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 характеристик продуктивності архітектур реляційних та нереляційних систем керування базами даних

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

_____ Мельничук С.А.
(підпис) (прізвище та ініціали)

Керівник _____ Никитюк В.В.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Дуда О.М.
(підпис) (прізвище та ініціали)

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

Рецензент _____ _____
(підпис) (прізвище та ініціали)

Тернопіль
2024

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)
"26" грудня 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Магістр
(назва освітнього ступеня)

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

студенту Мельничук Святослав Анатолійович
(прізвище, ім'я, по батькові)

1. Тема роботи **Аналіз характеристик продуктивності архітектур реляційних та нереляційних систем керування базами даних**

Керівник роботи **к.т.н., доц. Никитюк В.В.**
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від "*29*" *листопада* 2024 року № *4/7-1142*

2. Термін подання студентом завершеної роботи 26 грудня 2024 р.

3. Вихідні дані до роботи Літературні джерела з тематики роботи

4. Зміст роботи (перелік питань, які потрібно розробити)
ВСТУП; 1 ОГЛЯД ЛІТЕРАТУРНИХ ДЖЕРЕЛ ЗА ТЕМОЮ РОБОТИ; 1.1 Опис стану проблеми в досліджуваній області; 1.2 Аналіз літературних джерел; 1.3 Моделювання даних NoSQL MongoDB; 2 АНАЛІЗ ПОСЛУГИ DBaaS ДЛЯ НЕРЕЛЯЦІЙНИХ БАЗ ДАНИХ; 2.1 Ефективність переносу даних між різними нереляційними СКБД; 2.2 Практична реалізація принципу CAP для нереляційних баз даних; 3 ОСОБЛИВОСТІ РЕАЛІЗАЦІЇ НЕРЕЛЯЦІЙНИХ БАЗ ДАНИХ; 3.1 Реалізація розподілу даних; 3.2 Приклад порівняння продуктивності реляційної і нереляційної СКБД; 4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ; 4.1 Охорона праці та її актуальність в IT-сфері; 4.2 Шкідлива дія шуту та вібрації і захист від неї; ВИСНОВКИ; СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ; ДОДАТКИ

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
Тема роботи; Мета, задачі дослідження; Об'єкт і предмет дослідження; Характеристики для порівняння СКБД; SQL vs NoSQL; Концепції ACID та BASE; Концепція CAP для NoSQL СКБД; ;Виконання SQL-запитів SELECT та INSERT; Oracle RDBMS vs MongoDB; GIS-системи; Проблема міграції даних між CSP; Результати порівняння MySQL та MongoDB; ВИСНОВКИ

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.		
Безпека в надзвичайних ситуаціях	Клепчик В.М., ст. викл.		

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	14.11.24-15.11.24	*Виконано*
2.	Підбір наукових джерел по темі роботи	16.11.24-20.11.24	*Виконано*
3.	Переклад та опрацювання наукових джерел	20.11.24-23.11.24	*Виконано*
	по темі кваліфікаційної роботи		
4.	Виконання дослідження щодо теми	24.11.24-10.12.24	*Виконано*
	Кваліфікаційної роботи		
5.	Оформлення першого розділу	11.12.24-12.10.24	*Виконано*
6.	Оформлення другого розділу	12.12.24-13.12.24	*Виконано*
7.	Оформлення третього розділу	13.12.24-14.12.24	*Виконано*
8.	Виконання завдання до підрозділу "Охорона праці"	08.12.24-09.11.24	*Виконано*
9.	Виконання завдання до підрозділу "Безпека в	10.12.24-12.12.24	*Виконано*
	надзвичайних ситуаціях"		
10.	Оформлення кваліфікаційної роботи	12.12.24-31.12.24	*Виконано*
11.	Нормоконтроль	14.12.24-15.12.24	*Виконано*
12.	Перевірка на плагіат	15.12.24	*Виконано*
13.	Попередній захист кваліфікаційної роботи	18.12.24	*Виконано*
14.	Захист кваліфікаційної роботи	27.12.2024	

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

Керівник роботи _____ Никитюк В.В.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

"Аналіз характеристик продуктивності архітектур реляційних та нереляційних систем керування базами даних" // Кваліфікаційна робота освітнього рівня "Магістр" // Мельничук Святослав Анатолійович // Тернопільський національний технічний університет ім. І. Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра комп'ютерних наук, група СНмз-61 // Тернопіль, 2024 // с. – 66, рис. – 11, табл. – 2, джерел – 53.

Ключові слова: бази даних; SQL; NoSQL; Big Data; ACID; DBaaS

Правильна архітектура програмного забезпечення є ключовою для вирішення складної задачі обробки великих обсягів даних у системах реляційних і нереляційних баз даних. Реляційні бази даних (SQL) були розроблені для організації даних та забезпечення можливості горизонтального масштабування. Нереляційні бази даних (NoSQL), своєю чергою, підтримують горизонтальне масштабування і можуть ефективно обробляти великі обсяги неструктурованих даних. Вибір найбільш відповідної парадигми залежить від організаційних вимог. Різниця в проектуванні баз даних є основною ознакою між реляційними та нереляційними базами даних. Крім того, кожен тип нереляційної бази даних використовує різні моделі поєднання підходів. Це ускладнює перенос даних між різними хмарними сховищами для користувачів хмарних платформ (Cloud Service Provider – CSP). Існують різні моделі, якими управляють різні хмарні сервіси (IaaS, PaaS, SaaS і DBaaS).

Метою цієї роботи є дослідження робіт, що стосуються переносимості та сумісності даних у хмарних сховищах, а також програмної архітектури з використанням реляційних та нереляційних баз даних. Багато досліджень, що порівнюють їх можливості, зокрема реляційну Oracle RDBMS та нереляційну документо-орієнтовану (MongoDB), за такими характеристиками, як

масштабування, продуктивність, доступність, узгодженість і сегментація, були представлені в дослідженнях, які свідчать, що нереляційні бази даних, завдяки своїм спеціалізованим структурам, можуть бути оптимальним вибором для аналізу великих даних, тоді як реляційні бази даних краще підходять для обробки онлайнових транзакцій (OLTP).

ANNOTATION

"Analysis of Performance Characteristics of Relational and Non-Relational Database Management System Architectures" // Master's degree qualification paper // Melnychuk Sviatoslav // Ternopil Ivan Puluj National Technical University, Faculty of Computer Information Systems and Software Engineering, Computer Science Department, group СНмз-61 // Ternopil, 2024 // p. – 66, fig. – 11, tables – 2, references – 53.

Key words: data bases; SQL; NoSQL; Big Data; ACID; DBaaS

Proper software architecture is key to solving the complex task of processing large volumes of data in relational and non-relational database systems. Relational databases (SQL) were designed for data organization and to enable horizontal scaling. Non-relational databases (NoSQL), on the other hand, support horizontal scalability and can efficiently process large volumes of unstructured data. The choice of the most appropriate paradigm depends on organizational requirements. The difference in database design is the primary distinguishing factor between relational and non-relational databases. Furthermore, each type of non-relational database uses different models that combine various approaches. This complicates the transfer of data between different cloud storage services for users of cloud platforms (Cloud Service Providers – CSP). Various cloud services (IaaS, PaaS, SaaS, and DBaaS) govern different models.

The purpose of this work is to examine studies related to the portability and compatibility of data in cloud storage, as well as software architecture using relational and non-relational databases. Numerous studies comparing the capabilities of these systems, including the relational Oracle RDBMS and the non-relational document-oriented MongoDB, have been presented in research. These studies indicate that non-relational databases, with their specialized structures, can be the optimal choice for big data analytics, while relational databases are better suited for online transaction processing (OLTP).

ЗМІСТ

ВСТУП

Актуальність задачі.

Архітектура конкретного програмного забезпечення враховує такі нефункціональні характеристики, як надійність, зручність використання, масштабованість, продуктивність, сумісність, портативність, адаптивність і шардинг (розподіл) даних. Дослідження якості програмних продуктів в ТНТУ проводились в, наприклад, роботах [1, 2, 3]. Проте вклад в якість кінцевого продукту в залежності від вибраного типу системи керування базами даних в цих роботах не розглядався. Тому дане дослідження є актуальним.

