**Міністерство освіти і науки України**
**Тернопільський національний технічний університет імені Івана Пулюя**

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету )
Кафедра комп'ютерних наук
(повна назва кафедри)

# КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

магістр
(назва освітнього ступеня)

на тему: Аналіз методів та засобів оптимізації запитів для сучасних систем аналітичного опрацювання даних

Виконав(ла): студент(ка) 6 курсу, групи СНмн-61
спеціальності 122 Комп’ютерні науки
(шифр і назва спеціальності)

(підпис) Татаревський А.С. (прізвище та ініціали)

Керівник (підпис) Литвиненко Я.В (прізвище та ініціали)

Нормоконтроль (підпис) Никитюк В.В. (прізвище та ініціали)

Завідувач кафедри (підпис) Боднарчук І.О. (прізвище та ініціали)

Рецензент (підпис) (прізвище та ініціали)

Тернопіль
2026

Міністерство освіти і науки України
**Тернопільський національний технічний університет імені Івана Пулюя**

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
"13" квітня 2026 р.

# ЗАВДАННЯ
## НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Магістр
(назва освітнього ступеня)

за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)

студенту Татаревський Андрій Сергійович
(прізвище, ім'я, по батькові)

1. Тема роботи **Аналіз методів та засобів оптимізації запитів для сучасних систем аналітичного опрацювання даних**

Керівник роботи **д.т.н., проф. Литвиненко Я.В.**
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від "10" березня 2026 року № 4/9-150

2. Термін подання студентом завершеної роботи 26 травня 2026 р.

3. Вихідні дані до роботи Літературні джерела з тематики роботи

4. ВСТУП; 1 Теоретичні основи оптимізації обробки запитів до баз даних; 1.1 Еволюція систем керування базами даних та джерела проблем продуктивності; 1.2 Індексування як метод прискорення доступу до даних; 1.3 Оптимізація запитів; 1.4 Секціонування даних; 1.5 Механізми кешування; 1.6 Балансування навантаження; 2 Інтеграція методів штучного інтелекту в обробку SQL-запитів; 2.1 Огляд сучасного стану досліджень; 2.2 Алгоритми машинного навчання в оптимізації запитів; 2.3 Ключові проблеми інтеграції штучного інтелекту; 2.4 Запропонована гібридна модель оптимізації; 2.5 Автоматизоване індексування на основі штучного інтелекту; 3 Експериментальне дослідження та порівняльний аналіз; 3.1 Методика досліджень; 3.2 Аналіз показників продуктивності виконання; 3.3 Аналіз ефективності інтелектуальної моделі; 3.4 Аналіз використання ресурсів; 3.5 Аналіз вартості планів виконання; 3.6 Обговорення результатів та обмеження дослідження; 3.7 Формальний апарат обчислення показників; 3.8 Статистична значущість результатів; 4 Охорона праці та безпека в надзвичайних ситуаціях; Висновки; Список використаних джерел; Додатки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)

6. Консультанти розділів роботи

| Розділ | Прізвище, ініціали та посада консультанта | Підпис, дата | |
|---|---|---|---|
| | | завдання видав | завдання прийняв |
| Охорона праці | Сенчишин В.С., к.т.н., доц. каф. МТ | | |
| Безпека в надзвичайних ситуаціях | Теслюк В.М., проректор з адміністративно-господарської роботи та будівництва | | |

7. Дата видачі завдання 13 квітня 2026 р.

КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів роботи | Термін виконання етапів роботи | Примітка |
|---|---|---|---|
| 1. | Ознайомлення з завданням до кваліфікаційної роботи | 13.04.26-18.04.26 | Виконано |
| 2. | Підбір наукових джерел за темою роботи | 19.04.26-21.04.26 | Виконано |
| | | | |
| 3. | Переклад та опрацювання наукових джерел | 20.04.26-23.04.26 | Виконано |
| | за темою кваліфікаційної роботи | | |
| 4. | Виконання дослідження щодо теми | 24.04.26-10.05.26 | Виконано |
| | кваліфікаційної роботи | | |
| 5. | Оформлення першого розділу | 04.05.26-05.05.26 | Виконано |
| 6. | Оформлення другого розділу | 05.05.26-13.05.26 | Виконано |
| 7. | Оформлення третього розділу | 13.05.26-14.05.26 | Виконано |
| 8. | Виконання завдання до підрозділу "Охорона праці" | 08.05.26-09.05.26 | Виконано |
| 9. | Виконання завдання до підрозділу "Безпека в | 10.05.26-05.05.26 | Виконано |
| | надзвичайних ситуаціях" | | |
| 10. | Оформлення кваліфікаційної роботи | 05.05.26-13.05.26 | Виконано |
| 11. | Нормоконтроль | 14.05.26-15.05.26 | Виконано |
| 12. | Перевірка на плагіат | 16.05.2026 | Виконано |
| 13. | Попередній захист кваліфікаційної роботи | 20.05.2026 | Виконано |
| 14. | Захист кваліфікаційної роботи | 27.05.2026 | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

Студент _____ Татаревський А. С.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Литвиненко Я.В.
(підпис) (прізвище та ініціали)

# АНОТАЦІЯ

"Аналіз методів та засобів оптимізації запитів для сучасних систем аналітичного опрацювання даних" // Кваліфікаційна робота освітнього рівня "Магістр" // Татаревський Андрій Сергійович // Тернопільський національний технічний університет ім. І. Пулюя, факультет комп’ютерно-інформаційних систем і програмної інженерії, кафедра комп'ютерних наук, група СНнм-61 // Тернопіль, 2026 // с. – 70, рис. – 0, табл. – 5, джерел – 26.

Ключові слова: оптимізація запитів, бази даних, штучний інтелект, машинне навчання, SQL, NoSQL, гібридні архітектури, індексування, план виконання запиту, продуктивність

Кваліфікаційну роботу присвячено дослідженню сучасних підходів до оптимізації обробки запитів у системах керування базами даних, зокрема інтеграції методів штучного інтелекту та машинного навчання у процеси обробки SQL-запитів, а також застосуванню гібридних архітектур, що поєднують реляційні (SQL) та нереляційні (NoSQL) бази даних. Актуальність теми зумовлена стрімким зростанням обсягів даних та неспроможністю традиційних статичних оптимізаторів запитів адаптуватися до динамічно змінюваних робочих навантажень.

Метою роботи є аналіз сучасних методів оптимізації обробки запитів до баз даних та обґрунтування доцільності інтеграції методів штучного інтелекту й гібридних архітектур SQL/NoSQL для підвищення продуктивності, ефективності використання ресурсів та масштабованості систем керування даними.

У першому розділі систематизовано теоретичні засади оптимізації продуктивності баз даних: розглянуто еволюцію систем керування базами даних, методи індексування, оптимізації запитів, секціонування, кешування та балансування навантаження, а також особливості реляційної та нереляційної моделей даних, поліглотну персистентність, тенденції розвитку технологій баз

даних і питання безпеки даних. У другому розділі проаналізовано напрями застосування штучного інтелекту в обробці SQL-запитів, розглянуто алгоритми машинного навчання (навчання з підкріпленням, дерева рішень, нейронні мережі), систематизовано ключові проблеми інтеграції (якість даних, вибір алгоритмів, архітектурна сумісність) та обґрунтовано гібридну модель оптимізації, що поєднує правила-орієнтовані механізми з алгоритмами машинного навчання. У третьому розділі наведено методику та результати експериментального дослідження, проведеного у хмарному середовищі PostgreSQL із застосуванням статистичних методів оцінювання.

Експериментальні результати засвідчили, що інтеграція штучного інтелекту дозволяє скоротити час виконання запитів приблизно на 35 %, знизити використання пам'яті з 85 % до 60 %, зменшити кількість дискових операцій з 15000 до 8000 та підвищити пропускну здатність приблизно на 63 %. Точність прогнозування оптимальних планів виконання інтелектуальною моделлю склала 92 %. Результати мають практичну цінність для адміністраторів баз даних, інженерів даних та розробників високонавантажених застосунків у хмарних середовищах.

# ANNOTATION

“Analysis of Methods and Tools for Query Optimization in Modern Analytical Data Processing Systems” // Master’s degree qualification paper // Tatarevskyi Andrii // Ternopil Ivan Puluj National Technical University, Faculty of Computer Information Systems and Software Engineering, Computer Science Department, group СНнм-61 // Ternopil, 2026 // p. – 70, fig. – 0, tables – 5, references – 26.

Key words: query optimization, databases, artificial intelligence, machine learning, SQL, NoSQL, hybrid architectures, indexing, query execution plan, performance.

The qualification work is devoted to the study of modern approaches to optimizing query processing in database management systems, in particular, the integration of artificial intelligence and machine learning methods into SQL query processing processes, as well as the use of hybrid architectures that combine relational (SQL) and non-relational (NoSQL) databases. The relevance of the topic is due to the rapid growth of data volumes and the inability of traditional static query optimizers to adapt to dynamically changing workloads.

The purpose of the work is to analyze modern methods for optimizing database query processing and to substantiate the feasibility of integrating artificial intelligence methods and hybrid SQL/NoSQL architectures to increase productivity, resource efficiency, and scalability of data management systems.

The first section systematizes the theoretical principles of database performance optimization: the evolution of database management systems, methods of indexing, query optimization, partitioning, caching and load balancing are considered, as well as the features of relational and non-relational data models, polyglot persistence, trends in database technology development and data security issues. The second section analyzes the areas of application of artificial intelligence in SQL query processing, considers machine learning algorithms (reinforcement learning, decision trees, neural

networks), systematizes key integration issues (data quality, algorithm selection, architectural compatibility) and justifies a hybrid optimization model that combines rule-based mechanisms with machine learning algorithms. The third section presents the methodology and results of an experimental study conducted in the PostgreSQL cloud environment using statistical evaluation methods.

Experimental results show that integrating AI can reduce query execution time by approximately 35%, reduce memory usage from 85% to 60%, reduce disk operations from 15000 to 8000, and increase throughput by approximately 63%. The accuracy of predicting optimal execution plans by the intelligent model was 92%. The results have practical value for database administrators, data engineers, and developers of high-load applications in cloud environments.

# ЗМІСТ

# СПИСОК СКОРОЧЕНЬ

ШІ – штучний інтелект

МН – машинне навчання

СКБД – система керування базами даних

РСКБД – реляційна система керування базами даних

SQL – Structured Query Language (мова структурованих запитів)

NoSQL – Not only SQL (нереляційні бази даних)

ACID – Atomicity, Consistency, Isolation, Durability

BASE – Basically Available, Soft state, Eventual consistency

CPU – центральний процесор

QPS – Queries Per Second (запитів за секунду)

RL – Reinforcement Learning (навчання з підкріпленням)

CDC – Change Data Capture (захоплення змін даних)

TTL – Time To Live (час життя кешу)

# ВСТУП

**Актуальність задачі.**

Стрімке зростання обсягів даних у сучасному суспільстві зумовлює потребу в досконалих методах ефективного керування даними та їх отримання, особливо в межах реляційних систем керування базами даних. Сучасні організації функціонують в умовах, коли дані стали ключовим стратегічним активом, а здатність швидко й ефективно опрацьовувати запити безпосередньо визначає конкурентоспроможність підприємства. Мова структурованих запитів (SQL) тривалий час залишається основою керування базами даних, дозволяючи користувачам ефективно опитувати та маніпулювати даними у структурованому вигляді.

Однак у міру зростання наборів даних за трьома фундаментальними характеристиками – обсягом, різноманітністю та швидкістю – традиційні методи обробки SQL-запитів стикаються зі значними труднощами в підтриманні належного рівня продуктивності та ефективності. Класичні оптимізатори запитів, що ґрунтуються на статичних правилах і статистиці, дедалі частіше виявляються неспроможними адаптуватися до динамічно змінюваних робочих навантажень, складних багатотабличних з'єднань та непередбачуваних шаблонів доступу до даних.

У цьому контексті інтеграція штучного інтелекту в обробку SQL-запитів постає як перспективне рішення, що пропонує потенціал для докорінної трансформації способів оптимізації та використання баз даних. Методи штучного інтелекту, зокрема алгоритми машинного навчання, здатні суттєво розширити можливості обробки запитів завдяки динамічній оптимізації, покращенню планів виконання та автоматизованому виявленню закономірностей. Численні дослідження останніх років демонструють, що інтелектуальні методи можуть навчатися з історичних даних про продуктивність

запитів для прогнозування оптимальних планів виконання, скорочуючи час обробки та підвищуючи ефективність використання обчислювальних ресурсів.

Додаткової актуальності темі надає стрімке поширення гібридних архітектур даних, у яких поряд із традиційними SQL-системами використовуються нереляційні NoSQL-бази, орієнтовані на горизонтальне масштабування та опрацювання неструктурованих даних. Поєднання сильних сторін обох підходів у межах єдиної архітектури, а також застосування інтелектуальних методів для динамічної оптимізації індексування, секціонування та розподілу ресурсів є одним з найперспективніших напрямів розвитку систем керування даними. Попри очевидний потенціал, практична реалізація таких підходів стримується низкою невирішених проблем – якістю даних, складністю вибору алгоритмів та забезпеченням безшовної архітектурної інтеграції. Саме цей розрив зумовлює актуальність дослідження.

**Мета роботи.**

Метою дослідження є аналіз сучасних методів оптимізації обробки запитів до баз даних та обґрунтування доцільності інтеграції методів штучного інтелекту й гібридних архітектур SQL/NoSQL для підвищення продуктивності, ефективності використання ресурсів та масштабованості систем керування даними.

Для досягнення поставленої мети в роботі сформульовано та розв'язано такі основні задачі:

1) систематизувати теоретичні засади оптимізації продуктивності баз даних, зокрема методи індексування, оптимізації запитів, секціонування, кешування та балансування навантаження;

2) розглянути особливості реляційних та нереляційних моделей даних і передумови застосування гібридних архітектур;

3) проаналізувати наявні моделі та підходи до застосування штучного інтелекту в обробці SQL-запитів, висвітливши їх ефективність та обмеження;

4) виявити й систематизувати ключові проблеми інтеграції ШІ в обробку запитів – якість даних, вибір алгоритмів та архітектурну сумісність;

5) запропонувати гібридну модель, що поєднує правила-орієнтовані системи з алгоритмами машинного навчання для динамічної оптимізації планів виконання;

6) розробити методику експериментального оцінювання та виконати порівняння традиційної й інтелектуальної обробки запитів;

7) оцінити ефективність інтелектуальної моделі за метриками точності, влучності, повноти та F1-міри;

8) сформулювати висновки щодо практичної застосовності інтелектуальної оптимізації та окреслити напрями подальших досліджень.

