожий результат дослідження фігурує у роботі «Impact of Database Migration on Application Performance : A Case Study of Database Migration from AWS to GCP», зазначається пришвидшення до 41%. [35]

Застосування для таблиць з меншою кількістю рядків ефективнішим буде протилежний варіант. Так, наприклад, для таблиць AACCOUNT, яка містить 60966 рядків (Рис. 3. 9.), і CONTRAGENT з 4694 рядками (Рис. 3. 10.), було зафіксовано протилежну ситуацію.

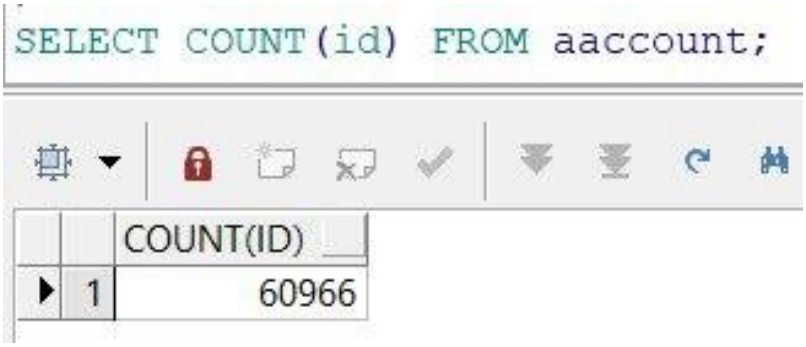


Рис. 3. 9. – Кількість записів у таблиці aacount

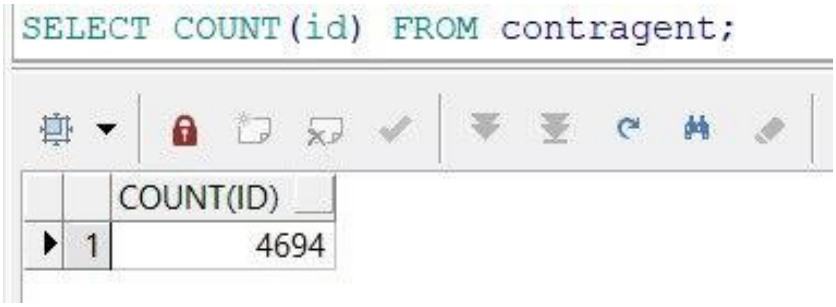


Рис. 3. 10. – Кількість записів у таблиці contragent

У випадку з IN використовується оператор NESTED LOOPS, що під час обробки кожного рядка таблиці AACCOUNT викликає унікальний індекс PK_CONTRAGENT для підтвердження існування відповідного запису. Обсяг оброблюваних даних становив лише 196 КБ. Загальна вартість запиту – 419 одиниць, що відносно невелике значення для тестової вибірки (Рис. 3. 12).

```
explain plan for SELECT a.ID, a.accountno, a.currencyid FROM AACCOUNT a
WHERE contragentid IN (
    SELECT 1
    FROM contragent c
    WHERE c.id = a.contragentid
    AND  SITEID = 300003
);

select * from table(DBMS_XPLAN.DISPLAY);
```

	PLAN_TABLE_OUTPUT
1	Plan hash value: 3000094621
2	
3	----------
4	| Id | Operation | Name | Rows | Bytes | Cost (%CPU)| Time |
5	----------
6	| 0 | SELECT STATEMENT | | 5297 | 196K| 419 (1)| 00:00:01 |
7	| 1 | NESTED LOOPS | | 5297 | 196K| 419 (1)| 00:00:01 |
8	|* 2 | INDEX UNIQUE SCAN| PK_CONTRAGENT | 1 | 10 | 1 (0)| 00:00:01 |
9	|* 3 | TABLE ACCESS FULL| AACCOUNT_INT | 5297 | 144K| 418 (1)| 00:00:01 |
10	----------
11	
12	Predicate Information (identified by operation id):
13	----------
14	
15	2 - access("C"."ID"=1 AND "SITEID"=300003)
16	3 - filter("A"."CONTRAGENTID"=1)