Між набором атрибутів якості завжди є компроміси, і архітектор програмного забезпечення стикається з важким завданням їх балансування. Системи великих даних [4] внутрішньо розподілені. Труднощі з доступністю та узгодженістю даних виникають через розподіл і реплікацію даних у великих системах даних. У зв'язку зі збільшенням кількості додатків даних технології баз даних зазнали значних змін. Протягом більш ніж десятиліття бази даних NoSQL зросли експоненціально, хоча класична автоматизація баз даних збереглася. Традиційний макет передбачає жорстку структуру схеми, що призводить до неясності масштабування та перешкоджає модифікації даних у кластерах. Навпаки, бази даних NoSQL підтримують прості прототипи. Основні властивості проектів баз даних NoSQL включають:

– Безсхемна структура.

– Дозволяє ефективно та динамічно розвивати представлення даних.

– Горизонтальне масштабування за допомогою колекцій реплікації даних і сегментування над масивними кластерами.

За останні роки численні організації накопичили величезні обсяги даних, які реляційні бази даних не можуть ефективно обробити. За останні чотири десятиліття відбулося величезне зростання використання реляційних баз даних. Вони відповідають атрибуту ACID (доступність, послідовність, ізоляція та довговічність) і призначені для структурованих даних. Хоча «великі дані»

включають інструменти та технології, які керують величезними обсягами даних у будь-якому масштабі, ці інструменти та технології є масштабованими. Великі дані включають 5V (обсяг, швидкість, різноманітність, правдивість і цінність) і величезну кількість неструктурованих даних різного характеру. Численні фреймворки, включаючи Hadoop/MapReduce, Spark, Flink, використовуються для обробки великих даних [5].

Мета роботи.

Метою роботи є дослідження слабких та сильних сторін реляційних та нереляційних баз даних та продуктивності програмних систем, побудованих на кожному з цих типів баз даних.

Для досягнення мети потрібно вирішити наступні задачі:

1. Виконати огляд літератури, пов’язаної з базами даних SQL і NoSQL.
2. Оцінити сильні та слабкі сторони баз даних SQL і NoSQL на основі даних, зібраних і проаналізованих у дослідженнях літератури.
3. Знайти області, в котрих ще недостатньо проведені дослідження.
4. Для досягнення основної мети дослідження потрібно вирішити наступні питання дослідження:

– обґрунтувати використання NoSQL для задач опрацювання великих даних;

– чому бази даних NoSQL відповідають властивості BASE замість властивості ACID реляційних бази даних?

– дослідити ефективність DBaaS для забезпечення взаємодії та переносимості даних у різних NoSQL базах даних.

Об’єкт дослідження: процеси визначення продуктивності програних систем на основі реляційних та нереляційних баз даних.

Предмет дослідження: програмні системи з базами даних.

Наукова новизна отриманих результатів.

– Це дослідження пов’язане з оцінками архітектури баз даних SQL і NoSQL, можливостей масштабування та аналізу продуктивності, зокрема Oracle

RDBMS і MongoDB як представника документо-орієнтованої БД. Крім того, досліджується рух даних між різними базами даних на кількох хмарних платформах.

– У цій роботі визначено прогалини в дослідженнях пов’язаних архітектур та їх причини.

Практичне значення отриманих результатів.

Дана кваліфікаційна робота може служити в якості основи для прийняття архітектурних рішень в інформаційних системах з базами даних для вибору типу СКБД у випадку критичних вимог до продуктивності системи.

Апробація результатів та особистий внесок здобувача.

Основні положення роботи доповідались, розглядались та обговорювались на науковій конференції Тернопільського національного технічного університету імені Івана Пулюя. Результати кваліфікаційної роботи опубліковані у тезах студентської наукової конференції "Актуальні задачі сучасних технологій – 2024", яка проводилась у ТНТУ.

1 ОГЛЯД ЛІТЕРАТУРНИХ ДЖЕРЕЛ ЗА ТЕМОЮ РОБОТИ

Тема продуктивності та оптимізації запитів SQL для корпоративних, виробничих, паралельних баз даних і великих даних [6] привернула підвищену увагу в останні роки. Неефективні та неоптимізовані запити можуть споживати ресурси системи та сервера, що призведе до блокування бази даних і проблем із втратою даних. Видобуток інформації передбачає вилучення фактів і структури логічної кореляції з вихідного набору даних, на відміну від самої інформації. Оптимізація запитів означає вибір оптимальної стратегії виконання запитів з мінімальними витратами та споживанням ресурсів системи. Алгоритми інтелектуального аналізу даних виконують поглиблені та обширні запити до бази даних, щоб витягти шаблони та знання з комплексних даних [7]. Альтернативні методології, такі як XML та об’єктні бази даних, ніколи не досягали такого рівня популярності, як технологія RDBMS.

Протягом останнього десятиліття наука та онлайнові торгові сервіси поставили під сумнів «універсальний » характер технології магазину даних (data store). Цей напрямок призвів до розробки нової альтернативної системи баз даних, відомої як NoSQL, що означає «Не тільки SQL». NoSQL описує використання веб-розробниками нереляційних баз даних [8]. Перерахуємо основні аспекти (характеристики) нереляційних баз даних:

1. Розподілена архітектура.
2. Ліцензія з відкритим кодом.
3. Горизонтальна масштабованість.
4. Відсутність схеми даних.
5. Легкий процес реплікації.
6. Простий програмний інтерфейс доступу (API).

У роботі [9] представлено принципи, котрим в ідеалі мала би відповідати кожна система керування нереляційними базами даних: цілісність (consistency), доступність (availability), фрагментація (partition tolerant) – CAP (див. рис. 1.1).

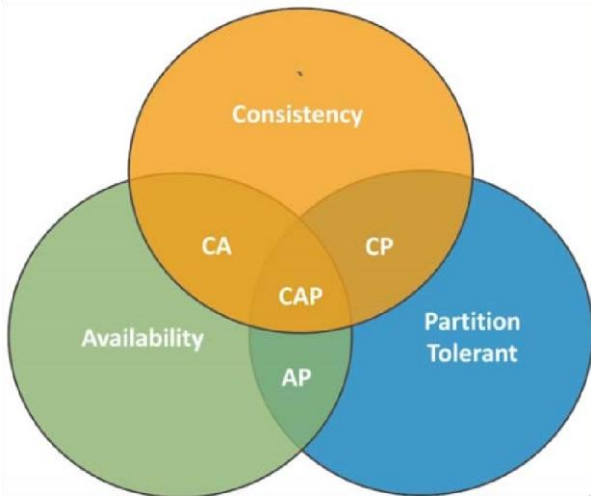

Рисунок 1.1 – Концепція CAP для нереляційних баз даних

Теоретично неможливо одночасно в рамках однієї системи виконати всі три вимоги. Тому на практиці кожна NoSQL СКБД реалізує свої два з трьох принципів CAP.

Наведемо переваги NoSQL баз даних:

– Обсяг: дані в стані спокою – від терабайтів до ексабайтів наявних даних для обробки.

– Швидкість: дані в русі – потокові дані, від мілісекунд до секунд для відповіді.

– Змінність: дані в багатьох формах – структуровані, неструктуровані, текстові тощо.

– Не побудовані на основі таблиць і не використовують SQL для обробки даних.

– Схема містить один з принципів: ключ-значення, документ, стовпець, графік тощо.

– Альтернатива традиційним реляційним базам даних.

– База даних для обробки неструктурованих і невпорядкованих даних.

– Корисно для роботи з великими наборами розподілених даних.

Бази даних NoSQL відрізняються від реляційних баз даних і мають власні моделі та архітектури. Зберігаючи бази даних NoSQL у хмарі, необхідно дотримуватися обережності через різноманітність цих баз даних і необхідність

взаємодії, переносимості та заходів безпеки. Постачальники хмарних послуг (CSP) пропонують масштабованість, доступність і конфіденційність, але важливо зашифрувати конфіденційні дані користувача перед наданням доступу до CSP. Це може бути складно через різноманітність баз даних NoSQL. База даних як послуга (DBaaS) – це хмарна платформа, яка перетворює традиційні архітектури на хмарні.