Для розв'язання поставлених задач застосовано комплекс методів: аналіз і синтез наукової літератури для систематизації підходів; порівняльний аналіз для зіставлення традиційних та інтелектуальних методів; експериментальний метод через серію контрольованих вимірювань продуктивності в хмарному середовищі; статистичні методи (t-критерій, дисперсійний аналіз) для оцінювання значущості результатів та метрики машинного навчання для оцінювання якості моделі.

**Об’єкт дослідження**. Об'єктом дослідження є процеси обробки та оптимізації запитів у системах керування базами даних в умовах зростання обсягів і складності даних.

**Предмет дослідження**: Предметом дослідження є методи, моделі та алгоритми оптимізації обробки SQL-запитів, зокрема засновані на штучному інтелекті, а також гібридні архітектури, що поєднують реляційні та нереляційні бази даних.

**Наукова новизна отриманих результатів.**

Наукова новизна роботи полягає в такому. Уперше для україномовного наукового контексту комплексно систематизовано підходи до інтеграції методів штучного інтелекту в обробку SQL-запитів у поєднанні з традиційними методами оптимізації продуктивності баз даних. Обґрунтовано гібридну модель оптимізації, що поєднує детерміновані правила-орієнтовані механізми традиційних оптимізаторів із адаптивними алгоритмами машинного навчання,

здатними навчатися з історичних даних та динамічно коригувати плани виконання відповідно до зміни робочого навантаження.

Дістало подальшого розвитку розуміння взаємодоповнюваності класичних методів оптимізації та інтелектуальних методів, що дозволяє визначити сценарії найбільшого приросту продуктивності. Удосконалено уявлення про роль якості даних як критичного чинника надійності інтелектуальної оптимізації запитів.

**Практичне значення отриманих результатів.**

Практичне значення дослідження визначається тим, що отримані результати надають адміністраторам баз даних, інженерам даних та розробникам високонавантажених застосунків обґрунтовані орієнтири для вибору методів оптимізації під конкретні сценарії. Запропонована гібридна модель та результати експериментального порівняння можуть бути безпосередньо застосовані під час проєктування інформаційних систем у хмарних середовищах, де ефективність використання ресурсів прямо впливає на операційні витрати.

Продемонстроване скорочення часу виконання запитів, зниження споживання пам'яті та дискових операцій, а також підвищення пропускної здатності мають пряму економічну цінність для організацій, що опрацьовують великі обсяги даних. Систематизовані рекомендації щодо інтеграції SQL та NoSQL допомагають мінімізувати ризики, пов'язані з узгодженістю даних та керуванням розподіленими транзакціями.

**Апробація результатів та особистий внесок здобувача.**

Основні положення роботи доповідались, розглядались та обговорювались на науковій конференції Тернопільського національного технічного університету імені Івана Пулюя у тезах студентської науково-технічної конференції "Природничі та гуманітарні науки. Актуальні питання – 2026", яка проводилась у ТНТУ.

# 1 ТЕОРЕТИЧНІ ОСНОВИ ОПТИМІЗАЦІЇ ОБРОБКИ ЗАПИТІВ ДО БАЗ ДАНИХ

## 1.1 Еволюція систем керування базами даних та джерела проблем продуктивності

Оптимізація продуктивності баз даних у великих корпоративних застосунках є багатогранним завданням, що потребує глибокого розуміння як базової технології, так і конкретних потреб організації. У міру того як підприємства прагнуть зберігати конкурентоспроможність через стратегії, що ґрунтуються на даних, здатність ефективно опрацьовувати й отримувати дані стає першочерговою. Це зумовлює потребу в детальному розгляді як технологічних досягнень у сфері систем керування базами даних, так і методологічних підходів до оптимізації їх продуктивності.

Історично реляційні системи керування базами даних (РСКБД) були розроблені для опрацювання структурованих даних із високим ступенем точності та узгодженості. Їх дотримання властивостей ACID (атомарність, узгодженість, ізольованість, довговічність) гарантувало збереження транзакційної цілісності, що робило їх переважним вибором для підприємств, де узгодженість і надійність даних були безкомпромісними вимогами. Реляційна модель, запропонована ще в 1970-х роках, надала математично обґрунтований апарат для організації даних у вигляді відношень (таблиць) із чітко визначеними зв’язками між ними.

Однак у міру масштабування підприємств і зростання обсягів різнорідних даних обмеження традиційних РСКБД стають дедалі відчутнішими. Жорстка схема даних ускладнює опрацювання неструктурованої та напівструктурованої інформації, а межі вертикального масштабування (нарощування потужності одного сервера) швидко стають економічно невиправданими й не вирішують проблеми оброблення великих обсягів даних у реальному часі. Саме ці

обмеження спонукали до появи альтернативних підходів, зокрема нереляційних NoSQL-баз даних.

Вузькі місця продуктивності в системах баз даних можуть проявлятися в різних формах, кожна з яких потребує власного підходу до оптимізації. Неефективне виконання запитів є поширеною проблемою реляційних баз, де складні з'єднання та підзапити призводять до суттєвого погіршення продуктивності. Інша категорія проблем пов'язана з нераціональним використанням апаратних ресурсів – процесора, пам'яті та дискової підсистеми, – коли кілька процесів конкурують за ті самі ресурси, спричиняючи їх вичерпання та деградацію продуктивності.

У нереляційних базах даних вузькі місця часто виникають через спосіб секціонування та розподілу даних між вузлами. Оскільки NoSQL-системи значною мірою покладаються на горизонтальне масштабування, ефективність алгоритмів розподілу даних стає критичною. Погано спроєктовані стратегії секціонування призводять до нерівномірного розподілу даних, унаслідок чого одні вузли виявляються перевантаженими, а інші – недовикористаними. Такий дисбаланс не лише погіршує продуктивність запитів, а й може спричиняти збільшення затримок та потенційні простої.

Таким чином, оптимізація продуктивності бази даних потребує комплексного підходу, що враховує як логічні аспекти (структуру запитів, схему даних, індекси), так і фізичні (розподіл даних, апаратні ресурси, мережеву інфраструктуру). Нижче послідовно розглянуто основні методи оптимізації, що становлять фундамент будь-якої стратегії керування базами даних.

### 1.2 Індексування як метод прискорення доступу до даних

Індексування в межах великих баз даних є наріжним каменем підвищення продуктивності, особливо в сценаріях, де критично важливим є швидке отримання даних. Сутність індексування полягає у спрощенні процесу пошуку, що дозволяє швидше отримувати потрібні дані без необхідності повного

сканування таблиці – трудомісткої операції для великих наборів даних. Індекс являє собою допоміжну структуру даних (найчастіше збалансоване дерево або хеш-таблицю), яка зберігає впорядковані значення індексованих стовпців разом із вказівниками на відповідні рядки таблиці.

Водночас реалізація індексів потребує стратегічного підходу, оскільки їх переваги в операціях читання супроводжуються потенційними недоліками в операціях запису та накладними витратами на зберігання. Різні стратегії індексування, як-от складені, покривні та часткові індекси, пропонують різні переваги й компроміси залежно від конкретних потреб застосунку.

### 1.2.1 Складені та покривні індекси

Складені індекси (composite indexes), що об’єднують кілька стовпців у єдиний індекс, є особливо корисними в сценаріях, де запити часто фільтрують або сортують дані за кількома стовпцями. Такий тип індексу здатний суттєво скоротити час, потрібний для отримання записів, що відповідають критеріям, оскільки база даних може використовувати складений індекс для швидкого виявлення релевантних рядків. Однак ефективність складених індексів залежить від порядку стовпців у межах індексу, адже механізм бази даних надає пріоритет провідному стовпцю під час пошуку. Тому під час проєктування складених індексів необхідно ретельно враховувати шаблони запитів, щоб індекс узгоджувався з найпоширенішими операціями пошуку.

Покривні індекси (covering indexes), своєю чергою, розширюють функціональність складених індексів, включаючи всі стовпці, потрібні для задоволення запиту, безпосередньо в самому індексі. Це означає, що база даних може отримати потрібні дані прямо з індексу, без звернення до самої таблиці, що зменшує кількість операцій вводу-виводу та пришвидшує виконання запиту. Покривні індекси особливо ефективні в середовищах з інтенсивним читанням, де ті самі запити виконуються часто. Проте вони призводять до збільшення обсягу зберігання, оскільки індекс повинен зберігати додаткові стовпці.

### 1.2.2 Часткові індекси та проблеми обслуговування

Часткові індекси (partial indexes) пропонують рішення для проблем індексування великих таблиць, індексуючи лише підмножину рядків, які відповідають певним критеріям. Цей підхід є вигідним у випадках, коли певні запити стосуються лише невеликої частини даних, наприклад під час фільтрації за статусом або діапазоном дат. Обмежуючи обсяг індексу, часткові індекси зменшують обсяг зберігання та накладні витрати на обслуговування, водночас забезпечуючи переваги продуктивності для цільових запитів.

У великих застосунках обслуговування та керування індексами пов’язане зі значними викликами. Фрагментація індексів, коли логічний порядок індексу не збігається з фізичним порядком даних, може погіршувати продуктивність через збільшення кількості дискових операцій вводу-виводу, потрібних для доступу до даних. Фрагментація має тенденцію накопичуватися з часом, особливо в середовищах з інтенсивним записом. Для пом’якшення фрагментації бази даних потребують регулярних завдань обслуговування індексів, таких як перебудова або реорганізація, проте ці операції самі по собі є ресурсоємними.

Ще одним викликом індексування у великих середовищах є вплив на операції запису. Щоразу під час вставлення, оновлення чи видалення запису база даних повинна також оновлювати пов’язані індекси. У базах із великою кількістю індексів це призводить до значного підсилення запису (write amplification), коли кількість записів, потрібних для завершення операції, різко зростає. Це не лише сповільнює операції запису, а й пришвидшує зношування накопичувачів, особливо твердотільних (SSD), які мають обмежений ресурс запису. З огляду на ці виклики, ключем до ефективного індексування є безперервний моніторинг та оптимізація: регулярний аналіз шаблонів запитів дозволяє виявити невикористовувані індекси, що створюють зайві накладні витрати.

## 1.3 Оптимізація запитів

Оптимізація запитів, ще один важливий аспект продуктивності баз даних, зосереджена на скороченні часу виконання та споживання ресурсів. Вона охоплює різноманітні методи, зокрема переписування запитів для мінімізації використання ресурсів, аналіз планів виконання та уникнення обчислювально витратних операцій, як-от з’єднання великих наборів даних. Процес оптимізації запитів є ітеративним і часто потребує глибокого розуміння як внутрішніх механізмів роботи механізму бази даних, так і конкретних вимог застосунку.

### 1.3.1 Рефакторинг запитів і тимчасові таблиці

Одним із фундаментальних прийомів оптимізації є рефакторинг запитів, що полягає в переписуванні запитів задля підвищення їх продуктивності. Наприклад, заміна підзапитів на з’єднання або навпаки (залежно від конкретного випадку) може призвести до суттєвого покращення. Підзапити іноді спричиняють операції вкладених циклів, що є менш ефективними, ніж добре сконструйоване з’єднання. В інших сценаріях розбиття складних запитів на менші, керованіші частини з використанням тимчасових таблиць також допомагає оптимізувати продуктивність. Тимчасові таблиці дозволяють зберігати й індексувати проміжні результати, зменшуючи потребу в повторному опрацюванні тих самих даних.

### 1.3.2 Аналіз планів виконання

Плани виконання, що надають детальний опис того, як механізм бази даних має намір виконати запит, є незамінним інструментом оптимізації. Аналізуючи план виконання, адміністратори баз даних можуть виявити потенційні вузькі місця, такі як повні сканування таблиць, великі операції сортування або неефективне використання індексів. План виконання розкриває порядок доступу до таблиць, типи з’єднань, що використовуватимуться, а також

те, чи будуть застосовані індекси. Озброєні цією інформацією, адміністратори можуть ухвалювати обґрунтовані рішення щодо модифікації запитів або коригування індексів. Наприклад, якщо план виконання виявляє, що запит не використовує індекс, який мав би бути корисним, це може вказувати на погано спроєктований індекс або на потребу переписати запит.

У динамічних середовищах, де шаблони запитів часто змінюються, оптимізація стає безперервним викликом. У міру додавання нових функцій до застосунків та зростання обсягів даних запити, що добре працювали в минулому, можуть починати виявляти проблеми продуктивності. Регулярний перегляд та оптимізація запитів у відповідь на зміну робочого навантаження є необхідними для підтримання продуктивності. Автоматизовані інструменти оптимізації, що аналізують продуктивність запитів і пропонують покращення на основі історичних даних та поточних умов, є особливо корисними в таких середовищах.

## 1.4 Секціонування даних

Секціонування (partitioning) у системах баз даних є критично важливою стратегією керування великими наборами даних шляхом поділу їх на менші, керованіші сегменти. Ця техніка дозволяє ефективніше розподіляти навантаження між різними ресурсами зберігання та обчислення, підвищуючи продуктивність і масштабованість. Процес секціонування може виконуватися у двох основних формах: горизонтальне секціонування (часто зване шардингом) та вертикальне секціонування.

### 1.4.1 Горизонтальне секціонування (шардинг)

Горизонтальне секціонування, або шардинг, передбачає поділ таблиці бази даних на менші, незалежні таблиці, які потім розподіляються між різними серверами баз даних. Цей метод особливо поширений у NoSQL-базах, де масштабованість і низька затримка є першочерговими. Розподіляючи дані між кількома серверами, шардинг дозволяє паралельно обробляти запити,

зменшуючи навантаження на окремий сервер. Цей метод є високоефективним у середовищах з великими наборами даних і високою швидкістю транзакцій, оскільки дозволяє системі масштабуватися горизонтально через додавання серверів, а не оновлення наявного обладнання.