Рис. 3. 11. – План запиту з використання оператора IN

Натомість у запиті з EXISTS оптимізатор застосовує NESTED LOOPS SEMI, який припиняє пошук у підзапиті після першого знайденого збігу, що є більш оптимальною семантикою для перевірки факту існування. Незважаючи на це, значення вартості виконання залишаються майже однаковими: вони варіюються між 419 і 423, оскільки в обох випадках доступ до основної таблиці здійснюється через повне сканування AACCOUNT_INT, а ключовим елементом продуктивності виступає унікальний індекс у таблиці CONTRAGENT. Обсяг оброблюваних даних для цього запиту становив більше 2 МБ (Рис. 3. 12.).

```
explain plan for SELECT a.ID, a.accountno, a.currencyid
FROM AACCOUNT a
WHERE EXISTS (
    SELECT 1
    FROM contragent c
    WHERE c.id = a.contragentid
      AND c.siteid = 300003
);
```

	PLAN_TABLE_OUTPUT
1	Plan hash value: 1315066697
2	
3	----------
4	| Id | Operation | Name | Rows | Bytes | Cost (%CPU)| Time |
5	----------
6	| 0 | SELECT STATEMENT | | 60901 | 2259K| 423 (2)| 00:00:01 |
7	| 1 | NESTED LOOPS SEMI | | 60901 | 2259K| 423 (2)| 00:00:01 |
8	| 2 | TABLE ACCESS FULL| AACCOUNT_INT | 60901 | 1665K| 419 (1)| 00:00:01 |
9	|* 3 | INDEX UNIQUE SCAN| PK_CONTRAGENT | 4639 | 46390 | 0 (0)| 00:00:01 |
10	----------
11	
12	Predicate Information (identified by operation id):
13	----------
14	
15	3 - access("C"."ID"="A"."CONTRAGENTID" AND "C"."SITEID"=300003)

Рис. 3. 12. – План запиту з використання оператора EXISTS

Аналіз показує, що при однаковій логічній умові фільтрації оператор IN виявився економнішим у порівнянні з прогнозованими 2259 КБ у запиті з EXISTS. Така різниця свідчить про суттєво менше навантаження на процес I/O та підтверджує потенційно ефективніше виконання варіанту з IN у даному сценарії. Практичний ефект оператора EXISTS залежить від обсягу зовнішньої таблиці та селективності підзапиту, що слід враховувати при оптимізації запитів у реальних умовах.

Таким чином, для даного набору даних та структури індексів продуктивність IN і EXISTS є практично ідентичною. Причиною швидшого виконання запиту з оператором IN є кількість даних, яка не перевищує навіть 100000 записів. Однак EXISTS на більших наборах даних або складніших умовах фільтрації буде мати перевагу, оскільки забезпечить більш оптимальний шлях виконання завдяки семантиці напівз’єднання.

Параметризовані запити є ефективним інструментом оптимізації в системах з високим обсягом транзакцій: вони дозволяють СУБД повторно використовувати плани виконання, знижуючи накладні витрати на парсинг і

компіляцію, що, за даними деяких досліджень, може зменшувати навантаження на процесор приблизно на 23 %. [35]

Рекомендація у контексті параметрів – це розбиття на партиції. Даний маневр дозволяє фізично розділити таблицю на частини, що пришвидшує доступ до даних. Користувач у разі можливості має розділяти на частини запитуваний параметр і уточнювати його. Наприклад, якщо потрібно відібрати не всі записи, а ті, які стосуються певного періоду дат, типів тощо (Рис. 3. 13).

```
explain plan for
SELECT * FROM deal
WHERE dealdate >= TO_DATE('01.02.2012', 'dd.mm.yyyy')
      AND dealdate < TO_DATE('01.01.2015', 'dd.mm.yyyy')
ORDER BY id

select * from table(DBMS_XPLAN.DISPLAY);
```