1.1 Опис стану проблеми в досліджуваній області

Поширення технології великих даних призвело до потреби в масштабованих системах, які можуть ефективно обробляти величезні обсяги даних. Реляційні бази даних, які базуються на SQL, можуть певною мірою керувати структурованими, напівструктурованими та неструктурованими даними, але мають обмеження щодо масштабованості. З іншого боку, бази даних NoSQL, які мають властивість BASE і можуть розширювати свою ємність зберігання по горизонталі, краще підходять для керування великими обсягами даних і адаптації до змін у типі та структурі даних.

Існує чотири типи баз даних NoSQL – бази даних типу "ключ-значення", бази даних документів, стовпців і графів – кожна зі своїми відмінними функціями та застосуваннями. Наприклад, графова база даних NoSQL зберігає інформацію у вузлах, а не в таблицях, і зберігає асоціації (з’єднання) між вузлами, що вимагає постійного часу виконання. Це дає їй перевагу перед базами даних SQL при обробці сильно взаємопов’язаних даних.

MongoDB є прикладом бази даних NoSQL, яка ефективно керує величезними даними завдяки чудовим характеристикам масштабованості. Однак перетворення баз даних OLTP на MongoDB може спричинити такі проблеми, як унікальні індекси, складені ключі, невідповідність даних і дублювання даних через складність їхньої схеми.

Іншою важливою проблемою при передачі даних до хмарних сховищ є захист особистої інформації користувачів. Поточні проекти постачальників

хмарних послуг (Cloud Service Provider – CSP) розробляються без урахування проблем сумісності та переносимості, що ускладнює пропонування та створення уніфікованого хмарного рішення для моделей NoSQL.

Ця робота є оглядом літературних джерел, який спрямований на синтез високоякісних первинних досліджень на основі даних, пов’язаних із конкретними дослідницькими питаннями. Дане дослідження зосереджено на оцінці архітектури бази даних SQL і NoSQL та оцінці продуктивності, а також на переносимості даних і сумісності між кількома хмарними платформами. Характеристиками, які відрізняють поточне дослідження від існуючих, є його системний процес збору даних, вичерпний перелік охоплених досліджень, сфокусований обсяг дослідження, а також його детальна класифікація та аналіз вибраних досліджень.

Критерії пошуку літературних джерел для первинних досліджень включають виявлення та збір літератури, яка відповідає критеріям включення та виключення. Щоб досягти цього, використовувалися різні методи пошуку, такі як пошук в електронній базі даних, пошук вручну, а також пошук у відповідних журналах і матеріалах конференцій.

На першому етапі ми отримали набір асоційованих рядків пошуку. Ми використали отриманий набір рядків, щоб знайти відповідні документи. Для пошуку пов’язаних статей були обрані найбільші бази даних:

- DBLP.
- IEEExplore (ieeexplore.ieee.org).
- Google Scholar.
- Цифрова бібліотека ACM (dl.acm.org).
- Springer (Springerlink.com).
- Elsevier (sciencedirect.com).
- Інтернет-бібліотека Wiley (onlinelibrary.wiley.com).

Було знайдено сотні статей під час процесу пошуку за назвами статей, анотаціями та ключовими словами. Серед них було приблизно п'ята частина

статей, які відповідали критеріям включення та виключення після видалення дублікатів.

Були створені різні комбінації рядків пошуку. Розроблені пошукові рядки було запущено на згаданих пошукових ресурсах для ідентифікації пов’язаної літератури:

- «SQL і NoSQL».
- «SQL або NoSQL».
- «Реляційна база даних і база даних документів»..
- «Реляційна база даних або база даних документів»
- «Реляційна база даних і база даних документів NoSQL».
- «Реляційні бази даних і MongoDB».
- «Реляційні бази даних або MongoDB».
- «Порівняння Oracle і MongoDB».
- «Порівняння баз даних SQL і NoSQL».
- «Переваги MongoDB перед RDBMS».
- «Реляційні та нереляційні бази даних».
- «Переносність та сумісність хмарних даних».

До опрацювання брались всі пов’язані документи, застосовуючи критерії включення/виключення. Далі ми оцінили якість і характеристики статей відповідно до наших дослідницьких запитань і завершили вибраний список.

Більшість робіт базувалися на емпіричних дослідженнях, які включають:

- Мету дослідження.
- Пов’язану література та підтримувані теорії.
- Перевірені гіпотези.
- Запропонований метод, дизайн, підхід, вимірювання та збір даних.
- Аналіз результатів даних.
- Висновок.

Згідно з нашими дослідницькими питаннями були включені наступні типи робіт, що відповідають критеріям:

– IC1: оглядові документи.

– IC2: запропоновані нові методи та підходи, що стосуються теми роботи.

– IC3: ефективні методи дослідження, представлені в запропонованому дослідженні.

Дослідницькі роботи були виключені з дослідження, коли вони відповідали таким критеріям:

– EC1: документи, які не стосуються згаданої області.

– EC2: нерелевантні документи.

– EC3: деякі статті на основі назви та анотації.

– EC4: нерецензовані матеріали та документи.

– EC4: дубльовані статті.

З кожної вибраної статті було отримано такі дані:

– Назва статті.

– Анотація.

– Джерело статті (журнал або конференція).

– Рік видання.

– Класифікація (тип, сфера застосування).

– Спорідненість із запропонованим дослідженням.

– Запропоновані цілі та проблеми дослідження.

– Анотація та метод.

Було обрано більшість емпіричних статей. Емпіричні дослідження складалися з наступних категорій оцінок статей, оцінок обговорень, експериментів та оглядів існуючих методів (SQL і NoSQL).

Крім того, інформація про кожну з вилучених даних була впорядкована та зведена в таблицю відповідно до питань дослідження таким чином:

1. Розгляд великих даних (структурованих і неструктурованих даних): для чого потрібна NoSQL?

2. Чому база даних NoSQL слідує властивості BASE замість властивості ACID бази даних SQL?

3. Чи ефективно DBaaS забезпечує взаємодію та переносимість даних у різних базах даних NoSQL?

В загальному аналізувалась множина робіт по джерелах отримання, показана на рисунку 1.2

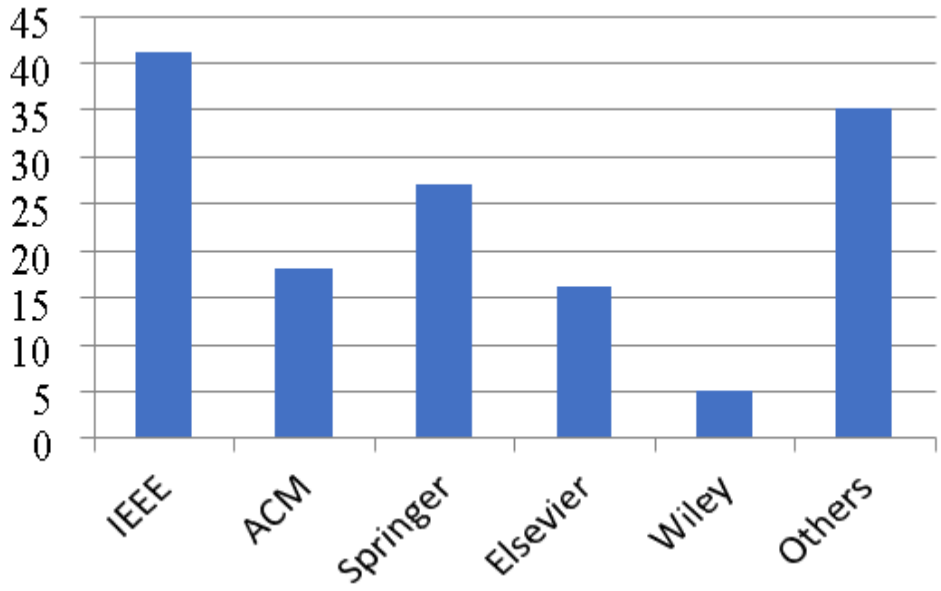

Рисунок 1.2 – Розподіл досліджуваних літературних джерел за виданням

На рисунку 1.3 показано перелік баз даних, котрі були предметом вибраних для огляду літературних джерел.