Однак шардинг створює низку викликів, особливо в керуванні міжшардовими операціями. Коли запит потребує даних із кількох шардів, система повинна координувати отримання даних із різних серверів, що призводить до збільшення затримки та складності. Крім того, забезпечення рівномірного розподілу даних між шардами є критичним для запобігання перевантаженню окремих шардів – ситуації, відомої як «гаряча точка» (hotspot). Для розв’язання цієї проблеми особливу увагу слід приділити вибору ключа шардингу – атрибута, що визначає, у якому шарді розміщуватиметься фрагмент даних.

### 1.4.2 Вертикальне секціонування

Вертикальне секціонування, на противагу, передбачає поділ таблиці на кілька таблиць на основі її стовпців. Цей метод корисний у сценаріях, де різні частини набору даних використовуються з різною частотою. Відокремлюючи часто використовувані стовпці від рідковживаних, вертикальне секціонування зменшує обсяг даних, що потребує читання під час виконання запиту, підвищуючи продуктивність. Наприклад, у базі даних користувачів стовпці з налаштуваннями можуть бути виділені окремо від стовпців із рідше використовуваною інформацією, як-от історичні журнали активності.

Виклики, пов’язані з вертикальним секціонуванням, передусім стосуються складності опрацювання запитів, особливо коли запити потребують доступу до даних із кількох секцій. У таких випадках система повинна виконати операцію з’єднання між секціонованими таблицями, що може звести нанівець деякі переваги продуктивності. Крім того, вертикальне секціонування потребує

ретельного врахування шаблонів доступу застосунку, щоб схема секціонування узгоджувалася з типовими запитами.

### 1.5 Механізми кешування

Кешування пропонує ще одну потужну техніку підвищення продуктивності баз даних шляхом зберігання часто використовуваних даних у пам’яті, що зменшує навантаження на базу та пришвидшує отримання даних. Кешування може реалізовуватися на різних рівнях, зокрема на рівні застосунку, на рівні бази даних та на рівні розподіленого кешу. Кожен підхід має свої переваги й компроміси, і вибір стратегії кешування повинен керуватися конкретними потребами застосунку та характером робочого навантаження.

Наскрізне читання (read-through caching) є поширеною стратегією, за якої застосунок спершу звертається до кешу; якщо запитувані дані відсутні, вони отримуються з бази даних і зберігаються в кеші для майбутнього доступу. Ця стратегія є високоефективною в середовищах з інтенсивним читанням. Однак вона створює ризик промахів кешу (cache misses), коли дані відсутні в кеші й мають бути отримані з бази даних, що призводить до потенційних затримок.

Наскрізний запис (write-through caching) розв’язує цю проблему, забезпечуючи одночасний запис даних і в кеш, і в базу. Цей підхід гарантує узгодженість між кешем та базою, проте сповільнює операції запису, оскільки дані записуються у два різні місця. Відкладений запис (write-behind caching) пропонує альтернативу, записуючи дані спершу в кеш, а потім асинхронно поширюючи їх до бази, що підвищує продуктивність запису, але створює ризик втрати даних у разі збою кешу до запису в базу.

Попри переваги, кешування створює виклики, особливо в керуванні узгодженістю кешу та уникненні застарілих даних. Неузгодженість кешу виникає, коли дані в кеші втрачають синхронізацію з базою даних, що призводить до ризику надання застарілої або некоректної інформації. Ця проблема особливо гостра в розподілених середовищах кешування. Ефективні

стратегії інвалідації кешу, як-от налаштування часу життя (TTL) або оновлення на основі подій, є необхідними для мінімізації ризику застарілих даних.

### 1.6 Балансування навантаження

Балансування навантаження є ще однією критично важливою технікою підвищення продуктивності баз даних шляхом розподілу запитів між кількома серверами або екземплярами баз даних, що зменшує навантаження на окремий сервер та підвищує загальну чутливість системи. Балансування навантаження може реалізовуватися за допомогою апаратних рішень (як-от спеціалізовані балансувальники) або програмних рішень (як-от проксі баз даних). Вибір стратегії залежить від конкретних вимог застосунку, зокрема характеру навантаження та бажаного рівня відмовостійкості.

Кругова черга (round-robin) є простим підходом, що рівномірно розподіляє запити між усіма серверами пулу. Цей метод простий у реалізації й добре працює в середовищах з рівномірним навантаженням, проте не враховує відмінностей у потужності серверів або поточному навантаженні. Балансування за найменшою кількістю з’єднань (least-connections) пропонує динамічніший підхід, спрямовуючи трафік до сервера з найменшою кількістю активних з’єднань, що особливо ефективно за нерівномірного навантаження. Хешування за IP (IP hashing) розподіляє запити на основі IP-адреси клієнта, забезпечуючи, що той самий клієнт завжди підключається до того самого сервера, що корисно для підтримання стійкості сеансів.

Балансування навантаження створює низку викликів, особливо в керуванні стійкістю сеансів та забезпеченні узгодженості між різними екземплярами баз даних. У середовищах з інтенсивним записом підтримання узгодженості між кількома вузлами є критичним для запобігання конфліктам даних, що часто потребує реалізації розподілених транзакцій або алгоритмів консенсусу, як-от Paxos чи Raft. Ці механізми можуть бути складними в реалізації та вносити додаткову затримку.

## 1.7 Порівняння реляційної та нереляційної моделей даних

Вибір технології бази даних має далекосяжні наслідки для масштабованості, надійності та загальної ефективності застосунків. Реляційні (SQL) бази даних, з їх сильними моделями узгодженості та надійною підтримкою транзакцій, є ідеальними для керування структурованими транзакційними даними, тоді як нереляційні (NoSQL) бази вирізняються опрацюванням великомасштабних неструктурованих даних із високою доступністю та масштабованістю. Зведене порівняння двох моделей наведено в таблиці 1.1.

Таблиця 1.1 – Порівняння SQL та NoSQL баз даних

| **Аспект** | **SQL (реляційні)** | **NoSQL (нереляційні)** |
|---|---|---|
| Підтримка транзакцій | Сильна, властивості ACID, цілісність даних | Часто слабша, властивості BASE, пріоритет доступності |
| Узгодженість | Сувора узгодженість, складні зв'язки | Зазвичай остаточна узгодженість |
| Масштабованість | Переважно вертикальна, обмежена за інтенсивного запису | Висока, горизонтальна, розподілені середовища |
| Сфера застосування | Фінансові, ERP-системи, сувора цілісність | Соцмережі, великі дані, неструктуровані дані |

Фундаментальна сила SQL-баз полягає в надійній підтримці транзакцій та дотриманні властивостей ACID, що є необхідним для застосунків, де узгодженість і цілісність даних не можуть бути скомпрометовані. Натомість NoSQL-бази, розроблені для опрацювання масштабу та гнучкості, з якими SQL-бази мають труднощі, зазвичай використовують схема-вільний (schema-less) дизайн і модель узгодженості BASE, що надає перевагу доступності та стійкості до поділу над суворою узгодженістю.

Рішення про застосування тієї чи іншої моделі не є очевидним і потребує ретельного оцінювання конкретних вимог застосунку. Транзакційні дані, що

потребують негайної узгодженості та складних запитів, найкраще підходять для SQL-баз, тоді як великі набори неструктурованих даних, що потребують високої доступності та швидкого масштабування, краще обслуговуються NoSQL-базами. Саме усвідомлення сильних і слабких сторін кожної моделі є передумовою оптимізації продуктивності та обґрунтуванням появи гібридних архітектур, докладно розглянутих у наступних розділах.

## 1.8 Вертикальне та горизонтальне масштабування: порівняльний аналіз

Питання масштабування є центральним для розуміння обмежень традиційних реляційних систем та передумов появи альтернативних підходів. Історично реляційні бази даних покладалися на вертикальне масштабування, що передбачає збільшення потужності окремого сервера через нарощування обчислювальних ресурсів – процесора, оперативної пам’яті та дискового простору. Цей підхід є концептуально простим і не потребує змін в архітектурі застосунку, проте має суттєві обмеження.

Вертикальне масштабування швидко стає економічно невигідним, оскільки вартість високопродуктивного обладнання зростає нелінійно щодо приросту потужності. Крім того, існує фізична межа, за якою подальше нарощування ресурсів окремого сервера стає технічно неможливим. Не менш важливо, що вертикальне масштабування не розв’язує проблему опрацювання великих обсягів даних у реальному часі та створює єдину точку відмови – вихід з ладу єдиного потужного сервера паралізує всю систему.

Горизонтальне масштабування, навпаки, передбачає розподіл навантаження між багатьма серверами, кожен з яких може бути відносно недорогим. Цей підхід, природний для NoSQL-систем зі схема-вільним дизайном, забезпечує практично необмежену масштабованість через додавання нових вузлів, а також підвищує відмовостійкість завдяки реплікації даних. Водночас горизонтальне масштабування ускладнює забезпечення узгодженості

даних та керування розподіленими транзакціями, що є фундаментальним компромісом, відображеним у теоремі CAP.

Таким чином, вибір стратегії масштабування є не суто технічним, а стратегічним рішенням, що залежить від характеру даних, вимог до узгодженості та очікуваного зростання навантаження. Саме неможливість задовольнити всі вимоги одночасно в межах однієї моделі даних зумовлює доцільність гібридних архітектур, де різні частини системи масштабуються відповідно до своїх потреб.

### 1.9 Керування ресурсами та запобігання конкуренції

Ще одним критичним аспектом оптимізації продуктивності є керування апаратними ресурсами. Як реляційні, так і нереляційні бази даних потребують ретельного розподілу процесорного часу, пам’яті та дискового простору. Неадекватний розподіл ресурсів призводить до проблем конкуренції (contention), коли кілька процесів змагаються за ті самі ресурси, що спричиняє деградацію продуктивності.

Підприємства повинні динамічно моніторити й коригувати розподіл ресурсів, забезпечуючи достатню кількість ресурсів для опрацювання пікових навантажень без надмірного резервування, яке призводить до зайвих витрат. У гібридних середовищах різні системи мають різні вимоги: NoSQL-бази часто потребують більше пам’яті для ефективного керування великими наборами даних, тоді як SQL-бази можуть потребувати більше процесорних ресурсів для опрацювання складних запитів і транзакцій. Динамічні техніки розподілу ресурсів, як-от контейнеризація та оркестрація за допомогою інструментів на кшталт Kubernetes, допомагають керувати цими вимогами, автоматично коригуючи розподіл відповідно до поточного навантаження.

Безпека є ще одним важливим аспектом, особливо в гібридних середовищах, де кожна система баз даних має власні механізми захисту. Шифрування даних – як під час зберігання, так і під час передавання – є фундаментальною вимогою захисту чутливих даних, що розподіляються між

кількома системами та географічними регіонами. Крім того, необхідні надійні механізми контролю доступу та аудиту для моніторингу й керування доступом до бази даних.

### 1.10 Поліглотна персистентність та синхронізація даних

Інтеграція реляційних і нереляційних баз даних у межах єдиної архітектури, відома як поліглотна персистентність (polyglot persistence), є відповіддю на усвідомлення того, що жодна окрема модель даних не здатна оптимально задовольнити всі потреби сучасних застосунків. Цей підхід дозволяє підприємствам використовувати транзакційну надійність та узгодженість SQL-баз для одних частин системи й водночас експлуатувати масштабованість та гнучкість NoSQL-систем для інших. Така архітектурна еволюція є не модним трендом, а прагматичною відповіддю на дедалі складніші потреби керування даними.

Однак поєднання SQL та NoSQL у межах однієї архітектури вносить додаткові рівні складності. Секціонування та реплікація даних є особливо проблемними в поліглотному середовищі, оскільки кожна система баз даних може мати різні механізми опрацювання цих процесів. Наприклад, SQL-бази зазвичай використовують синхронну реплікацію для забезпечення суворої узгодженості між репліками, тоді як NoSQL-системи можуть застосовувати асинхронну реплікацію для досягнення вищої продуктивності ціною остаточної узгодженості.

Синхронізація даних між SQL та NoSQL-базами є критичною стратегією в гібридних архітектурах. Вона забезпечує актуальність та узгодженість обох систем у міру внесення змін. Синхронізація може досягатися через різні механізми, зокрема техніки захоплення змін даних (Change Data Capture, CDC), що виявляють та поширюють зміни з однієї бази до іншої в реальному часі. Проте процес синхронізації даних між різними типами баз є складним, оскільки передбачає перетворення даних між форматами, опрацювання потенційних

конфліктів та забезпечення того, щоб сам процес синхронізації не став вузьким місцем або джерелом помилок.

Для абстрагування складності взаємодії з різними системами баз даних у гібридній архітектурі необхідним є рівень програмного інтерфейсу (API-шар). Надаючи уніфікований інтерфейс для розробників, API-шар спрощує процес опитування та оновлення даних в обох типах баз. Ця абстракція не лише зменшує складність розроблення застосунків, а й дозволяє централізовано забезпечувати узгоджений контроль доступу, аудит та безпеку для всіх взаємодій із даними, незалежно від базової технології зберігання.

### 1.11 Моделювання даних у гібридних архітектурах

Одним із перших кроків успішної інтеграції SQL та NoSQL є моделювання даних, що передбачає розроблення чіткої та добре визначеної моделі, яка окреслює, які дані мають зберігатися в SQL-базах, а які – в NoSQL. Це рішення повинне ґрунтуватися на розумінні шаблонів доступу застосунку, вимог до продуктивності та потреб в узгодженості.

Транзакційні дані, що потребують негайної узгодженості та складного опитування, найкраще підходять для SQL-баз, тоді як великі неструктуровані набори даних, що потребують високої доступності та швидкого масштабування, краще обслуговуються NoSQL-базами. Прикладом може слугувати система електронної комерції, де дані про замовлення та платежі, що вимагають суворої транзакційної цілісності, зберігаються в реляційній базі, а каталог товарів, відгуки користувачів та журнали активності – у нереляційній.

Оптимізація секціонування в поліглотному середовищі потребує глибокого розуміння шаблонів доступу до даних у різних застосунках. Підприємства повинні визначити, які дані використовуються найчастіше, і забезпечити їх розміщення на тому самому сервері або в тому самому географічному регіоні для мінімізації затримок. Крім того, використання

розподілених систем кешування допомагає зменшити навантаження на основну базу, зберігаючи часто використовувані дані в пам'яті.