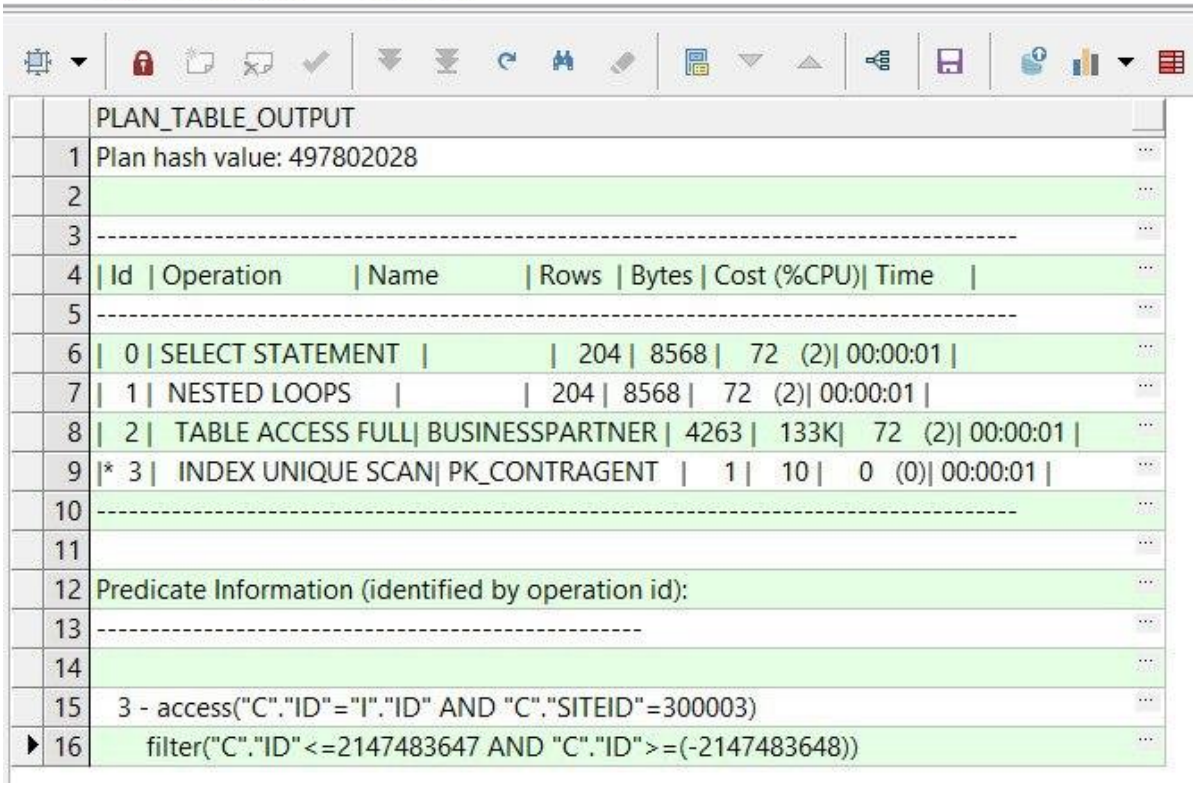
	PLAN_TABLE_OUTPUT
1	Plan hash value: 3910486994
2	
3	----------
4	| Id | Operation | Name | Rows | Bytes | Cost (%CPU)| Time |
5	----------
6	| 0 | SELECT STATEMENT | | 11 | 1540 | 8 (13)| 00:00:01 |
7	| 1 | SORT ORDER BY | | 11 | 1540 | 8 (13)| 00:00:01 |
8	| 2 | TABLE ACCESS BY INDEX ROWID BATCHED| DEAL | 11 | 1540 | 7 (0)| 00:00:01 |
9	|* 3 | INDEX RANGE SCAN | IN_DEAL_DEALDATE | 11 | | 2 (0)| 00:00:01 |
10	----------
11	
12	Predicate Information (identified by operation id):
13	---
14	
15	3 - access("DEALDATE">=TO_DATE(' 2012-02-01 00:00:00', 'syyyy-mm-dd hh24:mi:ss') AND
16	"DEALDATE"<TO_DATE(' 2015-01-01 00:00:00', 'syyyy-mm-dd hh24:mi:ss'))