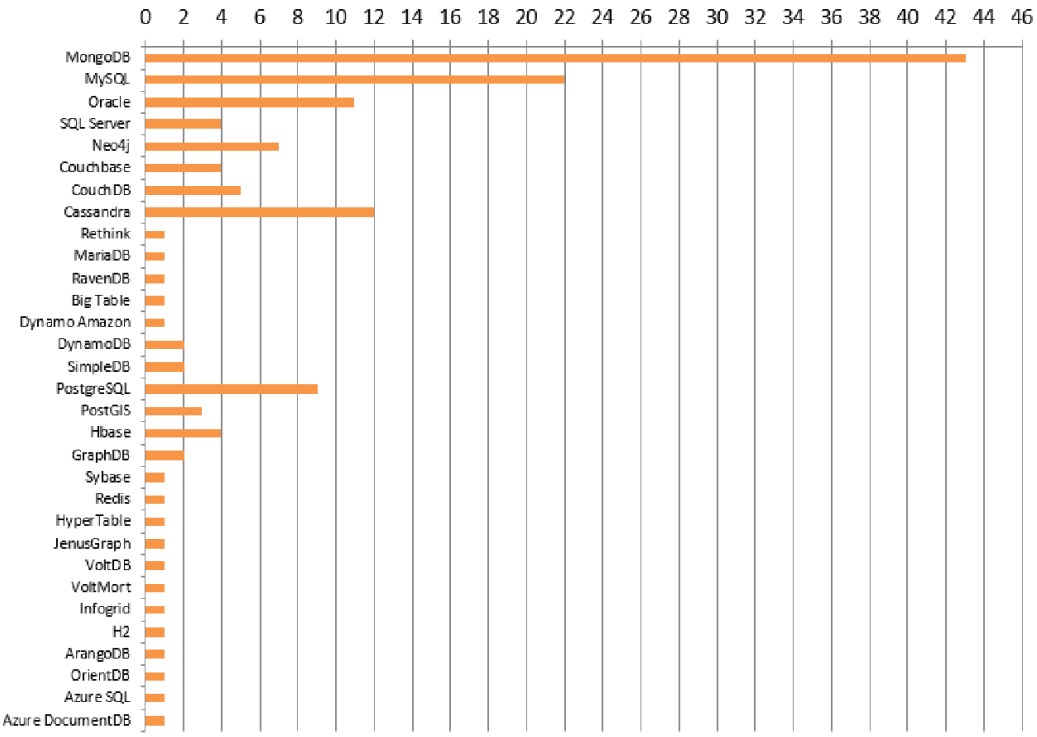

Рисунок 1.3 – Бази даних, використані в окремих дослідженнях

Під час аналізу ми згрупували всі отримані стратегії за категоріями та узагальнили наші висновки в три групи: методи дослідження, фази процесу дослідження та оцінка.

1.2 Аналіз літературних джерел

Емпіричні дослідження та статті належать до категорій порівнянь, оцінок, експериментів, категоризацій, діалогів та опитувань. Статті, які, на нашу думку, мали відношення до тем нашого дослідження, були обрані з літератури та оцінені у світлі наших гіпотез і критеріїв відбору.

1. Розгляд великих даних (структурованих і неструктурованих даних): для чого потрібні бази даних NoSQL? Запитання

2. Чому бази даних NoSQL використовують властивість BASE замість властивості ACID бази даних SQL?

Моделювання бази даних допомогло нам передбачити типи даних, які в ній зберігатимуться, і спосіб їх зберігання.

NoSQL, абревіатура від «Not only SQL» [10], – це підхід до управління базами даних, який чудово підходить для обробки великих обсягів неструктурованих даних і аналітики великих даних. З цими базами даних можна використовувати всі види різних мов запитів, і вони не дотримуються суворої, заздалегідь визначеної структури схеми. Однак протягом останніх кількох десятиліть реляційні бази даних використовували галузевий стандарт мови SQL. Документно-орієнтовані бази даних є підмножиною баз даних NoSQL. Бази даних, які зосереджені на зберіганні та отриманні документів, включають MongoDB і CouchDB. Бази даних цього типу використовуються для зберігання та адміністрування даних, які переважно базуються на документах. Складні формати даних, такі як JSON, BSON, XML і PDF, використовуються для зберігання інформації в документоорієнтованих базах даних.

І MongoDB, і CouchDB є безкоштовними та з відкритим кодом; однак MongoDB [11] більше підходить для розподілених налаштувань і JSON. Дослідникививчали кілька різних функцій бази даних NoSQL. Однією з популярних баз даних, яка була створена спеціально з урахуванням JSON, є MongoDB, яка використовує мову програмування C++. MongoDB використовує динамічну схему структури замість попередньо визначених статичних документів. Аналіз і пошук даних є швидкими та точними завдяки вдосконаленій обробці запитів, підтримці індексування та агрегації в пам’яті. Окрім забезпечення повної безпеки, він надає утиліти відновлення та резервного копіювання. Бази даних SQL і NoSQL показані на рисунку 1.4 разом із відповідною структурою зберігання даних.

І реляційні бази даних, наприклад SQL Server, і NoSQL (MongoDB, Neo4j) були предметом численних досліджень, які порівнювали їх структуру, дизайн і продуктивність [12 – 16]. Ці дослідження використовували запропоновані методи для ситуацій, пов’язаних із базами даних SQL і NoSQL, і аналізували результати. Бази даних NoSQL мають різні функції та програми. Бази даних NoSQL не можуть гарантувати властивість ACID через здатність масштабуватись горизонтально. У цьому плані база даних графів Neo4j [17 , 18]

є чудовим варіантом. Показано, що бази даних графів ефективно організовують і зберігають дані зі складними залежностями. Основною метою MongoDB є зберігання даних у вигляді документів.

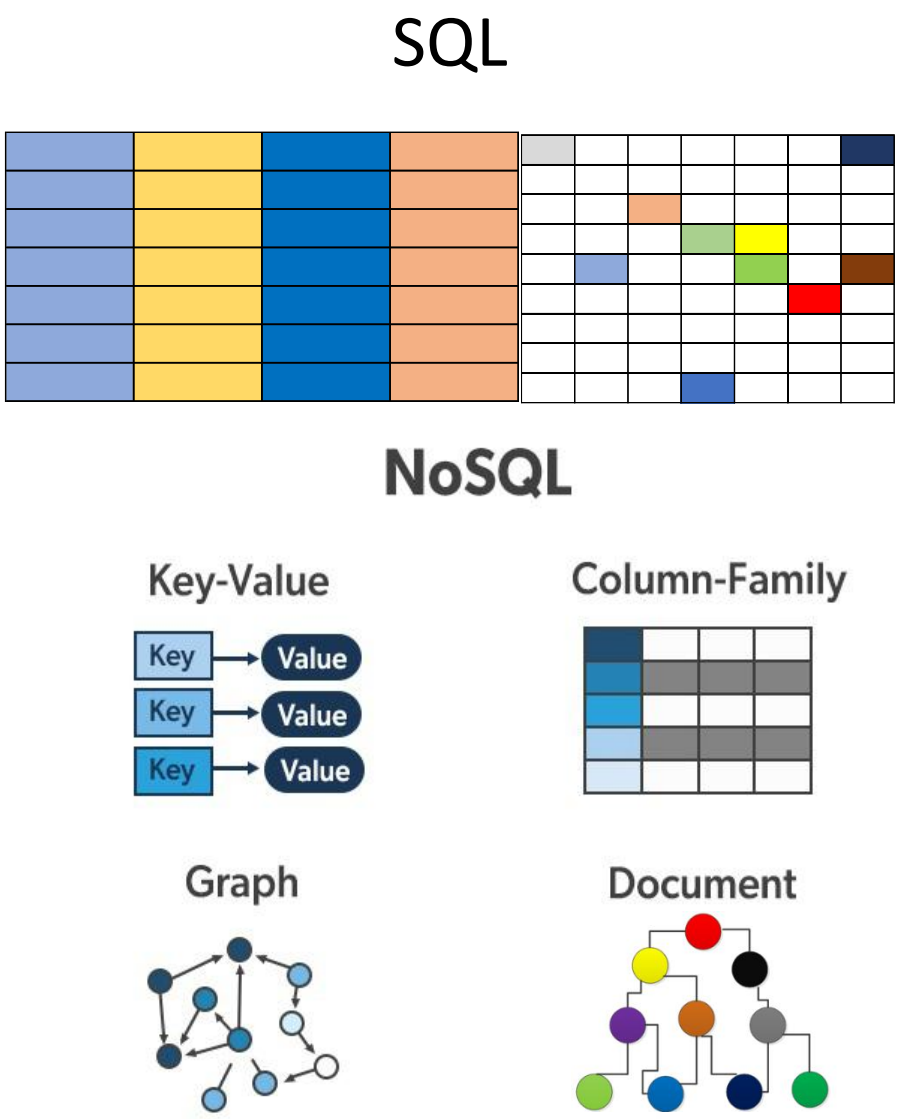

Рисунок 1.4 – Структури зберігання баз даних SQL і NoSQL

MongoDB використовує формат BSON, який є відгалуженням JSON.