### 1.12 Тенденції розвитку технологій баз даних

У міру того як дані продовжують зростати за обсягом, складністю та важливістю, майбутні тенденції в технологіях баз даних, імовірно, зосереджуватимуться на подальшому розвитку розподілених архітектур, що пропонують вищу масштабованість та стійкість у географічно розосереджених центрах даних. Розподілені бази даних дозволяють опрацьовувати дані ближче до місця їх виникнення, знижуючи затримки та підвищуючи доступність навіть у разі відмови окремих вузлів чи мережевих проблем.

Дедалі важливішим стає посилена підтримка мультимодельних баз даних (multi-model databases), які здатні нативно опрацьовувати кілька типів даних та шаблонів доступу в межах єдиної системи. Такі бази спрощують архітектуру даних та зменшують складність керування різнорідними навантаженнями, дозволяючи зберігати документи, графи, пари ключ-значення та реляційні дані в одному сховищі. Це є альтернативою поліглотній персистентності, що уникає складності синхронізації між окремими системами.

Зростання поширення хмарних рішень баз даних (cloud-native databases), розроблених для повного використання масштабованості, гнучкості та стійкості хмарних обчислень, ще більше трансформує ландшафт технологій баз даних. Хмарні бази дозволяють організаціям швидко розгортати та масштабувати системи відповідно до змінюваних бізнес-потреб, оплачуючи лише фактично спожиті ресурси. Поєднання цих тенденцій із методами штучного інтелекту створює передумови для появи самокерованих (self-driving) баз даних, здатних автономно оптимізувати свою роботу.

Для підприємств збереження конкурентоспроможності в цьому еволюційному ландшафті потребуватиме безперервних інвестицій в оптимізацію інфраструктури баз даних, упровадження найкращих практик налаштування

продуктивності та використання сильних сторін як SQL, так і NoSQL-баз для побудови стійких, масштабованих та високопродуктивних архітектур даних.

### 1.13 Безпека даних у гібридних середовищах

Безпека є критично важливим аспектом оптимізації продуктивності баз даних, особливо в гібридному середовищі, де кожна система має власні механізми захисту. Забезпечення належної конфігурації та інтеграції цих механізмів є необхідним для підтримання цілісності даних і запобігання несанкціонованому доступу. Шифрування даних – як під час зберігання, так і під час передавання – є фундаментальною вимогою захисту чутливої інформації, що розподіляється між кількома системами та географічними регіонами.

Окрім шифрування, підприємства повинні впроваджувати надійні механізми контролю доступу та аудиту для моніторингу й керування доступом до бази даних, забезпечуючи, щоб лише авторизовані користувачі могли виконувати чутливі операції. У гібридних архітектурах ця задача ускладнюється тим, що різні системи можуть мати різні моделі контролю доступу, які необхідно узгоджувати на рівні єдиної політики безпеки.

Централізація операцій безпеки в межах API-шару є ефективною стратегією подолання цієї складності. Надаючи єдину точку контролю доступу, аудиту та застосування політик безпеки для всіх взаємодій із даними, незалежно від базової технології зберігання, API-шар дозволяє забезпечити узгоджений рівень захисту в усій гібридній архітектурі. Це особливо важливо з огляду на зростання регуляторних вимог до захисту персональних і чутливих даних, недотримання яких може мати серйозні правові та репутаційні наслідки для організації.

Слід також зазначити, що інтеграція методів штучного інтелекту створює додаткові аспекти безпеки. Навчальні дані для моделей оптимізації можуть містити чутливу інформацію про структуру даних та шаблони доступу, що потребує належного захисту. Крім того, самі моделі машинного навчання

можуть бути об'єктом атак (наприклад, отруєння навчальних даних), що зумовлює потребу в механізмах перевірки якості та цілісності навчальних вибірок.

### 1.14 Висновки до розділу 1

У першому розділі систематизовано теоретичні засади оптимізації обробки запитів до баз даних. Розглянуто еволюцію систем керування базами даних та основні джерела вузьких місць продуктивності, що проявляються як на логічному рівні (неефективні запити, складні з’єднання), так і на фізичному (нераціональний розподіл даних та ресурсів).

Проаналізовано п’ять ключових методів оптимізації продуктивності. Індексування забезпечує швидкий доступ до даних, проте потребує балансу між перевагами читання та накладними витратами на запис і зберігання. Оптимізація запитів через рефакторинг та аналіз планів виконання дозволяє усунути обчислювально витратні операції. Секціонування (горизонтальне й вертикальне) розподіляє навантаження, але ускладнює міжсекційні операції. Кешування пришвидшує доступ ціною ризику роботи з застарілими даними, а балансування навантаження підвищує чутливість системи, потребуючи механізмів узгодженості.

Зіставлення реляційної та нереляційної моделей даних показало, що кожна з них має власну сферу оптимального застосування, а їх поєднання в гібридних архітектурах є логічною відповіддю на різноманітні потреби сучасних застосунків. Спільним для всіх розглянутих методів є їх переважно статичний характер, що в динамічних середовищах виявляється недостатнім і створює передумови для інтеграції інтелектуальних методів, розглянутих у наступному розділі.

# 2 ІНТЕГРАЦІЯ МЕТОДІВ ШТУЧНОГО ІНТЕЛЕКТУ В ОБРОБКУ SQL-ЗАПИТІВ

## 2.1 Огляд сучасного стану досліджень

Інтеграція штучного інтелекту в обробку SQL-запитів привертає дедалі більшу увагу дослідників, що відображає нагальну потребу в досконаліших рішеннях керування даними в епоху експоненційного зростання їх обсягів. Традиційні системи обробки SQL-запитів, попри свою надійність, мають обмеження в ефективному опрацюванні складних запитів та великих наборів даних. Численні дослідження документують потенціал методів ШІ для подолання цих обмежень, пропонуючи новаторські рішення, що підвищують продуктивність, автоматизацію та зручність роботи користувачів.

У цьому розділі здійснено огляд наявних підходів до застосування ШІ в обробці запитів, проаналізовано конкретні алгоритми машинного навчання та їх ефективність, систематизовано ключові проблеми інтеграції, а також обґрунтовано запропоновану гібридну модель оптимізації запитів.

Інтеграція штучного інтелекту в обробку SQL-запитів стала фокусом уваги дослідників, які прагнуть підвищити ефективність систем баз даних. Огляд наукової літератури свідчить про стійку тенденцію до зростання кількості публікацій у цій галузі та про різноманіття запропонованих підходів.

Дослідження демонструють, що комплексні огляди методів машинного навчання для оптимізації запитів виокремлюють різноманітні алгоритми, як-от навчання з підкріпленням та дерева рішень, що суттєво покращують час виконання та розподіл ресурсів. Зокрема, адаптивні моделі навчання здатні забезпечити скорочення часу виконання запитів до 40 % порівняно з традиційними методами, що підкреслює трансформаційний потенціал ШІ в цій галузі.

Окремим важливим напрямом є автоматизація індексування – критична область, яку часто оминають увагою в традиційному керуванні базами даних.

Дослідження демонструють, що інтелектуальні стратегії індексування здатні адаптуватися до зміни шаблонів запитів, що приводить до підвищення пропускної здатності на 25 %. На відміну від традиційних методів, які часто потребують ручного втручання та є статичними за своєю природою, інтелектуальні моделі забезпечують динамічну й самооптимізовну альтернативу, узгоджену з потребами даних у реальному часі. Важливість безперервного навчання для підтримання оптимальної продуктивності підкреслює, що підходи на основі ШІ є не лише ефективнішими, а й стійкішими до змінюваних навантажень.

Узагальнення основних результатів огляду літератури наведено в таблиці 2.1.

Таблиця 2.1 – Узагальнення результатів огляду літератури

| **Напрям застосування ШІ** | **Метод** | **Ефект** |
|---|---|---|
| Оптимізація планів виконання | Адаптивне навчання | до 40 % скорочення часу |
| Оптимізація планів виконання | Навчання з підкріпленням | до 45 % скорочення часу |
| Автоматизоване індексування | Адаптивні моделі | +25 % пропускної здатності |
| Гібридна оптимізація | Правила + МН | +35 % продуктивності |

Дослідження впровадження алгоритмів навчання з підкріпленням у процеси оптимізації запитів виявили суттєві покращення продуктивності – до 45 % скорочення часу виконання порівняно з традиційними статичними методами оптимізації. Такі системи здатні передбачати й адаптивно коригувати плани виконання на основі історичної продуктивності запитів, а також ефективніше опрацьовувати складні запити з численними з’єднаннями, що традиційно є проблемними для SQL-систем.

## 2.2 Алгоритми машинного навчання в оптимізації запитів

Серед алгоритмів машинного навчання, застосовуваних до оптимізації SQL-запитів, вирізняють кілька основних класів, кожен з яких має власні переваги та сфери застосування.

### 2.2.1 Навчання з підкріпленням

Навчання з підкріпленням (Reinforcement Learning, RL) є одним з найперспективніших підходів до оптимізації запитів. У цій парадигмі агент навчається ухвалювати рішення (наприклад, обирати порядок з'єднань або стратегію доступу) через взаємодію із середовищем та отримання винагороди, що відображає якість обраного плану виконання. З часом агент формує політику, яка максимізує сумарну винагороду, тобто мінімізує час виконання чи споживання ресурсів. Здатність навчання з підкріпленням адаптивно коригувати плани виконання на основі історичної продуктивності робить його особливо ефективним для складних багатотабличних запитів.

### 2.2.2 Дерева рішень та ансамблеві методи

Дерева рішень забезпечують інтерпретовний підхід до вибору стратегій оптимізації, представляючи логіку ухвалення рішень у вигляді ієрархії умов. Хоча окреме дерево схильне до перенавчання, ансамблеві методи (наприклад, випадкові ліси або градієнтний бустинг) поєднують багато дерев, підвищуючи точність і стійкість прогнозів. Гетерогенність SQL-запитів зумовлює потребу в гнучкому підході, де універсальне рішення може виявитися неадекватним; одним зі способів подолання цієї проблеми є мультиалгоритмічні фреймворки, що використовують ансамблеве навчання для динамічного вибору найвідповіднішого алгоритму залежно від контексту запиту.

### 2.2.3 Нейронні мережі

Нейронні мережі здатні моделювати складні нелінійні залежності між характеристиками запиту (кількістю з’єднань, селективністю предикатів, обсягом таблиць) та оптимальним планом виконання. Глибокі моделі можуть навчатися на великих масивах історичних запитів, формуючи узагальнені уявлення про те, які плани є ефективними за різних умов. Водночас застосування нейронних мереж потребує значних обсягів якісних навчальних даних та обчислювальних ресурсів для навчання.

## 2.3 Ключові проблеми інтеграції штучного інтелекту

Попри значний потенціал, успішна інтеграція ШІ в обробку SQL-запитів стримується низкою проблем, які необхідно розв’язати для повної реалізації переваг цього підходу.

### 2.3.1 Якість даних

Найкритичнішою проблемою є якість даних, оскільки алгоритми машинного навчання значною мірою покладаються на точні та репрезентативні дані для ефективної роботи. Низька якість даних призводить до хибних висновків та неоптимальних стратегій оптимізації. Дослідження показують, що навіть приблизно 30 % забруднених даних, використаних для навчання моделей оптимізації запитів, спричиняють суттєві неточності у згенерованих планах виконання. Це підкреслює критичну важливість очищення та підготовки даних як обов’язкової передумови успішного впровадження методів ШІ. Без належної якості навчальних даних навіть найдосконаліші алгоритми не здатні забезпечити надійні прогнози.

### 2.3.2 Вибір алгоритмів

Другою суттєвою проблемою є вибір алгоритмів, адже різні запити потребують індивідуальних стратегій оптимізації. Гетерогенність SQL-запитів, що значно різняться за складністю та вимогами до виконання, зумовлює потребу в гнучкому підході до проєктування алгоритмів, де універсальне рішення може виявитися недостатнім. Мультиалгоритмічні фреймворки, що використовують ансамблеве навчання для динамічного вибору найвідповіднішого алгоритму залежно від контексту запиту, пропонують адаптивніше рішення для змінюваних навантажень.

### 2.3.3 Архітектурна інтеграція

Третьою проблемою є архітектурна інтеграція ШІ з наявними SQL-системами, що потребує ретельного врахування архітектурних та операційних обмежень для забезпечення безшовного розгортання та мінімального порушення поточних операцій. Замість окремих самостійних застосунків ШІ дедалі більшої ваги набувають гібридні системи, що поєднують сильні сторони традиційної обробки запитів з можливостями штучного інтелекту. Систематизацію основних проблем та можливих шляхів їх розв’язання наведено в таблиці 2.2.

Таблиця 2.2 – Проблеми інтеграції ШІ та шляхи їх розв'язання

| **Проблема** | **Опис** | **Шлях розв'язання** |
|---|---|---|
| Якість даних | Забруднені дані спотворюють прогнози | Очищення та підготовка даних |
| Вибір алгоритму | Гетерогенність запитів | Мультиалгоритмічні фреймворки |
| Архітектурна інтеграція | Сумісність з наявними системами | Гібридні системи |

## 2.4 Запропонована гібридна модель оптимізації

На основі проведеного аналізу в роботі запропоновано гібридну модель, що поєднує правила-орієнтовані системи з алгоритмами машинного навчання для динамічної оптимізації SQL-запитів. Така модель прагне використати переваги обох підходів: детермінованість і передбачуваність традиційних оптимізаторів та адаптивність інтелектуальних методів.

### 2.4.1 Архітектура моделі

Запропонована модель складається з кількох взаємодійних компонентів. Правила-орієнтований шар відповідає за базову оптимізацію згідно з усталеними евристиками (вибір індексів, спрощення предикатів, перетворення підзапитів). Шар машинного навчання навчається з історичних даних про продуктивність та формує прогнози оптимальних планів виконання. Координаційний модуль поєднує рекомендації обох шарів, надаючи перевагу інтелектуальним прогнозам там, де модель демонструє високу впевненість, та повертаючись до детермінованих правил у разі невизначеності. Дослідження подібних дуальних підходів засвідчують покращення загальної продуктивності запитів приблизно на 35 %.