Рис. 3. 13. – План виконання запиту з уточненими параметрами

До використання оператора JOIN варто підходити з розумінням побудови таблиць. Об’єднання користується широким попитом, однак при недостатньо коректних налаштуваннях, кожне використання JOIN може наражати на сповільнення часу обробки запитів. Об’єднувати варто лише ті таблиці, які дадуть вичерпний кінцевий результат (Рис. 3. 14).

```
select * from table(DBMS_XPLAN.DISPLAY);

explain plan for
SELECT i.id, i.NAME
FROM businesspartner i
JOIN contragent c
      ON c.id = i.id
     AND c.siteid = 300003
```



	PLAN_TABLE_OUTPUT
1	Plan hash value: 497802028
2	
3	----------
4	| Id | Operation | Name | Rows | Bytes | Cost (%CPU)| Time |
5	----------
6	| 0 | SELECT STATEMENT | | 204 | 8568 | 72 (2)| 00:00:01 |
7	| 1 | NESTED LOOPS | | 204 | 8568 | 72 (2)| 00:00:01 |
8	| 2 | TABLE ACCESS FULL| BUSINESSPARTNER | 4263 | 133K| 72 (2)| 00:00:01 |
9	|* 3 | INDEX UNIQUE SCAN| PK_CONTRAGENT | 1 | 10 | 0 (0)| 00:00:01 |
10	----------
11	
12	Predicate Information (identified by operation id):
13	----------
14	
15	3 - access("C"."ID"="I"."ID" AND "C"."SITEID"=300003)
16	filter("C"."ID"<=2147483647 AND "C"."ID">=(-2147483648))

Рис. 3. 14. – План виконання запиту з застосуванням JOIN

Крім того, заміна оператора UNION на UNION ALL у звітних запитах часто покращує продуктивність шляхом уникнення дорогої операції видалення дублікатів. У системах, що генерують великі набори даних, це дає значний приріст продуктивності.

Як підтвердження цього пункту можна звернутися до, наприклад, роботи «Impact of Database Migration on Application Performance : A Case Study of Database Migration from AWS to GCP», у якій повідомляється, що запит, побудований з урахуванням цієї поради, пришвидшується у виконання близько на 31 %. [35]

3.4. Хмарні сервіси як інструмент оптимізації управління базами даних у бізнесі

Сучасний ринок пропонує широкий вибір готових рішень у форматі DBaaS. Підприємства частіше схиляються до хмарних рішень через бажання зменшити накладні витрати на управління базами даних і перевагами хмарної масштабованості та надійності.

Згідно з результатами опитування 451 Research «Voice of the Enterprise: Data & Analytics, Data Platforms 2022» впровадження хмарних технологій та міграція в хмару залишається трендом серед підприємств, особливо для операційних та аналітичних БД. 46% респондентів опитування використовують локальну приватну хмару, 27% використовують пропозицію керованої DBaaS.Тим часом приблизно 60% респондентів називають публічну хмару місцем розгортання своїх систем платформ даних, тоді як приблизно 40% кажуть, що їхні середовища платформ даних працюють у локальному, нехмарному середовищі. [30]

За результатами проведених аналітичних досліджень 2025 року [29] високі позиції в рейтингу найкращих DBaaS для бізнес-компаній (табл. 3.1):