Маючи величезні дані, MongoDB працює добре і неодноразово демонстрував себе. Дослідження в [19] порівнювало базу даних NoSQL MongoDB з реляційною базою даних Postgre SQL для служби соціальних мереж і даних датчиків потоку. У сценаріях, де порівнювали дві бази даних, MongoDB вийшла на перше місце. На відміну від систем керування реляційними базами

даних (RDBMS), база даних NoSQL добре підходить для використання в кластері, де її гнучка та потужна архітектура може вмістити величезні обсяги даних у різноманітних формах.

MongoDB і MySQL порівнювалися з канонічною системою управління базами даних у [20]. MongoDB перевершив MySQL у пошуку та вставці даних. Грунтуючись на проведених порівняннях, було зрозуміло, що NoSQL MongoDB є найкращим варіантом, ніж MySQL, при обробці великих обсягів даних. Для статей було проведено поглиблене дослідження передумов для ефективного управління величезними обсягами неструктурованих даних. Вони віддали перевагу використанню бази даних NoSQL MongoDB замість використання MySQL RDBMS. Реляційна та нереляційна моделі баз даних були детально розглянуті [21]. Залежно від аналізу база даних NoSQL MongoDB перевершила базу даних MySQL. Вони припустили, що для невеликого набору даних із базовими запитами MySQL є ефективним, тоді як для великих наборів даних із складними запитами більш прийнятним рішенням є MongoDB.

Автори [22] порівняли продуктивність Oracle RDBMS з базою даних NoSQL MongoDB. У результаті свого дослідження вони дійшли висновку, що бази даних NoSQL не є життєздатною альтернативою базам даних SQL. Вони стверджували, що компанії можуть вибрати базу даних, яка відповідає їхнім потребам, враховуючи специфіку своєї компанії.

Що стосується масштабованості даних, високої продуктивності та доступності даних, ми існує багато публікацій, в яких обговорюється перехід від реляційної бази даних до бази даних NoSQL. Завдяки масштабованості, цілісності, розповсюдженню, безпеці [11, 23] і настроюваному дизайну MongoDB добре підходить для обробки величезних обсягів даних. Згідно з цитованими роботами, перехід від реляційної бази даних до бази даних NoSQL MongoDB – це спосіб вирішення проблеми великих даних. Для зберігання інформації MongoDB використовує формат двійкової нотації об’єктів JSON (BSON). Оскільки для неї не потрібні з’єднання, як для реляційних баз даних, MongoDB здатна ефективно зберігати та отримувати велику кількість документів

в одній колекції. MongoDB досить універсальний, щоб обробляти та обробляти дані в широкому спектрі форматів, включаючи структуровані, напівструктуровані та неструктуровані дані. Реляційні бази даних були створені з явною метою обробки впорядкованих даних, і тому вони можуть обробляти певну кількість даних, які є вертикальними за своєю природою.

Щоб було зрозуміло, бази даних NoSQL не призначені для заміни реляційних баз даних, але їх підвищена продуктивність і корисність означають, що вони мають місце поряд з реляційними базами даних у деяких контекстах. Згідно з дослідженням [24], система реляційної бази даних погано масштабується з великими обсягами даних. Автоматичне відображення від MySQL RDBMS до бази даних MongoDB NoSQL з використанням метаданих, збережених у MySQL RDBMS, було запропоновано в роботі [25]. Бази даних NoSQL не обіцяють запропонувати ту саму атомарність, послідовність, ізоляцію та довговічність, які є ознаками систем управління реляційними базами даних.

Функція BASE (Basically Available, Soft state, Eventual consistency) [26] спостерігається в базах даних NoSQL. NoSQL [24] – це загальний термін для набору технологій, які мають спільні цілі доступності даних, ефективного масштабування даних, ефективного управління сховищами та покращеної продуктивності. Дослідження в [12] розглядає мотиви для переходу від систем управління реляційними базами даних до нереляційних баз даних, орієнтованих на документи.

Дослідження теми автоматичного перетворення схеми з SQL на NoSQL можна знайти в [27].

У статтях [28, 29] порівнювалася ефективність кількох баз даних NoSQL (включаючи Couchbase, MongoDB, RethinkDB і in-memory) при використанні з даними реального світу. Плануючи запропоновані тестуваня, автори враховували кілька факторів, наприклад час відгуку різних баз даних. У дослідженні було проведено порівняльний аналіз продуктивності Oracle NoSQL і MongoDB. Бази даних вивчалися з різних точок зору, включаючи їх модель зберігання, масштабованість, паралельність і реплікацію. Автори дійшли

висновку, що в цілому MongoDB був більш популярним, ніж пропозиція Oracle NoSQL. Згідно з рейтингом DBEngine.com, MongoDB є п’ятою найкращою базою даних NoSQL, тоді як Oracle NoSQL займає 78 місце.

У статті [30] стверджується, що порівняно з MapReduce MongoDB, агрегація RDBMS Oracle працює краще. З іншого боку, щодо швидкості відповідей на запити MongoDB перевершив Oracle RDBMS. Відповідно до [22], агрегація Oracle була кращою за агрегацію MongoDB для робочих навантажень SUM, COUNT і AVG. У цій роботі використовувався набір даних із приблизно 500000 записами, чого було недостатньо для аналізу великих даних. Схема процесу оператора SQL Select в Oracle 11g RDBMS зображена на рисунку 1.5.

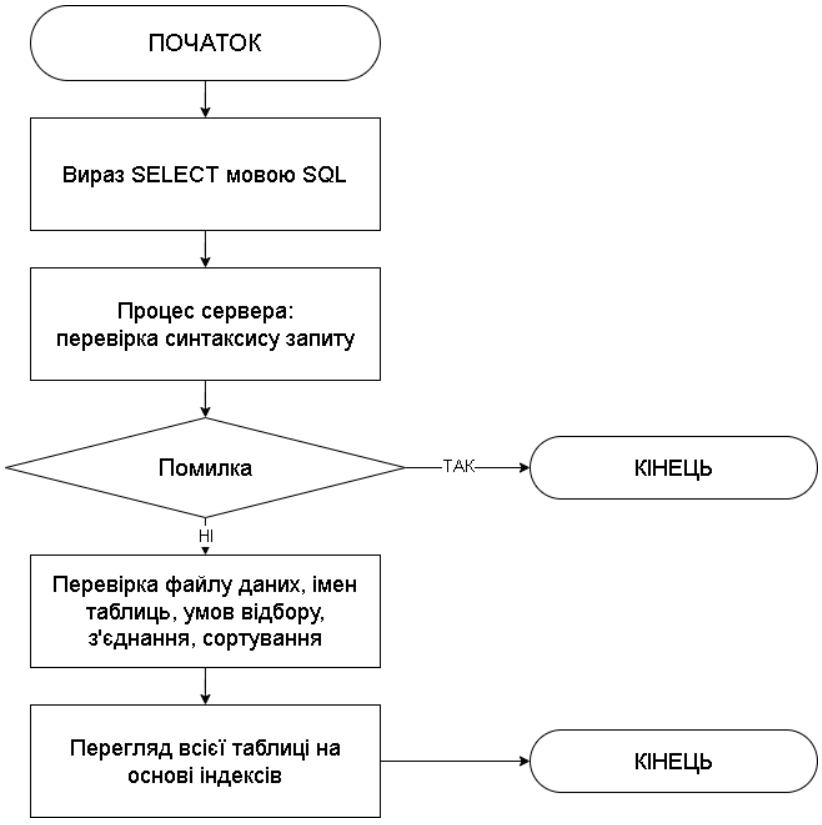

Рисунок 1.5 – Блок-схема процесу оператора SQL Select

Отримання даних MongoDB є простим і легким завдяки структурі піддокументів і не вимагає обмежень перевірки чи будь-яких пунктів, на відміну від оператора SQL Select Oracle 11g RDBMS. Потік процесу оператора SQL Insert показано на рисунку 1.6.