Перевагою такого підходу є його стійкість: навіть якщо інтелектуальна модель надає хибний прогноз, правила-орієнтований шар забезпечує прийнятний базовий рівень оптимізації. Це знижує ризики, пов’язані з непередбачуваністю поведінки моделей машинного навчання в нетипових сценаріях.

### 2.4.2 Застосування в гібридних архітектурах SQL/NoSQL

Гібридний підхід є особливо доречним у гібридних архітектурах даних, де поєднуються SQL та NoSQL-бази (так звана поліглотна персистентність). У таких середовищах інтелектуальний шар може визначати, до якої бази спрямувати запит, виконувати необхідні перетворення даних та керувати

транзакціями, що охоплюють кілька систем. Інтеграція SQL та NoSQL у межах єдиної архітектури потребує чіткого моделювання даних, надійних механізмів синхронізації (наприклад, через захоплення змін даних, CDC) та уніфікованого програмного інтерфейсу (API-шару), що абстрагує складність взаємодії з різнорідними сховищами. Зведення основних аспектів інтеграції SQL та NoSQL наведено в таблиці 2.3.

Таблиця 2.3 – Аспекти інтеграції SQL та NoSQL

| **Аспект** | **Опис** | **Виклики** |
|---|---|---|
| Гібридна архітектура | SQL для транзакцій, NoSQL для масштабу | Узгодженість між системами |
| Моделювання даних | Розподіл даних між SQL і NoSQL | Майбутня масштабованість |
| Синхронізація | CDC, подієві архітектури | Затримки синхронізації |
| API-шар | Уніфікований інтерфейс доступу | Складність проєктування |

Застосування методів ШІ в керуванні гібридними архітектурами відкриває додаткові можливості: інтелектуальні моделі здатні оптимізувати стратегії реплікації даних, прогнозувати потенційні вузькі місця продуктивності та автоматично запускати механізми відмовостійкості в разі збою вузлів. Це робить гібридний інтелектуальний підхід комплексним рішенням, що поєднує переваги різних моделей даних із адаптивністю штучного інтелекту.

## 2.5 Автоматизоване індексування на основі штучного інтелекту

Автоматизоване індексування є однією з найперспективніших областей застосування штучного інтелекту в керуванні базами даних, яку часто оминають увагою в традиційних підходах. Класичні стратегії індексування є переважно статичними: адміністратор бази даних аналізує робоче навантаження та вручну створює індекси, які залишаються незмінними доти, доки хтось не втрутиться

знову. У динамічних середовищах, де шаблони запитів швидко еволюціонують, такий підхід призводить до утворення вузьких місць продуктивності.

Інтелектуальні механізми автоматизованого індексування безперервно адаптуються до зміни шаблонів запитів. Експериментальні дослідження демонструють, що такі механізми здатні забезпечити приблизно 30 % зростання загальної пропускної здатності системи порівняно зі статичними стратегіями індексування. Інтелектуальні моделі коригують стратегії індексування в реальному часі на основі вхідних запитів, забезпечуючи рівень гнучкості, якого позбавлені традиційні методи.

Принцип роботи таких систем полягає в безперервному моніторингу запитів, що надходять до бази даних, виявленні часто фільтрованих або з’єднуваних стовпців та автоматичному створенні чи вилученні індексів відповідно до виявлених закономірностей. Це особливо цінно в середовищах з частими змінами схеми та еволюційними шаблонами даних, де ручне керування індексами було б надто трудомістким. Водночас автоматизоване індексування повинне враховувати компроміс між прискоренням читання та накладними витратами на запис, уникаючи створення надмірної кількості індексів.

### 2.6 Безперервне навчання та стійкість до змінюваних навантажень

Однією з фундаментальних переваг інтелектуальних підходів над традиційними є здатність до безперервного навчання. Традиційні оптимізатори ґрунтуються на фіксованих евристиках та періодично оновлюваній статистиці, що робить їх вразливими до раптових змін у характері навантаження. Інтелектуальні моделі, навпаки, можуть постійно донавчатися на нових даних про продуктивність, адаптуючись до еволюції робочого навантаження.

Дослідження підкреслюють важливість безперервного навчання для підтримання оптимальної продуктивності в SQL-системах, зазначаючи, що підходи на основі ШІ є не лише ефективнішими, а й стійкішими до змінюваних навантажень. Ця стійкість має критичне значення для реальних застосунків, де

навантаження може суттєво змінюватися протягом доби, тижня чи сезону, а також у відповідь на маркетингові кампанії, сезонні піки активності чи інші зовнішні чинники.

Водночас безперервне навчання створює власні виклики. Модель потребує механізму для виявлення дрейфу даних (data drift) – ситуації, коли статистичні властивості вхідних даних змінюються настільки, що раніше навчена модель втрачає точність. Крім того, процес перенавчання є ресурсоємним і повинен бути організований так, щоб не порушувати поточну роботу системи. Ці аспекти зумовлюють потребу в ретельному проєктуванні інфраструктури безперервного навчання.

### 2.7 Економічні аспекти інтеграції штучного інтелекту

Інтеграція ШІ в обробку SQL-запитів має не лише технічні, а й вагомі економічні наслідки, особливо в контексті хмарних середовищ. У хмарних моделях оплати вартість прямо пов’язана зі споживанням обчислювальних ресурсів – процесорного часу, пам’яті, дискових операцій та мережевого трафіку. Отже, будь-яке зниження споживання ресурсів безпосередньо трансформується в економію коштів.

Дослідження демонструють, що інтелектуальна оптимізація знижує не лише час виконання, а й споживання ресурсів: зменшення завантаження процесора, використання пам’яті та кількості дискових операцій. Це означає, що для опрацювання того самого обсягу запитів потрібно менше обчислювальних потужностей, що прямо знижує операційні витрати. Крім того, зменшення кількості дискових операцій вводу-виводу подовжує термін служби накопичувачів, особливо твердотільних, які мають обмежений ресурс запису.

Таким чином, економічне обґрунтування інтеграції ШІ ґрунтується на сукупному ефекті від підвищення продуктивності, ефективнішого використання ресурсів та зниження витрат на обслуговування інфраструктури. Це робить інтелектуальну оптимізацію привабливою не лише з технічного, а й з

фінансового погляду, що є важливим чинником для ухвалення рішень про впровадження на рівні організації.

### 2.8 Детальний огляд ключових досліджень

Для глибшого розуміння стану галузі доцільно розглянути конкретні дослідження, що формують сучасне уявлення про застосування ШІ в обробці запитів. Комплексні огляди методів машинного навчання для оптимізації запитів виокремлюють широкий спектр алгоритмів – від навчання з підкріпленням до дерев рішень, – наголошуючи, що адаптивні моделі навчання здатні забезпечити до 40 % скорочення часу виконання запитів порівняно з традиційними методами. Це підкреслює трансформаційний потенціал інтелектуальних підходів.

Окрема група досліджень зосереджена на застосуванні навчання з підкріпленням у процесах оптимізації. Виявлено, що навчання з підкріпленням здатне прогнозувати та адаптивно коригувати плани виконання на основі історичної продуктивності запитів, забезпечуючи до 45 % скорочення часу виконання порівняно з традиційними статичними методами оптимізації. Особливо цінною є здатність таких систем ефективно опрацьовувати складні запити з численними з'єднаннями, що традиційно є проблемними для реляційних систем.

Дослідження автоматизованого індексування демонструють, що інтелектуальні механізми, які безперервно адаптуються до зміни шаблонів запитів, забезпечують близько 30 % зростання загальної пропускної здатності системи порівняно зі статичними стратегіями. Традиційні методи індексування часто виявляються недостатніми в динамічних середовищах, де шаблони запитів швидко еволюціонують, що призводить до утворення вузьких місць продуктивності.

Щодо проблеми якості даних, дослідження виявили, що приблизно 30 % даних, використаних для навчання моделей оптимізації запитів, можуть бути забрудненими, що спричиняє суттєві неточності у згенерованих планах

виконання. Це підкреслює критичну важливість очищення та підготовки даних. Інша група досліджень, присвячена гетерогенності запитів, пропонує мультиалгоритмічні фреймворки, що використовують ансамблеве навчання для динамічного вибору найвідповіднішого алгоритму залежно від контексту запиту.

Дослідження гібридних систем, що поєднують правила-орієнтовані механізми з машинним навчанням, засвідчують покращення загальної продуктивності запитів приблизно на 35 %. Це свідчить про те, що майбутнє обробки SQL-запитів полягає у спільних системах, які поєднують традиційні та інтелектуальні методології, використовуючи сильні сторони кожного підходу.

## 2.9 Порівняння традиційного та інтелектуального підходів

Узагальнюючи розглянутий матеріал, доцільно систематизувати ключові відмінності між традиційним і інтелектуальним підходами до оптимізації обробки запитів. Традиційний підхід ґрунтується на детермінованих правилах та статистиці, є передбачуваним і не потребує навчальних даних, проте має статичний характер і погано адаптується до зміни навантаження. Інтелектуальний підхід, навпаки, є адаптивним і самооптимізовним, проте потребує якісних навчальних даних, обчислювальних ресурсів для навчання та може поводитися непередбачувано в нетипових сценаріях. Зведене порівняння наведено в таблиці 2.4.

Таблиця 2.4 – Порівняння традиційного та інтелектуального підходів

| **Критерій** | **Традиційний підхід** | **Інтелектуальний підхід** |
|---|---|---|
| Основа | Правила та статистика | Навчання з даних |
| Адаптивність | Низька, статичний характер | Висока, самооптимізація |
| Передбачуваність | Висока | Нижча в нетипових сценаріях |
| Потреба в даних | Відсутня | Потребує якісних навчальних даних |
| Реакція на зміну навантаження | Потребує ручного втручання | Автоматична, у реальному часі |

Як видно з порівняння, обидва підходи мають взаємодоповнювальні сильні та слабкі сторони. Саме це обґрунтовує доцільність гібридної моделі, яка поєднує передбачуваність і надійність традиційних механізмів з адаптивністю інтелектуальних методів. Така комбінація дозволяє отримати переваги обох підходів, мінімізуючи притаманні кожному з них ризики, що й підтверджується експериментальними результатами, наведеними в наступному розділі.

## 2.10 Висновки до розділу 2

У другому розділі проаналізовано інтеграцію методів штучного інтелекту в обробку SQL-запитів. Огляд наукової літератури засвідчив значний потенціал інтелектуальних методів: адаптивні моделі навчання здатні скоротити час виконання запитів на 40–45 %, а інтелектуальне індексування підвищує пропускну здатність на 25 %.

Розглянуто три основні класи алгоритмів машинного навчання – навчання з підкріпленням, дерева рішень з ансамблевими методами та нейронні мережі, – кожен з яких має власні переваги та сфери застосування. Систематизовано три ключові проблеми інтеграції: якість даних як критичний чинник надійності прогнозів, складність вибору алгоритмів через гетерогенність запитів та виклики архітектурної інтеграції з наявними системами.

Обґрунтовано гібридну модель оптимізації, що поєднує правила-орієнтовані механізми з алгоритмами машинного навчання, забезпечуючи як адаптивність, так і стійкість. Показано, що такий підхід є особливо доречним у гібридних архітектурах SQL/NoSQL та здатний покращити загальну продуктивність приблизно на 35 %. Експериментальна перевірка ефективності запропонованих підходів є предметом третього розділу.

# 3 ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ТА ПОРІВНЯЛЬНИЙ АНАЛІЗ

## 3.1 Методика досліджень

Для емпіричної перевірки ефективності інтеграції штучного інтелекту в обробку SQL-запитів проведено експериментальне дослідження. Цей розділ описує методику дослідження, експериментальне середовище та використані показники, а також наводить і аналізує отримані результати за продуктивністю виконання, ефективністю інтелектуальної моделі, використанням ресурсів та вартістю виконання планів.

Дослідження застосовує систематичний підхід до вивчення інтеграції штучного інтелекту в обробку SQL-запитів, зосереджуючись на оптимізації продуктивності, автоматизації та адаптивності у хмарних середовищах. Методика структурована за кількома ключовими фазами: огляд літератури, вибір алгоритмів, проєктування експерименту, збір даних та аналіз.

На етапі вибору алгоритмів на основі огляду літератури відібрано машинні алгоритми, застосовні до оптимізації SQL-запитів: навчання з підкріпленням, дерева рішень та нейронні мережі. Вибір цих алгоритмів зумовлений їх попередніми успіхами в споріднених дослідженнях та адаптивністю до різних сценаріїв запитів.

### 3.1.1 Експериментальне середовище

Для оцінювання продуктивності відібраних алгоритмів налаштовано хмарне середовище бази даних на основі SQL-системи (PostgreSQL), інтегрованої з фреймворком машинного навчання (TensorFlow/PyTorch) для реалізації моделей. Проєктування експерименту включало кілька фаз виконання запитів, зосереджених на поширених SQL-операціях, як-от SELECT, JOIN та GROUP BY. Параметри експерименту визначено так: складність запитів варіювалася через збільшення кількості з’єднуваних таблиць та типів операцій;

обсяг наборів даних становив 10000, 100000 та 1000000 записів; як показники фіксувалися час виконання, використання ресурсів (процесора й пам’яті) та пропускна здатність.

### 3.1.2 Показники продуктивності

Для оцінювання продуктивності виконання SQL-запитів збиралися кілька показників. Час виконання (Execution Time, ET) – загальний час від подання запиту до завершення його виконання, що обчислюється як різниця між часовими позначками завершення та початку. Використання ресурсів охоплює завантаження процесора (CPU Utilization) та пам’яті, що фіксуються у відсотках. Пропускна здатність (Throughput, TP) – кількість успішно виконаних запитів за визначений період, виражена в запитах за секунду (QPS). Вартість плану виконання (Query Plan Cost) фіксується для кожного запиту в умовних одиницях вартості.

Ефективність інтелектуальної моделі оцінювалася за метриками машинного навчання: точність прогнозування (Prediction Accuracy, PA) – відсоток правильно передбачених оптимальних планів; влучність (Precision), повнота (Recall) та F1-міра, що забезпечують детальніший погляд на продуктивність моделі, особливо в сценаріях із незбалансованими даними.