Таблиця 3. 1

Рейтинг DBaaS

Назва	Особливості	Приклади використання
1	2	3
Amazon RDS (AWS)	Вирізняється підтримкою великої кількості реляційних СУБД як власних, так і сторонніх, масштабованістю і надійністю без потреби у керуванні серверним середовищем	Новинний ресурс Dow Jones завдяки AWS збільшив швидкість розробки продуктів щонайменше на 30%. [32] Airbnb мігрувала свою основну базу MySQL до Amazon RDS з downtime всього 15 хвилин. [28]

Закінчення таблиці 3.1

1	2	3
Google Cloud SQL	Дозволяє масштабувати БД під час високих навантажень без потреби в локальних апаратних рішеннях.	Для YOHO, ринок електроніки, перехід на Cloud SQL дозволив обробляти значно більшу кількість транзакцій і уникнути простоїв. [52]
Azure SQL Database	Особливістю є серверлес-модель із гнучкою оплатою «pay-as-you-go», інтеграція з іншими сервісами Microsoft.	У результаті використання цього інструменту компанія Inlitix знизила витрат на середовище розробки і тестування у 10 разів. [37]
MongoDB Atlas	Хмарне рішення для документно-орієнтованих БД, висока продуктивність на великому обсязі запитів, можливість оптимізації кількості шардів.	Для європейської e-commerce компанії Idealo міграція на цей продукт зменшила кількість шардів зі 25 до 12, знизила розмір вузлів на 66 %, встановило підтримку до 200 000 запитів за секунду.[35] Vainu, фінський B2B-сервіс, зменшив щомісячні витрати на інфраструктуру на 14000 євро. [40]
Snowflake	Хмарне сховище даних з аналітичними можливостями, часто класифікується як DBaaS/DWaaS; інтеграція з AI/ML сервісами для прогнозування.	First Tech Federal Credit Union: використала Snowflake для створення цілісного погляду на аналітику й об’єднання даних. [33] US Foods з понад 300000 клієнтами використовує для аналітики продажів і замовлень, прогнозних моделей, зменшення відтоку клієнтів. [47]

Реальні приклади використання цих технологій у відомих компаніях демонструють практичну ефективність DBaaS у підвищенні продуктивності, зниженні витрат і забезпеченні безперервної роботи інформаційних систем

Водночас важливою частиною успішного впровадження DBaaS є його економічна перевага: система оплати за фактичне використання ресурсів дозволяє знизити витрати на підтримку інфраструктури та персоналу, що робить цей сервіс привабливим як для нових компаній, так і для великих корпорацій, які бажають оптимізувати свої управлінські процеси. Однак існують і ризики у впровадженні DBaaS: наприклад, потенційні затримки при нестабільному з’єднанні, залежність від дистриб'ютора хмарного сервісу. Також важливо усвідомлювати, що провайдеру передається відповідальність щодо зберігання персональних даних.

ВИСНОВКИ

В рамках виконання кваліфікаційної магістерської роботи було досліджено актуальний стан і ключові напрямки тенденцій у сфері оптимізації процесів управління базами даних бізнес-компаній.

У магістерській кваліфікаційній роботі здійснено комплексне дослідження підходів до оптимізації процесів управління базами даних у бізнес-компаніях, що дало змогу системно проаналізувати сучасні моделі зберігання даних, механізми підвищення продуктивності СУБД та інструменти забезпечення безперервності функціонування інформаційних систем.

По-перше, було розглянуто теоретичні основи управління даними, їхню роль у корпоративних інформаційних системах. Узагальнено теоретичні підходи до оптимізації процесів керування даними. Також проаналізовано специфіку функціонування бізнес-компаній і важливість застосування оптимізаційних механізмів, які забезпечують стабільність і високу продуктивність БД.

По-друге, проведено розширений аналіз моделей БД з оцінкою їх придатності до використання в умовах різної бізнес-логіки і навантаження. Досліджено можливості індексації, підходи до оптимізації запитів, особливості застосування хмарних платформ, а також сучасні методи забезпечення кіберстійкості, відмовостійкості та захисту даних. Окрему увагу приділено використанню технологій штучного інтелекту, які поступово стають ключовим інструментом автоматизації адміністрування та прогнозування стану СУБД.