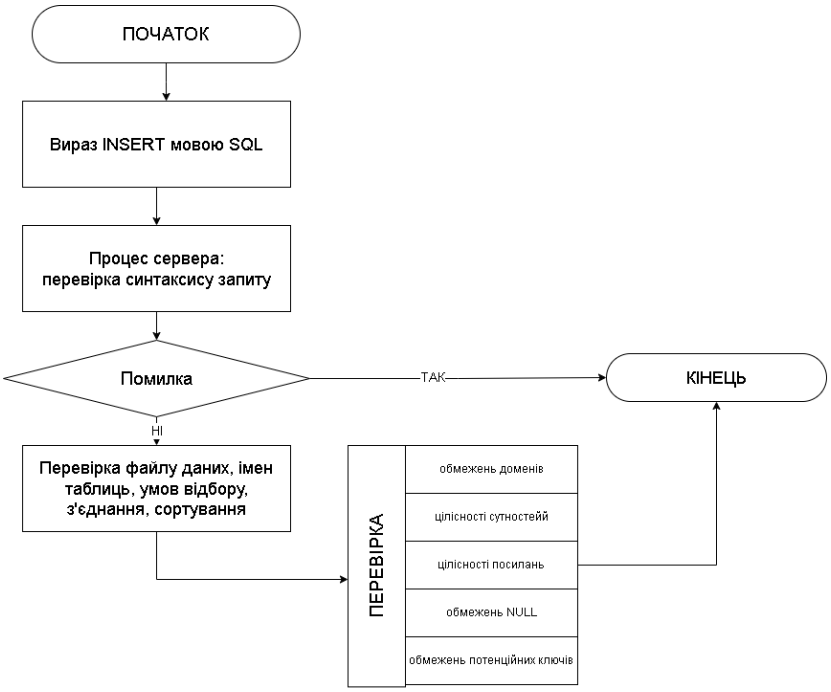

Рисунок 1.6 – Блок-схема процесу оператора SQL Insert

Порівняно з Oracle RDBMS, вставка MongoDB є швидшою, оскільки їй не потрібно перевіряти кроки, показані на рисунку 1.6 оператора SQL Insert.

Навпаки, MapReduce – це модуль програмування, який добре працює в сценаріях розповсюдження та є більш ідеальним для аналізу великих даних [31] порівняно з базовим агрегуванням [30] у кластерних (багатосерверних) контекстах. Перед початком етапу карти операції MapReduce можуть виконувати будь-яке довільне сортування та обмежують документи в одній колекції, яку вони використовують як вхідні дані. Для роботи алгоритму MapReduce потрібні дві фази MapReduce, «Map» і «Reduce». MongoDB використовує фазу карти для кожного документа, який подається в MapReduce. Функція map повертає пару критичних значень. MongoDB використовує процес під назвою «зменшення», щоб зібрати та стиснути агреговані дані, а потім MongoDB зберігає результати в колекції. Наприклад, у наборі даних про злочини в Чикаго функція карти обчислює всі злочини проти кожного дня, а потім функція зменшення бере день як ключ і витягує відповідні значення (Ключ: значення).

Фреймворк MapReduce–Merge був розроблений шляхом включення операції злиття в архітектуру MapReduce. Продуктивність MapReduce була

збільшена завдяки операції злиття та змогла обробляти дані в кластері. Таким чином, агрегація Oracle RDBMS перевершила MongoDB в агрегації MapReduce.

За останнє десятиліття Oracle RDBMS широко використовувалася компаніями будь-якого розміру. Через архітектуру абстрактної системи з одним зображенням СКБД SQL погано справлялися з обробкою неструктурованих даних.

Дослідження [32] вивчало кілька особливостей і класифікацій баз даних NoSQL , що працюють поверх Hadoop. Воно показало, що бази даних NoSQL можна розбити на підкатегорії відповідно до таких факторів, як масштабованість і тип даних (підключений чи документ). Таблиця 1.1 порівнює Oracle RDBMS з MongoDB і перераховує їхні основні функції.

Таблиця 1.1 – Характеристики MongoDB і Oracle RDBMS

SQL	**MongoDB**
Жорстка схема	Гнучка схема
Таблиця	Колекція
рядок	документ
Колонка	Поле
Щоб отримати повну інформацію про одну особу, студента або клієнта, потрібні кілька об'єднань, що спричиняє повільний доступ до даних.	Доступ до даних дуже швидкий, оскільки всі дані зберігаються в одному документі
Вертикальна масштабованість	Горизонтальна масштабованість
Складний шардинг даних	Шардинг даних стає простішим
Розглядає ефективність зберігання даних	Враховує швидкість, продуктивність, час розробника
Добре працює в агрегації	Краще працює під час пошуку
Читання/запис не виконується дуже швидко в аналітиці великих даних	Виконує читання/запис дуже швидко завдяки функції відображення пам'яті в аналітиці великих даних.
Підтримує реляційну алгебру	Підтримує реляційну алгебру

Згідно з цитованими роботами [33, 34, 35], спостерігається помітне збільшення кількості геопросторових і геолокованих даних. Було розроблено багато програм для обробки та використання геопросторових даних, і це стосується багатьох різних галузей (управління надзвичайними ситуаціями, археологія, Інтернет речей та розумні міста). Для того, щоб ефективно обробляти

величезні обсяги географічних даних, потрібна потужна система управління базами даних.

При роботі з величезними обсягами даних найкращим варіантом для веб-додатків є бази даних NoSQL, а не бази даних SQL [36].

У багатьох звітах, наприклад [37, 38] було запропоновано, що бази даних NoSQL здатні ефективно обробляти величезні обсяги неструктурованих даних стосовно геолокації. У роботі повідомляється, що дослідники вперше дізналися про просторові дані в основоположних документах Географічних інформаційних систем. Було досліджено дві проблеми з ефективною обробкою даних геопросторових запитів. По-перше, звичайні методи оптимізації були недостатніми для вирішення географічних запитів. Взявши до уваги велику аналітику, стало зрозуміло, що всі методи та підходи до географічних запитів, випробувані досі, були недостатніми для величезних обсягів даних. На відміну від традиційних текстових даних, які містять лише обмежений набір характеристик, геопросторові дані пропонують широкий спектр додаткових функцій. Широка обробка геопросторових даних, згідно з кількома дослідженнями, наприклад [39], вимагає добре налагоджених методів обробки величезної кількості доступних геопросторових даних. У статті показано, що добре відомі системи RDBMS стикаються з численними труднощами при спробі аналізу географічних даних. Використовуючи порівняльний аналіз продуктивності, автори [40] розглянули, як бази даних документів NoSQL MongoDB порівнюються з системою керування реляційними базами даних PostGIS. Як показали їхні тести, MongoDB працює краще, ніж PostGIS при обробці геопросторових даних.

Модель просторового зберігання даних Oracle [41] мала два основні компоненти: розташування та форму. Моделі зберігання, такі як Index Engine і Geometry Engine, використовувані для аналізу запитів, використовували тип даних SDO_GEOMETRY. Для служб визначення місцезнаходження використовувався геокодер для перетворення адреси в інформацію SDO_GEOMETRY. Для візуалізації використовували Oracle Maps і Map Viewer.