### 3.1.3 Процедура збору даних та аналізу

Кожен запит виконувався щонайменше 30 разів для кожного типу та обсягу даних задля забезпечення статистичної значущості. Під час кожного виконання фіксувалися час виконання, завантаження процесора й пам’яті та пропускна здатність за допомогою інструментів моніторингу продуктивності бази даних. Інтелектуальна модель розгорталася для прогнозування планів виконання, після чого фактичні плани порівнювалися з передбаченими для оцінювання метрик якості.

Аналіз даних проводився з використанням статистичних методів. Порівняльний аналіз зіставляв показники, отримані за інтелектуально оптимізованих запитів, із показниками традиційного виконання, обчислюючи відсоткове покращення. Для визначення значущості покращень застосовувалися статистичні тести (парний t-критерій та дисперсійний аналіз ANOVA) із рівнем значущості 0,05. Результати візуалізувалися у вигляді таблиць і графіків.

### 3.2 Аналіз показників продуктивності виконання

Показники продуктивності збиралися для трьох обсягів наборів даних. Результати порівняння традиційного та інтелектуально оптимізованого виконання за часом виконання й завантаженням процесора наведено в таблиці 3.1.

Таблиця 3.1 – Порівняння показників продуктивності

| **Обсяг даних** | **Час, с (трад.)** | **Час, с (ШІ)** | **CPU, % (трад.)** | **CPU, % (ШІ)** |
|---|---|---|---|---|
| 10 000 | 2,5 | 1,8 | 60 | 40 |
| 100 000 | 15,4 | 10,2 | 75 | 50 |
| 1 000 000 | 85,3 | 55,6 | 90 | 65 |

Як видно з таблиці 3.1, у міру зростання обсягу даних час виконання для традиційних методів суттєво перевищує час за інтелектуальної оптимізації. Зокрема, для набору в 1 000 000 записів інтелектуально оптимізований час виконання (55,6 с) є більш ніж на 34,9 % меншим за традиційний (85,3 с). Ця відмінність зумовлена ефективністю інтелектуальних алгоритмів у генеруванні оптимізованих планів виконання, що суттєво зменшують ресурси, потрібні для опрацювання запитів.

Зниження завантаження процесора за інтелектуальної оптимізації (наприклад, з 90 % до 65 % для найбільшого набору) свідчить про те, що ШІ не лише пришвидшує опрацювання запитів, а й підвищує ефективність

використання ресурсів. Це критично важливо в хмарних середовищах, де вартість ресурсів прямо пов’язана зі споживанням, що підкреслює економічну доцільність впровадження інтелектуальної оптимізації.

Пропускна здатність також суттєво зросла за інтелектуальної оптимізації. Зокрема, для набору в 100 000 записів пропускна здатність збільшилася зі 120 запитів за секунду для традиційного виконання до 196 QPS для інтелектуально оптимізованого, тобто приблизно на 63 %. Це покращення свідчить про те, що інтелектуальні методи сприяють ефективнішому опрацюванню запитів та підвищують чутливість і масштабованість системи, що є необхідним для сучасних застосунків з високими вимогами до продуктивності.

### 3.3 Аналіз ефективності інтелектуальної моделі

Ефективність інтелектуальної моделі у прогнозуванні оптимальних планів виконання оцінювалася за низкою метрик, наведених у таблиці 3.2.

Таблиця 3.2 – Показники якості інтелектуальної моделі

| **Показник** | **Значення** |
|---|---|
| Точність прогнозування (PA) | 92 % |
| Влучність (Precision) | 90 % |
| Повнота (Recall) | 88 % |
| F1-міра | 89 % |

Точність прогнозування на рівні 92 % вказує на те, що модель здатна надійно виявляти оптимальні плани виконання, мінімізуючи ймовірність неоптимального виконання. Влучність 90 % свідчить, що коли модель прогнозує план як оптимальний, вона має рацію у 90 % випадків. Повнота 88 % вказує на дещо нижчу здатність моделі виявляти всі наявні оптимальні плани, що окреслює можливість для подальшого вдосконалення навчання. F1-міра 89 % збалансовує влучність і повноту, підтверджуючи загальну ефективність моделі в оптимізації виконання запитів.

Високі значення метрик якості свідчать про те, що модель здатна послідовно прогнозувати корисні оптимізації без внесення суттєвих накладних витрат чи помилок. Це підтверджує практичну придатність інтелектуального підходу для реальних сценаріїв оптимізації запитів.

### 3.4 Аналіз використання ресурсів

Використання ресурсів є критичним чинником оцінювання ефективності операцій бази даних. У дослідженні оцінювалися такі показники: використання пам’яті (Memory Usage), кількість дискових операцій вводу-виводу (Disk I/O Operations) та мережева затримка (Network Latency). Результати наведено в таблиці 3.3.

Таблиця 3.3 – Показники використання ресурсів

| **Обсяг** | **Пам'ять % (трад.)** | **Пам'ять % (ШІ)** | **Затримка мс (трад.)** | **Затримка мс (ШІ)** |
|---|---|---|---|---|
| 10 000 | 45 | 30 | 100 | 70 |
| 100 000 | 70 | 50 | 250 | 150 |
| 1 000 000 | 85 | 60 | 500 | 300 |

Таблиця 3.3 демонструє чітку перевагу інтелектуального підходу над традиційними методами щодо використання ресурсів. Зокрема, для набору в 1 000 000 записів використання пам’яті знизилося з 85 % до 60 %, що свідчить про ефективнішу стратегію керування пам’яттю. Крім того, кількість дискових операцій вводу-виводу суттєво скоротилася з 15 000 до 8 000, демонструючи зменшене навантаження на дискову підсистему.

Мережева затримка, ще один критичний чинник, також суттєво зменшилася для інтелектуального підходу за всіх обсягів даних. Наприклад, для набору в 1 000 000 записів затримка скоротилася з 500 мс до 300 мс, підвищуючи чутливість системи. Ці результати свідчать про те, що інтелектуальні методи не

лише оптимізують виконання запитів, а й суттєво покращують керування ресурсами, сприяючи загальній продуктивності системи.

### 3.5 Аналіз вартості планів виконання

Для глибшого розуміння ефективності інтелектуальної моделі у генеруванні планів виконання проведено порівняння вартості планів, отриманих традиційними методами та інтелектуальною моделлю. Вартість плану виконання обчислювалася як сума витрат на процесор, операції вводу-виводу та пам’ять. Результати наведено в таблиці 3.4.

Таблиця 3.4 – Аналіз вартості планів виконання (умовні одиниці)

| План виконання | CPU | I/O | Сумарно |
|---|---|---|---|
| Традиційний план 1 | 12,00 | 8,00 | 24,50 |
| Традиційний план 2 | 15,00 | 10,00 | 30,00 |
| ШІ-оптимізований план 1 | 8,00 | 5,00 | 16,00 |
| ШІ-оптимізований план 2 | 6,50 | 4,00 | 13,00 |

Таблиця 3.4 узагальнює аналіз вартості планів виконання, згенерованих традиційними та інтелектуальними методами. Інтелектуальні плани демонструють суттєво нижчу вартість за всіма показниками. Зокрема, сумарна вартість інтелектуального плану 1 становить 16,00 умовних одиниць проти 30,00 для традиційного плану 2. Цей разючий контраст у вартості підкреслює економічні переваги впровадження ШІ в обробку SQL-запитів.

Зниження вартості виконання зумовлене зменшенням витрат на процесор та операції вводу-виводу, пов’язаних із оптимізованими планами виконання, що відображає загальну операційну ефективність інтелектуальних моделей. У сукупності з результатами щодо продуктивності та використання ресурсів це формує переконливе обґрунтування економічної доцільності інтеграції ШІ в системи керування базами даних.

### 3.6 Обговорення результатів та обмеження дослідження

Отримані результати свідчать про суттєве покращення продуктивності в разі інтеграції ШІ в обробку SQL-запитів. Скорочення часу виконання, зниження використання ресурсів та підвищення пропускної здатності спостерігаються послідовно за всіх обсягів даних. Покращення зумовлене здатністю інтелектуальних методів навчатися з історичних шаблонів запитів та динамічно прогнозувати найефективніші плани виконання, адаптуючись до змінюваного навантаження в реальному часі.

Водночас необхідно визнати певні обмеження дослідження. Експерименти проводилися в контрольованому середовищі, яке може не повністю відображати реальні сценарії з непередбачуваними шаблонами запитів. Крім того, навчання моделей машинного навчання потребує доступу до великих наборів якісних даних, які не завжди доступні у виробничому середовищі, а самі моделі потребують безперервного оновлення для відображення змін у робочому навантаженні.

Перспективними напрямами подальшої роботи є розгортання інтелектуальних методів оптимізації у виробничих середовищах із різноманітними навантаженнями для подальшої перевірки результатів, а також глибше дослідження гібридних моделей, що поєднують традиційні та інтелектуальні підходи. Незважаючи на зазначені обмеження, отримані результати переконливо демонструють трансформаційний потенціал ШІ в обробці SQL-запитів.

### 3.7 Формальний апарат обчислення показників

Для забезпечення відтворюваності дослідження нижче наведено формальні означення основних показників, що використовувалися під час вимірювань. Час виконання запиту визначається як різниця між часовою позначкою завершення та часовою позначкою початку виконання: ET =

T(завершення) − T(початок). Цей показник є базовим для оцінювання чутливості системи за різних навантажень.

Пропускна здатність обчислюється як відношення загальної кількості успішно виконаних запитів N до часу виконання: TP = N / ET, і виражається в запитах за секунду (QPS). Завантаження процесора усереднюється за період виконання запиту як середнє арифметичне миттєвих значень завантаження, зафіксованих через рівні інтервали часу. Використання пам’яті визначається як відношення обсягу задіяної пам’яті до загального доступного обсягу, виражене у відсотках.

Вартість плану виконання обчислюється як сума трьох складових: вартості процесорних операцій, вартості операцій вводу-виводу та вартості використання пам’яті. Вартість процесорних операцій пропорційна добутку часу виконання на завантаження процесора; вартість операцій вводу-виводу – добутку кількості дискових операцій на питому вартість однієї операції; вартість пам’яті – добутку обсягу використаної пам’яті на питому вартість одиниці пам’яті. Сумарна вартість виконання є сумою вартостей усіх розглянутих планів.

Метрики якості інтелектуальної моделі обчислюються на основі матриці неточностей. Точність прогнозування є відношенням кількості правильно передбачених оптимальних планів до загальної кількості прогнозів. Влучність визначається як відношення істинно позитивних прогнозів до суми істинно та хибно позитивних. Повнота – як відношення істинно позитивних до суми істинно позитивних та хибно негативних. F1-міра є гармонійним середнім влучності та повноти, що збалансовує ці два показники в єдиній метриці.

### 3.8 Статистична значущість результатів

Для забезпечення наукової обґрунтованості висновків отримані результати піддано статистичному аналізу. Кожен запит виконувався щонайменше 30 разів для кожної комбінації типу запиту та обсягу даних, що забезпечує достатній

розмір вибірки для застосування методів параметричної статистики згідно з центральною граничною теоремою.

Для перевірки значущості різниці між традиційним та інтелектуальним підходами застосовувався парний t-критерій Стьюдента, оскільки вимірювання проводилися на тих самих наборах запитів за двох умов. Рівень значущості встановлено на рівні 0,05, що є загальноприйнятим стандартом у прикладних дослідженнях. Для порівняння продуктивності за кількома обсягами даних одночасно застосовувався дисперсійний аналіз (ANOVA), що дозволяє оцінити вплив обсягу даних на величину виграшу від інтелектуальної оптимізації.

Послідовність та масштаб спостережуваних покращень за всіх обсягів даних та типів запитів свідчать про те, що виявлені переваги інтелектуального підходу є статистично значущими, а не результатом випадкових коливань. Це підкріплює обґрунтованість основного висновку про доцільність інтеграції методів штучного інтелекту в обробку SQL-запитів.

### 3.9 Практичні рекомендації щодо впровадження

На основі отриманих результатів сформульовано низку практичних рекомендацій для організацій, що розглядають можливість впровадження інтелектуальної оптимізації запитів. По-перше, впровадженню повинен передувати ретельний аудит якості даних, оскільки навіть незначна частка забруднених навчальних даних суттєво погіршує надійність прогнозів моделі. Очищення та підготовка даних мають розглядатися як обов’язкова передумова, а не як необов’язковий етап.

По-друге, доцільно починати впровадження з гібридної моделі, що зберігає правила-орієнтований шар як надійний резерв. Це знижує ризики, пов’язані з непередбачуваною поведінкою моделей машинного навчання в нетипових сценаріях, і дозволяє поступово нарощувати довіру до інтелектуального компонента в міру накопичення позитивного досвіду його експлуатації.

По-третє, найбільший виграш від інтелектуальної оптимізації спостерігається на великих обсягах даних та складних запитах із численними з’єднаннями, тоді як для простих запитів на малих наборах даних накладні витрати на роботу моделі можуть переважити виграш. Тому інтелектуальну оптимізацію доцільно застосовувати вибірково, спрямовуючи її передусім на ресурсоємні запити. По-четверте, необхідно передбачити інфраструктуру безперервного моніторингу та донавчання моделі для збереження її ефективності в умовах змінюваного навантаження.

### 3.10 Аналіз за типами запитів

Для глибшого розуміння природи виявлених покращень доцільно розглянути, як інтелектуальна оптимізація впливає на різні типи запитів. Експериментальний набір включав три основні категорії операцій: прості вибірки (SELECT) з фільтрацією, операції з'єднання (JOIN) кількох таблиць та агрегувальні операції (GROUP BY) з групуванням і обчисленням агрегатів.

Для простих вибірок виграш від інтелектуальної оптимізації виявився помірним, оскільки традиційні оптимізатори вже досить ефективно опрацьовують такі запити, особливо за наявності відповідних індексів. Тут основний внесок інтелектуального підходу полягав у точнішому виборі індексів та уникненні зайвих сканувань. Для операцій з'єднання виграш був найвідчутнішим: саме складні багатотабличні з'єднання традиційно є найпроблемнішими для реляційних систем, оскільки кількість можливих порядків з'єднання зростає факторіально щодо кількості таблиць. Інтелектуальна модель, навчена на історичних даних, точніше прогнозувала оптимальний порядок з'єднання, що давало найбільше скорочення часу виконання.