У третьому розділі продемонстровано практичні рекомендації щодо оптимізації процесів управління БД у бізнес-компаніях: структурування даних, вибору моделі БД відповідно до бізнес-вимог, побудови індексів, удосконалення запитів, реального застосування хмарних сервісів. У межах дослідження також враховано актуальні аналітичні рейтинги популярних

СУБД та сервісів DBaaS, які відображають тенденції розвитку ринку та допомагають визначити оптимальні технологічні рішення для бізнес-компаній.

Запропоновані підходи мають прикладний характер і можуть бути використані при модернізації існуючих систем або створенні нових корпоративних рішень. Результати дослідження мають значну цінність для практики управління корпоративними інформаційними системами, оскільки дозволяють підвищити ефективність роботи БД, оптимізувати використання ресурсів та забезпечити стабільність функціонування бізнес-процесів у динамічному ринковому середовищі.

Подальше дослідження може бути спрямоване на вивчення нових технологій оптимізації, удосконалення методів аналізу продуктивності та більш поглибленому застосуванні інтелектуальних рішень для автоматизації керування базами даних.

Мету кваліфікаційної магістерської роботи, а саме дослідження й аналіз сучасних тенденцій й інструментів оптимізації управління базами даних у бізнес-компаніях, було досягнуто шляхом послідовного аналізу теоретичних засад управління базами даних, дослідження сучасних підходів до їх оптимізації та застосуванні практичних рекомендацій, спрямованих на підвищення продуктивності та надійності корпоративних СУБД.

Результати проведеного дослідження засвідчили, що комплексний підхід до впровадження сучасних методів оптимізації процесів управління базами даних не лише надає перевагу у прискоренні обробки даних, а й є необхідною умовою для забезпечення стабільності роботи, зменшення операційних витрат і підвищення якості управлінських рішень. Такий підхід формує основу для ефективного масштабування, зміцнення конкурентних позицій та стійкого розвитку підприємства в умовах цифрової трансформації.

ПЕРЕЛІК ПОСИЛАНЬ

1. Закон України від 02.10.92 N 2657-XII «Про інформацію» (Відомості Верховної Ради України (ВВР), 1992, № 48, ст.650)

2. Закон України «Про електронні документи та електронний документообіг» (ВВР, 2003, № 36, ст.275)

3. Закон України «Про електронну комерцію» (ВВР, 2015, № 45, ст.410)

4. Закон України «Про захист персональних даних» (ВВР, 2010, № 34, ст.481)

5. Закон України «Про товариства з обмеженою та додатковою відповідальністю» (ВВР, 2018, № 13, ст.69)

6. Господарський кодекс України. Стаття 62. Підприємство як організаційна форма господарювання (ВВР, 2003, № 18, № 19-20, № 21-22, ст.144)

7. ДСТУ ISO/IEC 2382:2017 Інформаційні технології. Словник термінів (ISO/IEC 2382:2015, IDT)

8. Адамик О. В., Адамик К. Б. Реляційні бази даних як сучасний стандарт накопичення інформації в комп'ютерній системі бухгалтерського обліку. Зб. наук. пр. за матеріалами II Всеукраїнської науково-практичної конференції, м. Дніпро, 29–30 березня 2018 р. С. 698–703.

9. Берко А. Ю., Верес О. М. Організація баз даних: практичний курс. Львів: Вид-во Нац. ун-ту «Львівська політехніка», 2003. – 153 с.

10. Гайна Г. А. Основи проектування баз даних: Навчальний посібник. К.: КНУБА, 2005. – 204 с.

11. Грофф Джеймс Р., Вайнберг Пол Н., Оппель Ендрю Дж. SQL: повний посібник, 3-е видання. М.: Вільямс, 2014. – 960 с.

12. Дейт К. Дж. Введення у системи баз даних, 8-е вид. М.: Вільямс, 2005. – 1328 с.