SQL Server 2016 і його хмарна версія SQL Azure включають різноманітні геофункції для аналітики геопросторових даних, як і просторову базу даних SQL Oracle, а бази даних NoSQL, наприклад Azure DocumentDB і MongoDB. Парадигма «База даних як послуга» (DBaaS), яка використовується базою даних Azure SQL, також має такі ж функції сервера Microsoft SQL і надає хмарні служби. PostgreSQL – це безкоштовна відкрита система реляційної бази даних. Для порівняння, PostGIS є розширенням PostgreSQL, яке служить просторовою базою даних для роботи з геопросторовою інформацією.

База даних NoSQL Azure DocumentDB створена компанією Microsoft і забезпечує ті самі географічні операції та функції, що й MongoDB. MongoDB підтримує стандартний формат даних GeoJSON. Автори [42] провели аналіз ефективності географічних даних. Згідно з їхніми дослідженнями, Azure DocumentDB була швидшою, ніж база даних Azure SQL, але менш масштабованою, ніж база даних Azure SQL. Основні геопросторові властивості популярних баз даних SQL і NoSQL наведено в таблиці 1.2 .

Таблиця 1.2 – Геопросторові характеристики баз даних SQL і NoSQL

База даних	**Oracle**	**PostGIS**	**Azure SQL**	**MongoDB**	**DocumentDB**
Геометричні об'єкти підтримуються	Point, LineString , полігон, MultiPoint, MultiLinePoint , MultiPolygon , GeometryCollection	Point, LineString , полігон, MultiPoint, MultiLinePoint , MultiPolygon , GeometryCollection	Point, LineString , полігон, MultiPoint, MultiLinePoint , MultiPolygon , GeometryCollection	Point, LineString , полігон, MultiPoint, MultiLinePoint , MultiPolygon , GeometryCollection	Point, LineString , полігон, MultiPoint, MultiLinePoint , MultiPolygon , GeometryCollection
Підтримуються функції геометрії	Для екземплярів геометрії Oracle підтримує Open Geospatial Консорціум (OGC)	Для екземплярів геометрії PostGIS підтримує Open Geospatial Консорціум (OGC)	Для прикладів геометрії, Azure SQL	Включення, Перетин, Відстань/Близькість	Включення, Відстань/Близькість
Підтримуються просторові індекси	B-дерева, збірки паралельних індексів для просторових індексів R-дерев	GiST , індекс R-Tree, індекс B-Tree	B-дерева, 2D індекс площини	2D індекс, 2D індекс сфери	2D плоский індекс, квадродерево
Сумісність з GeoServer	так	так	так	так	так
DBaaS	так	немає	Так (платформа хмарних обчислень)	так	так
Горизонтальна масштабованість	немає	немає	немає	так	так

1.3 Моделювання даних NoSQL MongoDB

На рисунку 1.6 зображено шард-вузли, сервери конфігурації та сервери маршрутизації (або mongos), які складають архітектуру MongoDB.

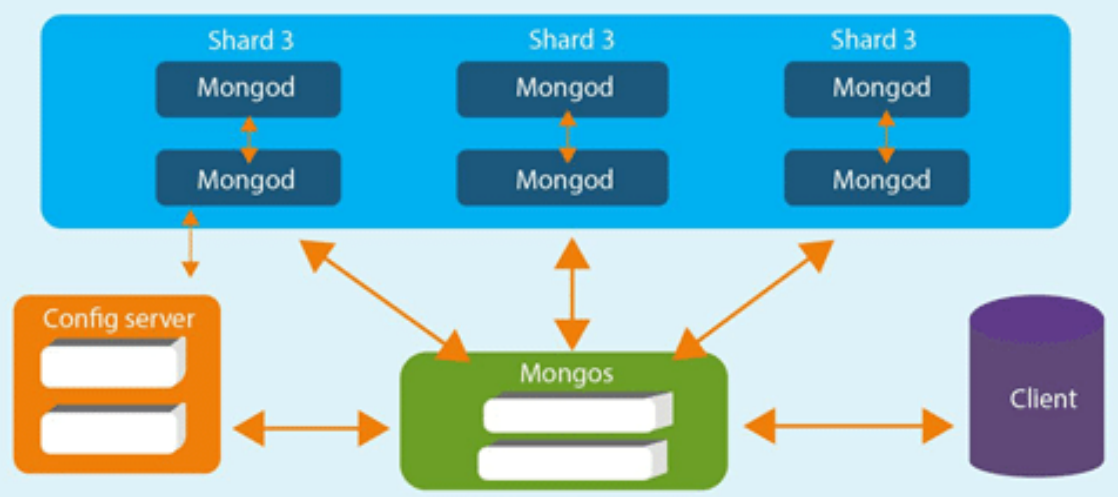

Рисунок 1.7 – Архітектура MongoDB

Дані зберігаються в сегменті, і кластер MongoDB неможливо побудувати без одного або кількох шардів. Коли вузол виходить з ладу, інформація для цього фрагмента зберігається його репліками. Транзакції даних (читання/запис) визначають, який шард буде використовуватися. Один або кілька серверів використовуються реплікованим вузлом, а вторинний вузол моделюється як вихідний сервер. Якщо основний сервер вийде з ладу, один із резервних серверів перейде на його роль. Сервер є центром, навколо якого обертаються всі операції (читання/запис). Зрештою, кластер буде синхронізований з усіма розподіленими транзакціями читання.

Набір серверів конфігурації в кластері відповідає за зберігання метаданих. Шарди даних ідентифікуються цими серверами, які також передають, яка частина даних до якого шарду належить. Служба обслуговування клієнтів вимагає, щоб або сервери маршрутизації, або MongoDB виконали дію. Перш ніж отримати підтвердження від клієнта, MongoDB призначає кожне завдання користувача відповідному сегменту на основі типу завдання, а потім об’єднує

отримані дані. Оскільки Mongos [43] не мають статусу, їх можна використовувати в розподілених умовах.

Файли з відображенням пам’яті використовуються MongoDB, щоб максимально використовувати доступну пам’ять, що, у свою чергу, підвищує продуктивність. Індексування в базі даних MongoDB використовує B-дерева. У кластері MongoDB користувач може претендувати на право власності на певну колекцію розділів за допомогою шардового ключа.

2 АНАЛІЗ ПОСЛУГИ DBaaS ДЛЯ НЕРЕЛЯЦІЙНИХ БАЗ ДАНИХ

2.1 Ефективність переносу даних між різними нереляційними СКБД

Для цієї роботи було проведено поглиблене дослідження літератури про архітектуру DBaaS. Аналіз і отримані дані показують, що хмарний підхід DBaaS, розроблений для реляційних баз даних, не є оптимальним для баз даних NoSQL. Експерти можуть скоротити час і зусилля, а безпеку можна покращити, якщо вчені та дослідники зможуть знайти уніфіковане DBaaS рішення для баз даних SQL і NoSQL.

Завдяки стандартним API також немає потреби перепроектовувати програми для використання з різними CSP. Портативність даних і сумісність між різними хмарними провайдерами є основними перешкодами, які необхідно подолати. Інтероперабельність визначається по-різному кожною з трьох парадигм: IaaS, PaaS і SaaS. Відкриті стандарти [44] допомагають пом’якшити проблему сумісності; однак наше рішення зосереджено саме на рівні IaaS. Під час передачі інформації між різними хмарними провайдерами зазвичай необхідні уніфіковані API. Немає єдиної моделі зберігання даних, яка використовується всіма хмарними службами. Коли розробник переходить від одного CSP (Cloud Service Provider) до іншого, дані передаються відповідно до архітектури високого рівня, зображеної на рисунку 2.1.