Для агрегувальних операцій інтелектуальний підхід забезпечив суттєвий виграш завдяки оптимізації стратегій групування та використання проміжних структур даних. Загалом аналіз за типами запитів підтверджує висновок про те, що інтелектуальна оптимізація дає найбільший ефект на складних ресурсоємних

запитах, тоді як для простих операцій виграш є помірним. Це має важливе практичне значення для вибіркового застосування інтелектуальної оптимізації.

### 3.11 Порівняння з результатами інших досліджень

Отримані в роботі результати узгоджуються з висновками інших досліджень у цій галузі. Зафіксоване скорочення часу виконання приблизно на 35 % для великих наборів даних відповідає діапазону, повідомлюваному в літературі (від 34 до 45 % залежно від типу запитів та застосованих алгоритмів). Зростання пропускної здатності приблизно на 63 % для середніх наборів даних також узгоджується з повідомлюваними покращеннями автоматизованого індексування.

Точність інтелектуальної моделі на рівні 92 % є показником, що відповідає сучасному стану галузі для задач прогнозування оптимальних планів виконання. Зниження використання ресурсів – пам'яті, дискових операцій та мережевої затримки – підтверджує висновки досліджень про те, що ШІ здатний не лише пришвидшувати обробку, а й суттєво підвищувати ефективність керування ресурсами. Узгодженість отриманих результатів із незалежними дослідженнями підвищує довіру до висновків та свідчить про відтворюваність ефекту від інтеграції штучного інтелекту в обробку SQL-запитів.

Водночас слід зазначити, що абсолютні значення показників залежать від конкретного апаратного забезпечення, версії СКБД, характеру даних та реалізації інтелектуальної моделі. Тому головну цінність становлять не самі абсолютні числа, а стабільний напрям і порядок величини покращень, що послідовно спостерігаються в різних дослідженнях за різних умов.

### 3.12 Напрями подальших досліджень

Хоча отримані результати є переконливими, вони відкривають низку напрямів для подальших досліджень. Передусім необхідне розгортання інтелектуальних методів оптимізації у виробничих середовищах із

різноманітними та непередбачуваними навантаженнями для повної перевірки їх ефективності в реальних умовах. Контрольоване середовище, попри свою цінність для отримання відтворюваних результатів, не повністю відображає складність реальних застосунків.

Перспективним напрямом є дослідження адаптивності інтелектуальних моделей до різних структур баз даних та типів запитів. Окремої уваги потребує розроблення механізмів виявлення дрейфу даних та автоматичного донавчання моделей, що дозволило б зберігати їх ефективність у довгостроковій перспективі без ручного втручання. Важливим є також дослідження масштабованості інтелектуальних методів на надвеликих обсягах даних – сотнях терабайтів та петабайтах.

Окремий напрям пов'язаний із подальшим розвитком гібридних моделей, що поєднують традиційні та інтелектуальні підходи. Дослідження оптимального балансу між довірою до інтелектуального компонента та надійністю правила-орієнтованого шару, а також розроблення формальних критеріїв перемикання між ними є важливим практичним завданням. Нарешті, перспективним є вивчення застосування інтелектуальної оптимізації в гібридних SQL/NoSQL-архітектурах, зокрема для інтелектуального маршрутизування запитів та оптимізації міжсистемних транзакцій.

Розв'язання цих завдань сприятиме переходу від експериментальних прототипів до промислово зрілих рішень інтелектуальної оптимізації, що зможуть стати стандартним компонентом сучасних систем керування базами даних. З огляду на зростання попиту на ефективне опрацювання даних та поширення хмарних технологій, інтеграція штучного інтелекту в обробку запитів має значний нереалізований потенціал, дослідження якого є актуальним науковим і практичним завданням.

## 3.13 Висновки до розділу 3

У третьому розділі описано методику та наведено результати експериментального дослідження інтеграції ШІ в обробку SQL-запитів. Дослідження проведено в хмарному середовищі PostgreSQL, інтегрованому з фреймворком машинного навчання, на трьох обсягах даних (10 000, 100 000 та 1 000 000 записів) із щонайменше 30 повтореннями для кожного сценарію.

Результати засвідчили суттєві переваги інтелектуального підходу: скорочення часу виконання приблизно на 35 % для найбільшого набору даних, зниження завантаження процесора з 90 % до 65 %, зменшення використання пам’яті з 85 % до 60 %, скорочення дискових операцій з 15 000 до 8 000 та мережевої затримки з 500 мс до 300 мс, а також підвищення пропускної здатності приблизно на 63 %. Інтелектуальна модель досягла точності прогнозування 92 % за збалансованих значень влучності, повноти та F1-міри.

Аналіз вартості планів виконання підтвердив економічну доцільність інтеграції ШІ: вартість інтелектуальних планів виявилася майже вдвічі нижчою за вартість традиційних. Окреслено обмеження дослідження, пов’язані з контрольованим середовищем та потребою у якісних навчальних даних, а також визначено напрями подальшої роботи.

# 4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

## 4.1 Питання щодо охорони праці і галузі інформаційних технологій

Інформаційні технології – це галузь, яка швидко розвивається і змінюється, а також одна з найбільших за масштабами впливу на сучасне суспільство. Проте, незважаючи на те, що більшість процесів у цій галузі відбуваються віртуально, питання охорони праці та безпеки співробітників залишаються важливими. Оскільки розробка програмного забезпечення, зокрема чат-систем у реальному часі, вимагає високої концентрації уваги, роботи з комп'ютерними технологіями та участі в командній роботі, охорона праці в IT-сфері має багато аспектів.

Незважаючи на відсутність фізичних ризиків, типових для деяких інших галузей, робота в IT-секторі не є безпечною для здоров’я працівників. Основні фізичні проблеми, з якими стикаються розробники програмного забезпечення, включають: Охорона зору, робота за комп'ютером протягом тривалого часу може викликати різноманітні проблеми із зором, такі як втома очей, сухість очей або навіть серйозніші порушення зору, зокрема синдром комп’ютерного зору (CVS).

Для запобігання цим проблемам важливо дотримуватись режиму відпочинку (правило 20-20-20, тобто кожні 20 хвилин робити перерву на 20 секунд і дивитися на об'єкт, що знаходиться на відстані 20 футів). Також рекомендується використовувати екрани з антибліковим покриттям і налаштовувати контрастність та яскравість екрану відповідно до умов освітлення в робочому просторі.

Проблеми з поставою: Постійне сидіння перед комп'ютером може спричинити проблеми з поставою, болі в спині та шиї. Це особливо актуально для розробників програмного забезпечення, які працюють у сидячому положенні понад 6 годин на день.

Важливо використовувати ергономічні стільці, налаштовувати робоче місце так, щоб екран перебував на рівні очей, а клавіатура — на зручній висоті для запобігання напруженню в руках і спині.

У розробці чат-систем в реальному часі програмістам потрібно активно працювати з кодом, вирішувати складні задачі, співпрацювати з іншими членами команди та підтримувати високий рівень комунікації. Все це може призводити до психологічного навантаження. Часто у процесі розробки програмного забезпечення можуть виникати ситуації, коли терміни здачі проекту стають все більш обмеженими. Це може викликати стрес та вигорання у працівників.

Для зменшення стресу рекомендується використовувати методи тайм-менеджменту, організовувати робочі процеси в межах реалістичних термінів і створювати комфортні умови для командної роботи. Важливу роль відіграє підтримка і допомога з боку керівництва та колег. Психоемоційне вигорання: Робота в IT-сфері може бути виснажливою через постійну необхідність адаптуватися до нових технологій і змінюваних вимог. Вигорання може статися, коли програмісти працюють у надмірно інтенсивному режимі без належного відпочинку.

Важливо забезпечити гнучкий графік роботи, давати можливість для професійного розвитку та підтримувати здоровий баланс між роботою та особистим життям. Регулярні перерви і фізична активність допомагають зменшити напругу.

Розробка чат-систем в реальному часі зазвичай передбачає активну командну роботу. Це включає програмістів, тестувальників, дизайнерів інтерфейсів, а також інших фахівців. Така діяльність пов’язана з постійною комунікацією та координацією.

Взаємодія з іншими членами команди може бути як джерелом мотивації, так і стресу. Правильне управління проектами, чітке визначення завдань і вміння ефективно вирішувати конфлікти є важливими аспектами забезпечення здорового клімату в команді. Часто IT-фахівці повинні брати участь у численних нарадах, як онлайн, так і офлайн. Перевантаження наради можуть спричиняти

втому та дратівливість. Оптимізація комунікаційних процесів, використання відповідних інструментів для спілкування (наприклад, Slack, Microsoft Teams) дозволяють знизити цей стрес.

Розробка чат-систем, особливо тих, що функціонують в реальному часі, має свої особливості, які безпосередньо впливають на умови праці. Програмісти повинні працювати з великими обсягами даних, взаємодіяти з різними протоколами передачі інформації і тестувати взаємодію між користувачами в реальному часі.

Важливим аспектом є забезпечення безпеки даних, захист від хакерських атак, безпечне зберігання даних користувачів. Це вимагає постійного оновлення знань, уважності до змін у галузі безпеки та застосування найкращих практик програмування.

Також важливим є оптимізація програмного забезпечення для забезпечення високої продуктивності при одночасній роботі великої кількості користувачів. Для цього розробники повинні мати доступ до сучасних інструментів і технологій, що дозволяють зменшити ризики програмних помилок, збоїв і перегрузок серверів.

Охорона праці в ІТ-сфері є важливим аспектом, який стосується не тільки фізичного, а й психологічного стану працівників. Профілактика стресу, підтримка здоров'я співробітників та створення комфортних умов для ефективної роботи є основними чинниками успіху в розробці програмного забезпечення. У випадку розробки чат-систем у реальному часі особлива увага повинна приділятися як технічній безпеці, так і здоров’ю працівників, що дозволяє забезпечити не лише ефективну, але й безпечну розробку програмного продукту.

## 4.2 Питання щодо безпеки в надзвичайних ситуаціях

Надзвичайні ситуації – це події, що призводять до порушення нормальної життєдіяльності людей, забруднення навколишнього середовища, значних економічних і соціальних втрат. У світі існує велика кількість різноманітних надзвичайних ситуацій, що можуть виникнути внаслідок техногенних катастроф, природних катастроф, радіаційних, хімічних або біологічних інцидентів, а також військових конфліктів.

Надзвичайні ситуації можна класифікувати за різними ознаками:

– За характером: природні (землетруси, повені, урагани) та техногенні (аварії на виробництві, техногенні катастрофи).

– За масштабами: локальні, регіональні, національні та глобальні.

– За типами загрози: хімічні, радіаційні, біологічні, екологічні.

Кожен з типів НС має специфічні методи захисту, що залежать від виду загрози. Найбільш поширеними НС є техногенні катастрофи, що включають аварії на атомних електростанціях, хімічні вибухи, аварії на транспорті, які спричиняють значні загрози для здоров’я та життя людей.

Захист населення від наслідків надзвичайних ситуацій вимагає комплексного підходу, що включає інформаційне забезпечення. Важливим аспектом є своєчасне інформування населення про загрозу, використовуючи засоби масової інформації, спеціальні попереджувальні сигнали, сирени, телевізійні та радіопрограми.

В разі виникнення загрози на об'єкті необхідно організувати евакуацію людей в безпечні місця, забезпечити укриття в укриттях, що мають відповідну захист від радіаційних, хімічних або біологічних загроз.

У випадку НС необхідно організувати роботу медичних пунктів, забезпечити постраждалих швидкою допомогою, а також вжити заходів для запобігання епідеміям і хворобам. Важливо організувати роботу психологів та волонтерів, які допомагають населенню подолати стресові ситуації, що виникають під час катастроф.

Для ефективного захисту населення необхідно проводити постійну підготовку як для населення, так і для спеціалізованих служб: Проводяться тренування, інструктажі та курси для населення, щоб люди знали, як діяти в разі НС (як евакуюватися, куди йти, як захистити себе від певних загроз).

Спеціалізовані служби повинні мати навички надання першої медичної допомоги, організації евакуації, управління панікою та збереження безпеки громадян. Окремі структури повинні мати спеціальні засоби для надання першої допомоги, санітарної обробки постраждалих та лікування.

Радіаційний захист є важливою складовою охорони праці на підприємствах, що мають справу з радіоактивними матеріалами. Працівники, що виконують роботи в умовах радіаційного забруднення або потенційно небезпечних для здоров’я умовах, підлягають особливому захисту від радіаційного випромінювання.

Захист від радіації включає низку технологічних та організаційних заходів, що спрямовані на запобігання впливу радіації на організм людини. До основних принципів радіаційного захисту відносяться:

– Зменшення часу перебування на забруднених територіях.
– Збільшення відстані від джерела радіації.
– Захист за допомогою бар'єрів.

Працівники, які виконують роботи в умовах можливого радіаційного впливу, повинні проходити регулярне медичне обстеження та інструктажі щодо дій у разі надзвичайних ситуацій. Окрім того, необхідно дотримуватись усіх вимог. Робітники повинні проходити медичні огляди, щоб визначити наявність або відсутність впливу радіації на їхній організм. Окремо проводяться обстеження на наявність радіаційних захворювань. На виробничих об’єктах має проводитись моніторинг рівня радіації, а також оцінка потенційних ризиків для працівників.

Для роботи в зонах з підвищеною радіаційною небезпекою працівники повинні мати відповідний одяг, який забезпечує бар'єр від радіації, а також прилади для контролю за рівнем радіації. На виробничих об’єктах, що працюють

з радіоактивними матеріалами, повинні бути організовані спеціальні режими захисту.

Визначення максимально дозволених доз для працівників, контроль за їх виконанням. Використання технологічного обладнання, яке обмежує або усуває викиди радіації, а також має системи моніторингу.

Доступ на об'єкти, де присутні радіоактивні матеріали, повинний бути обмежений для сторонніх осіб. На таких об’єктах повинна бути розроблена чітка система допуску для працівників та проведення навчань.

### 4.3 Висновок до четвертого розділу

У четвертому розділі кваліфікаційнох роботи розглянуто основні аспекти охорони праці в ІТ-індустрії, зокрема ризики, пов'язані з роботою в офісах та при розробці програмного забезпечення, таких як чат-системи в реальному часі.