13. Зарицька О. Л. Бази даних та інформаційні системи: Методичний посібник. Житомир: Вид-во ЖДУ ім. І. Франка, 2009. – 132 с.

14. Зінов'єва І. С., Артемчук В. О. Сучасні підходи до подальшої еволюції концепції баз даних. Proceedings from III International scientific and practical conference «Dynamics of the development of world science». 2019. С. 34–44.

15. Карпенко М. Ю. Інформаційні системи і технології в управлінні організацією. Харків: Вид. ХНАМГ, 2012 – 96 с.

16. Криковцева Н. О., Казакова О. Б., Саркісян Л. Г., Авдєєнко Л. Л., Дяченко Г. А., Курська Л. С., Сахарова О. Н. Комерційна діяльність. К.: Центр учбової літератури, 2007. – 296 с.

17. Куліков С.С. Реляційні бази даних в прикладах. EPAM Systems, 2020–2023 – 424 с.

18. Лосєв М. Ю. Бази даних: навчально-практичний посібник для самостійної роботи студентів. Харків: ХНЕУ ім. С. Кузнеця, 2018. – 233 с.

19. Фіайлі Кріс. SQL: Посібник з вивчення мови. М.: Peachpit Press, 2003. – 456 с.

20. Фуллер Л. У., Кук К. Access 2010 для чайників. М.: "Діалектика", 2010. – 384 с.

21. Шаров С. В., Осадчий В. В. Бази даних та інформаційні системи. Мелітополь: Вид-во МДПУ ім. Б. Хмельницького, 2014. – 352 с.

22. Швачич Г. Г., Толстой В. В., Петречук Л. М., Іващенко Ю. С., Гуляєва О. А., Соболенко О. В. Сучасні інформаційно-комунікаційні технології: Навчальний посібник. Дніпро: НМетАУ, 2017. – 230 с.

23. Шутенко Л. М., Стадник Г. В., Степаненко С. А., Торкатюк В. І., Штерн Г. Ю., Прасол В. М. Основи комерційної діяльності: Навчальний посібник. Харків: ХНАМГ, 2007. – 379 с.

24. Яцюк С. М., Муляр В. П. Використання макросів бази даних Access при вивченні інформатики. Комп'ютерно-інтегровані технології: освіта, наука, виробництво. 2016. № 24-25. С. 54–60.

25. Індекси в базах даних. Види індексів [Електронний ресурс] – Режим доступу:https://av-webmaster.com/ua/posts/indexes-in-databases-types-of-indexes

26. SQL чи NoSQL – ось в чому питання [Електронний ресурс] – Режим доступу: https://alternativescience.net/programming/242-sql-chy-nosql-os-v-chomu-pytannya/

27. AI Workloads and Databases: Hidden Performance Risks That Slow Scaling [Електронний ресурс] – Режим доступу: https://www.enteros.com/blog/software-engineering/ai-workloads-and-databases-hidden-performance-risks-that-slow-scaling

28. Airbnb Case Study [Електронний ресурс] – Режим доступу: https://aws.amazon.com/solutions/case-studies/airbnb-case-study/

29. Best Database Software for Small Business [Електронний ресурс] – Режим доступу: https://automaticnation.com/best-database-software-for-small-business

30. Cloud permeates nearly every facet of data platforms technology [Електронний ресурс] – Режим доступу: https://www.spglobal.com/market-intelligence/en/news-insights/research/cloud-permeates-nearly-every-facet-of-data-platforms-technology

31. DB-Engines Ranking - Trend Popularity [Електронний ресурс] – Режим доступу: https://db-engines.com/en/ranking_trend

32. Dow Jones Case Study [Електронний ресурс] – Режим доступу: https://aws.amazon.com/solutions/case-studies/dow-jones/

33. First Tech Federal Credit Union Predicts a 500%+ ROI by Switching to Snowflake [Електронний ресурс] – Режим доступу: https://www.snowflake.com/wp-content/uploads/2023/02/first-tech-federal-credit-union-predicts-a-500-roi-by-switching-to-snowflake.pdf

34. Global Cybersecurity Outlook 2024 [Електронний ресурс] – Режим доступу: https://www.weforum.org/publications/global-cybersecurity-outlook-2024/

35. Idealo Increases Traffic 6x for Black Friday Using MongoDB Atlas on AWS [Електронний ресурс] – Режим доступу: https://aws.amazon.com/partners/success/idealo-mongodb/

36. Impact of Database Migration on Application Performance: A Case Study of Database Migration from AWS to GCP [Електронний ресурс] – Режим доступу: https://example.com

37. Inlitix: Azure SQL Database [Електронний ресурс] – Режим доступу: https://www.microsoft.com/en/customers/story/24039-inlitix-azure-sql-database

38. Isolation Levels in the Database Engine [Електронний ресурс] – Режим доступу: https://technet.microsoft.com/en-us/library/ms189122(v=SQL.105).aspx

39. Master Advanced Subquery Techniques - Essential Skills for Every MS SQL Developer [Електронний ресурс] – Режим доступу: https://moldstud.com/articles/p-master-advanced-subquery-techniques-essential-skills-for-every-ms-sql-developer

40. MongoDB Atlas provides Vainu Scalability, Security, and Reliability Using AWS [Електронний ресурс] – Режим доступу: https://www.mongodb.com/

41. Non-Relational SQL Market Report [Електронний ресурс] – Режим доступу: https://www.globalgrowthinsights.com/market-reports/non-relational-sql-market-107327

42. NoSQL vs SQL [Електронний ресурс] – Режим доступу: https://www.coursera.org/articles/nosql-vs-sql

43. RavenDB: 5 Key Trends in NoSQL Database Usage for 2024 [Електронний ресурс] – Режим доступу: https://ravendb.net/wp-content/uploads/2024/03/InfoQ-Trend-Report.pdf

44. Sathish Srinivasan, Suresh Bysani Venkata Naga, Krishnaiah Narukulla. AI-Enhanced Distributed Databases: Optimizing Query Processing and Replication Strategies for High-Throughput Applications. International Journal of Artificial Intelligence, Data Science, and Machine Learning. 2022. [Електронний ресурс] – Режим доступу: https://doi.org/10.63282/3050-9262.IJAIDSML-V3I2P109

45. Stack Overflow Developer Survey [Електронний ресурс] – Режим доступу: https://survey.stackoverflow.co/

46. Understanding the Role of Indexing in Database Query Optimization

[Електронний ресурс] – Режим доступу: https://moldstud.com/articles/p-understanding-the-role-of-indexing-in-database-query-optimization-boost-performance-and-efficiency

47. US Foods analyzes transactions from 300,000 customers with snowflake and datarobot [Електронний ресурс] – Режим доступу: https://www.snowflake.com/wp-content/uploads/2021/02/US-Foods-Analyzes-Transactions-From-300000-Customers-with-Snowflake-and-Datarobot.pdf

48. Warveen Merza Eido, Hajar Maseeh Yasin. Machine Learning Approaches for Enhancing Query Optimization in Large Databases. Engineering and Technology Journal. 2025. [Електронний ресурс] – Режим доступу: https://doi.org/10.47191/etj/v10i03.39

49. What is database as a service (DBaaS)? [Електронний ресурс] – Режим доступу: https://www.ibm.com/think/topics/dbaas

50. What is Structured Query Language (SQL)? [Електронний ресурс] – Режим доступу: https://www.ibm.com/think/topics/structured-query-language

51. Why Uber Engineering Switched from Postgres to MySQL [Електронний ресурс] – Режим доступу: https://www.uber.com/en-TR/blog/postgres-to-mysql-migration

52. Yoho: Google Cloud Customer Story [Електронний ресурс] – Режим доступу: https://cloud.google.com/customers/yoho