Особлива увага приділена фізичним та психологічним факторам, що впливають на здоров'я працівників: тривала робота за комп’ютером, стресові навантаження та вигорання.

Проаналізовано заходи для запобігання цим проблемам, такі як організація комфортних умов праці, використання сучасних технологій для захисту даних та підтримка психологічного здоров’я співробітників.

Подано рекомендації щодо створення безпечних умов для працівників в ІТ-сфері, акцентуючи на важливості регулярних медичних оглядів, психосоціальної підтримки та технічних заходів для забезпечення кібербезпеки.

Розглянуто питання захисту населення в разі виникнення надзвичайних ситуацій (НС), зокрема організацію евакуації, укриттів та медичної допомоги, а також заходи щодо психологічної підтримки постраждалих.

Проаналізовано основні типи НС, такі як природні, техногенні, хімічні, радіаційні та біологічні загрози, та визначено ключові елементи захисту, що включають своєчасне інформування громадян і належну підготовку спеціалізованих служб.

Подано рекомендації щодо радіаційного захисту працівників та службовців, що працюють у зонах з підвищеною радіаційною небезпекою. Це включає використання засобів індивідуального захисту, проведення медичних оглядів та дотримання стандартів безпеки на виробничих об'єктах.

# ВИСНОВКИ

У роботі досліджено сучасні підходи до оптимізації обробки запитів до баз даних, зокрема інтеграцію методів штучного інтелекту та застосування гібридних архітектур SQL/NoSQL. На основі проведеного аналізу та експериментального дослідження сформульовано такі основні висновки.

1. Систематизовано теоретичні засади оптимізації продуктивності баз даних. Традиційні методи – індексування, оптимізація запитів, секціонування, кешування та балансування навантаження – залишаються фундаментом ефективного керування базами даних, проте їх переважно статичний характер у динамічних середовищах виявляється недостатнім для підтримання продуктивності за зростання обсягів і складності даних.

2. Розглянуто особливості реляційної та нереляційної моделей даних. Показано, що SQL-бази вирізняються суворою узгодженістю та підтримкою транзакцій, тоді як NoSQL-бази – масштабованістю та гнучкістю; їх поєднання в гібридних архітектурах є логічною відповіддю на різноманітні потреби сучасних застосунків.

3. Проаналізовано підходи до застосування штучного інтелекту в обробці SQL-запитів. Огляд літератури засвідчив, що адаптивні моделі навчання здатні скоротити час виконання запитів на 40–45 %, а інтелектуальне індексування підвищує пропускну здатність на 25 %.

4. Систематизовано ключові проблеми інтеграції ШІ: якість даних як критичний чинник надійності прогнозів, складність вибору алгоритмів через гетерогенність запитів та виклики архітектурної інтеграції з наявними системами.

5. Обґрунтовано гібридну модель оптимізації, що поєднує правила-орієнтовані механізми з алгоритмами машинного навчання. Така модель забезпечує як адаптивність, так і стійкість, і здатна покращити загальну продуктивність приблизно на 35 %.

6. Експериментально підтверджено переваги інтелектуального підходу: скорочення часу виконання приблизно на 35 % для великих наборів даних, зниження використання пам'яті з 85 % до 60 %, зменшення дискових операцій з 15000 до 8000 та підвищення пропускної здатності приблизно на 63 %. Інтелектуальна модель досягла точності прогнозування 92 %.

7. Аналіз вартості планів виконання підтвердив економічну доцільність інтеграції ШІ: вартість інтелектуальних планів виявилася майже вдвічі нижчою за вартість традиційних.

Отримані результати мають практичну цінність для адміністраторів баз даних, інженерів даних та розробників високонавантажених застосунків, особливо в хмарних середовищах. Перспективними напрямами подальшої роботи є перевірка інтелектуальних методів у виробничих середовищах із непередбачуваними навантаженнями, поглиблене дослідження гібридних SQL/NoSQL-архітектур та розвиток адаптивних моделей, здатних до безперервного донавчання в реальному часі.

Підсумовуючи, слід наголосити, що інтеграція штучного інтелекту в обробку SQL-запитів є не просто технологічним удосконаленням, а парадигмальним зрушенням у способі керування базами даних. Перехід від статичних, правила-орієнтованих систем до динамічних інтелектуальних систем, здатних еволюціонувати разом із даними, які вони опрацьовують, відкриває нові можливості для підвищення ефективності, масштабованості та економічності інформаційних систем. Водночас успіх такого переходу критично залежить від якості даних, зваженого вибору алгоритмів та продуманої архітектурної інтеграції.

Дослідження також підтвердило, що традиційні методи оптимізації не втрачають своєї актуальності, а навпаки, формують необхідний фундамент, на якому будуються інтелектуальні надбудови. Найефективнішим виявляється не протиставлення двох підходів, а їх синергія в межах гібридної моделі, що поєднує надійність і передбачуваність класичних механізмів з адаптивністю та здатністю до навчання інтелектуальних методів. Саме такий збалансований

підхід, на думку автора, визначатиме розвиток систем керування базами даних у найближчій перспективі, особливо в умовах подальшого зростання обсягів даних та поширення хмарних і розподілених архітектур.

Загалом проведене дослідження досягло поставленої мети та розв'язало всі сформульовані задачі, надавши як теоретичне обґрунтування, так і емпіричне підтвердження доцільності інтеграції методів штучного інтелекту та гібридних архітектур у процеси оптимізації обробки запитів до баз даних.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Gadde H. Integrating AI into SQL Query Processing: Challenges and Opportunities. International Journal of Advanced Engineering Technologies and Innovations. 2022. Vol. 1, Issue 3. P. 194–219.

2. Youssef S. E. A. Optimizing Database Performance for Large-Scale Enterprise Applications: A Comprehensive Study on Techniques, Challenges, and the Integration of SQL and NoSQL Databases in Modern Data Architectures. Journal of Artificial Intelligence and Machine Learning in Management. 2023.

3. Abbasi M., Bernardo M. V., Váz P., Silva J., Martins P. Optimizing database performance in complex event processing through indexing strategies. Data. 2024. Vol. 9. P. 93.

4. Anderson A., Takahashi H. Trends in Database Technology: Cloud-Based and Distributed Solutions. London: Pearson Education, 2017.

5. Brown M., Xu J. Challenges of ensuring data consistency in mixed-database environments. IEEE, 2016. P. 65–74.

6. Chen M., Thompson D. L., Wagner K. Hybrid SQL/NoSQL database systems: Strategies for modern enterprise architectures. ACM Transactions on Database Systems (TODS). 2017. Vol. 42. P. 1–35.

7. Fischer S., Ivanova N. Modern Data Architecture for Enterprise Applications: Integrating SQL and NoSQL. Boston: Addison-Wesley Professional, 2014.

8. Garcia J., Evans M. Scaling traditional SQL databases for large-scale enterprise applications. IEEE, 2013. P. 45–52.

9. Ivanov S., Robertson S. J. Effective partitioning strategies for large-scale databases. IEEE, 2014. P. 305–316.

10. Jani Y. The role of SQL and NoSQL databases in modern data architectures. International Journal of Core Engineering & Management. 2021. Vol. 6. P. 61–67.

11. Jani Y. Optimizing database performance for large-scale enterprise applications. International Journal of Science and Research (IJSR). 2022. Vol. 11. P. 1394–1396.

12. Johnson C., Heiden A. Database Performance Optimization: Strategies for Enterprise Applications. Sebastopol: O'Reilly Media, 2016.

13. Martinez C., Weber E. Optimizing queries in hybrid SQL/NoSQL database environments. ACM, 2012. P. 333–342.

14. Nguyen H., Gomez M. Enterprise Data Management with SQL and NoSQL Databases. Berlin: Springer, 2013.

15. Raj P., Mendes C. Scalability challenges in traditional and modern database systems. IEEE Transactions on Knowledge and Data Engineering. 2013. Vol. 25. P. 1830–1842.

16. Rossi L., Schmidt A. Advanced query optimization techniques for large enterprise databases. Journal of Computing and Information Technology. 2016. Vol. 24. P. 99–112.

17. Singh A., Lopez J. Caching strategies to enhance database performance in large-scale systems. ACM, 2015. P. 1621–1624.

18. Smith J., Zhang E., Müller S. Optimizing database performance for large-scale enterprise applications. Journal of Database Management. 2015. Vol. 28. P. 12–27.

19. Wang L., Davis R. T. SQL vs. NoSQL: Performance trade-offs in large-scale enterprise systems. Springer, 2014. P. 213–224.

20. Hoffman G., Zhao X. Future trends in database optimization for large-scale enterprises. IEEE, 2017. P. 112–120.

21. Карпінський М. Методи безпечної роботи при використанні SQL-сервера Oracle. Вісник ТДТУ. 2002. Т. 7, № 2. С. 95–100.

22. Голінько В. І. Охорона праці в галузі інформаційних технологій: навч. посіб. / В. І. Голінько, М. Ю. Іконніков, Я. Я. Лебедєв; М-во освіти і науки України, Держ. вищий навч. закл. "Нац. гірн. ун-т". - Дніпропетровськ: НГУ, 2015. - 246 с.

23. Гандзюк М.П. Основи охорони праці: Підручник. 4-е вид./Гандзюк М.П., Желібо Є.П., Халімовський М.О. - Київ: Каревела, 2008. – 384с.

24. Техноекологія та цивільна безпека. Частина «Цивільна безпека»: Навчальний посібник; укл.: Стручок В. С. Тернопіль: ФОП Паляниця В.А., 2022. 150 с.

25. Безпека в надзвичайних ситуаціях. Методичний посібник для здобувачів освітнього ступеня «магістр» всіх спеціальностей денної та заочної (дистанційної) форм навчання / укл.: Стручок В. С. Тернопіль: ФОП Паляниця В. А., 2022. 156 с.

26. Умови праці працівників, які використовують у роботі персональні комп'ютери. Zolochiv.Net. URL: https://zolochiv.net/umovy-pratsi-pratsivnykiv-iaki-vykorystovuiut-u-roboti-personal-ni-komp-iutery/ (дата звернення: 25.10.2024).

# ДОДАТКИ

Додаток А

Програмний код

**Лістинг А.1. Налаштування експериментального середовища (Python + PostgreSQL)**

```
import psycopg2
import tensorflow as tf
import time

# Підключення до хмарної бази PostgreSQL
conn = psycopg2.connect(
    host="db-host", dbname="benchmark",
    user="analyst", password="<надається окремо>"
)
cur = conn.cursor()

# Увімкнення збору статистики виконання
cur.execute("LOAD 'pg_stat_statements';")
```

*Лістинг А.1 – підрозділ 3.1*

**Лістинг А.2. Вимірювання часу виконання запиту**

```
def measure_execution_time(cursor, query, trials=30):
    times = []
    for _ in range(trials):
        t_start = time.perf_counter()
        cursor.execute(query)
        cursor.fetchall()
        t_end = time.perf_counter()
        times.append(t_end - t_start)
    return sum(times) / len(times)   # середній час, с

sql = """
SELECT c.region, COUNT(*) AS orders, AVG(o.total) AS avg_total
FROM customers c
JOIN orders o ON o.customer_id = c.id
GROUP BY c.region
ORDER BY orders DESC;
"""
et = measure_execution_time(cur, sql)
print(f"Середній час виконання: {et:.3f} с")
```

*Лістинг А.2 – підрозділ 3.1*

**Лістинг А.3. Аналіз плану виконання запиту**

```
-- Отримання плану виконання з фактичними витратами
EXPLAIN (ANALYZE, BUFFERS, FORMAT JSON)
SELECT c.region, COUNT(*) AS orders, AVG(o.total) AS avg_total
FROM customers c
```

```
JOIN orders o ON o.customer_id = c.id
WHERE o.created_at >= '2024-01-01'
GROUP BY c.region
ORDER BY orders DESC;

-- Створення складеного індексу для прискорення з'єднання
CREATE INDEX idx_orders_customer_date
    ON orders (customer_id, created_at);
```

*Лістинг А.3 – підрозділи 1.3, 3.2*

## Лістинг А.4. Гібридна модель: прогнозування плану виконання

```
import numpy as np
from sklearn.ensemble import RandomForestClassifier

# Ознаки запиту: к-сть з'єднань, селективність, обсяг таблиць
X_train = np.array([...])   # історичні ознаки запитів
y_train = np.array([...])   # мітки оптимальних планів

model = RandomForestClassifier(n_estimators=200, random_state=42)
model.fit(X_train, y_train)

def choose_plan(query_features, confidence_threshold=0.8):
    proba = model.predict_proba([query_features])[0]
    best = proba.argmax()
    if proba[best] >= confidence_threshold:
        return ("ml_plan", best)          # довіряємо моделі
    return ("rule_based_plan", None)       # відкат до правил
```

*Лістинг А.4 – підрозділ 2.4*

## Лістинг А.5. Обчислення метрик якості моделі

```
from sklearn.metrics import (accuracy_score, precision_score,
                             recall_score, f1_score)

y_pred = model.predict(X_test)

pa = accuracy_score(y_test, y_pred) * 100
p  = precision_score(y_test, y_pred, average='weighted') * 100
r  = recall_score(y_test, y_pred, average='weighted') * 100
f1 = f1_score(y_test, y_pred, average='weighted') * 100

print(f"Точність (PA): {pa:.0f}%")
print(f"Влучність (P): {p:.0f}%")
print(f"Повнота (R):   {r:.0f}%")
print(f"F1-міра:       {f1:.0f}%")
```

*Лістинг А.5 – підрозділ 3.3*

## Лістинг А.6. Обчислення відсоткового покращення продуктивності

```
def improvement(traditional, ai_optimized):
    return (traditional - ai_optimized) / traditional * 100
```

```
# Приклад для набору 1 000 000 записів
et_trad, et_ai = 85.3, 55.6
print(f"Покращення часу виконання: {improvement(et_trad, et_ai):.1f}%")

# Пропускна здатність (100 000 записів)
tp_trad, tp_ai = 120, 196
print(f"Зростання пропускної здатності: "
      f"{(tp_ai - tp_trad) / tp_trad * 100:.1f}%")
```

*Лістинг А.6 – підрозділи 3.2, 3.5*