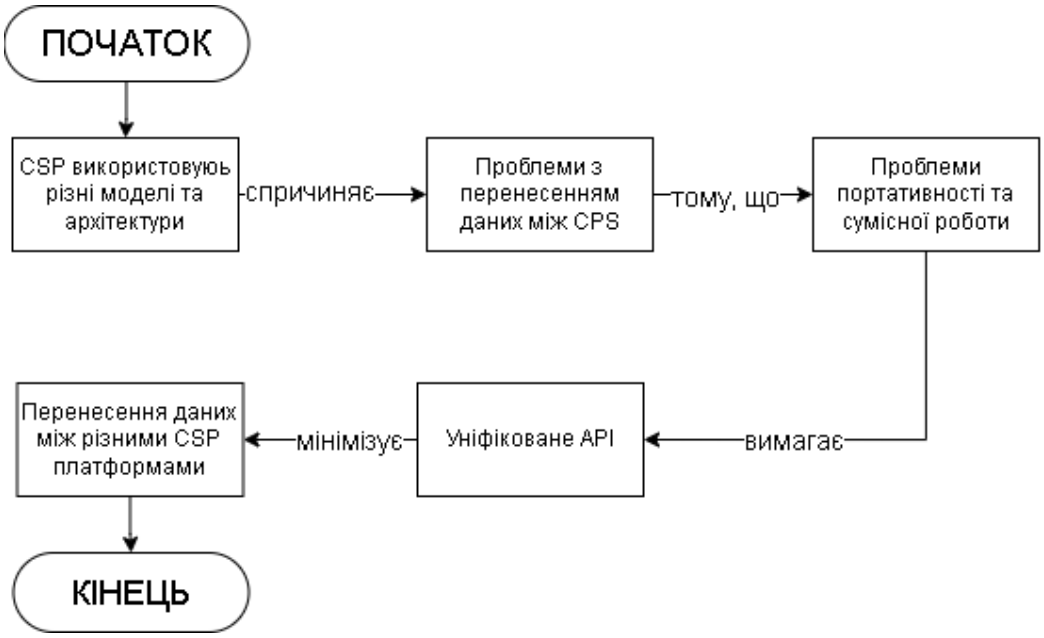

Рисунок 2.1 – Переміщення даних між CSP

Бази даних, які є одночасно узгодженими та пов’язаними, а також ефективне управління ними, є ключовими та критичними проблемами в епоху сучасних інформаційних технологій. Основними особливостями системи баз даних є гарантія постійного сховища даних щодо незалежності даних від базового фізичного носія, а також можливості обробки запитів, що забезпечуються декларативними запитами (DBS).

У секторі баз даних спостерігається велика різноманітність підходів до різних областей. Кілька DMS, включаючи ACID, OOM, XML і сховища даних, засновані на реляційній моделі даних. Однак атрибут BASE [26] підтримується в базах даних NoSQL, які в основному призначені для обробки великих обсягів даних.

Провайдери хмарних послуг пропонують клієнтам нові хмарні послуги та можливості за низькою ціною та з високою ефективністю. Проте кілька постачальників хмарних послуг пропонують однакові функції, використовуючи різні реалізації та інтерфейси користувача, що неминуче спричиняє проблеми з сумісністю, несумісністю та портативністю [45]. Це складні проблеми для провайдерів хмарних послуг під час впровадження та полегшення хмарних технологій. Інтероперабельність IaaS, PaaS і SaaS – це різні терміни з різним

значенням і застосуванням у сфері хмарних сервісів. Користувачі хмарних послуг можуть виконувати переміщення з одного CSP до іншого з наступних причин:

- високі показники простою або відмови;
- розірвання контракту;
- зміни корпоративного плану;
- кращі альтернативи з низькою вартістю;
- юридичні труднощі.

Клієнти, які використовують кілька хмарних провайдерів, не можуть легко перенести свої дані між ними. Моделі хмарних сервісів прагнуть контролювати можливості клієнта, оскільки їхні ключові архітектури несумісні. Цю проблему вирішують за допомогою блокування від постачальника, що створює серйозну проблему безпеки для хмарних моделей.

Портативність даних покращиться в близькому майбутньому, оскільки все більше постачальників хмарних послуг (CSP) приймають відкритий стандарт для вирішення проблем сумісності. У результаті розробнику може бути складно забезпечити узгодженість даних і програм у різних хмарних службах.

Відкритий формат віртуалізації (Open Virtualization Format – OVF), інтерфейс керування хмарною інфраструктурою (Cloud Infrastructure Management Interface – CIMI), відкритий інтерфейс хмарних обчислень (Open Cloud Computing Interface – OCCI) та інтерфейс керування хмарними даними (Cloud Data Management Interface – CDMI) розглядалися як потенційні рішення для керування сумісністю між CSP. Ці стандарти підтримують лише рівень IaaS. Подібним чином, багато ініціатив, таких як MOSAIC [46], MODACLOUDS [47], були створені для вирішення проблеми семантичної сумісності на рівні PaaS. Наявність кількох API для кожного PaaS означає, що ініціативи не можуть самостійно вирішити проблему сумісності.

Як і очікувалося, стандарт CIMI сумісний з API IaaS і допомагає зменшити ступінь сумісності, необхідний користувачам хмари та їхнім провайдерам інфраструктурних послуг. Використовуючи OCCI, сумісність між CSP може

бути зменшена, водночас одержуючи адекватні результати. Оскільки ці стандарти не включають функції сумісності в їх базові архітектури, вони не широко використовуються різними CSP. Крім того, існує проблема з доступністю стандартизованих інтерфейсів та API.

Шаблони проектування, хмарні проміжні інфраструктури, хмарна платформа надання послуг (Service Delivery Cloud Platform – SDCP) і інструменти міграції були розглянуті в інших роботах [48, 49], щоб полегшити переміщення даних між різними хмарами. Незважаючи на економію часу користувачів, ці методи не вирішують проблему переносимості між різними CSP, яка виникає через необхідність вивчення та впровадження кількох API. Amazon Web Service (AWS), Microsoft Azure та Google App Engine (GAE) є прикладами постачальників хмарних послуг, які допомагають споживачам створювати та запускати хмарні програми. Крім того, вони надають хмарну платформу Database as a Service (DBaaS) для підтримки розробників баз даних. Перенесення даних у DBaaS, яке може відбуватися між базами даних SQL і NoSQL, а також усередині кожного типу бази даних NoSQL, потенційно може бути проблематичним через проблему сумісності. Багато типів баз даних NoSQL дотримуються несумісних форматів зберігання та парадигм даних. Існує потреба в стандартизованих API, які можуть регулювати рух даних між різними платформами хмарного зберігання.

Щоб розмістити свої дані та додатки, розробники програмного забезпечення можуть скористатися перевагами DBaaS [50], високомасштабованої та доступної серверної хмарної служби. Як наслідок, DBaaS зараз є найпривабливішим варіантом для хмарних клієнтів. База даних як послуга (DBaaS) – це хмарна служба, яку пропонують хмарні провайдери (CSP), яка полегшує перехід від традиційної архітектури бази даних до архітектури хмарної бази даних. SaaS полегшує віддалений доступ до комп’ютерних програм та їхніх функцій і можливостей. PNUTS, HBase, SimpleDB і Google BigTable – це деякі інші хмарні служби баз даних. Потенційно структура DBaaS може добре працювати з традиційними базами даних. Однак продуктивність DBaaS

знижується в таких сферах, як узгодженість даних, конфіденційність, цілісність, доступність і відсутність безпеки через різні підходи, яких дотримуються численні бази даних. Якби конфіденційність і безпека даних були гарантовані, модель аутсорсингу, відома як DBaaS, може бути фінансово успішною. Хмарні обчислення дозволяють інтегрувати численні програми в структуру самого сервісу. Масштабованість і адаптивність хмарних обчислень коштує нижче, ніж будь-коли раніше.

Щоб забезпечити більшу сумісність даних, портативність і безпеку під час перенесення даних, у статті [51] досліджено дві уніфіковані структури API, CDPort і SecloudDB , для баз даних SQL і NoSQL. Перш ніж передати його третій стороні, запропонований API забезпечив конфіденційність критичної інформації користувачів (CSP). У рамках запланованого MCTool запити перетворюються у відповідні моделі, а потім передаються до відповідної бази даних, враховуючи моделі, які вона може обробляти. Ключі шифрування/дешифрування метаданих гарантують, що лише авторизовані користувачі та адміністратор бази даних мають доступ до даних, які зберігаються в різних хмарах, і можуть вносити зміни в них. Шифрування та дешифрування підтримуються запропонованою платформою.

2.2 Практична реалізація принципу CAP для нереляційних баз даних

Дебати SQL проти NoSQL не стосуються реляційних і нереляційних баз даних, незважаючи на назви. Моделі транзакцій обох баз даних аналізуються та порівнюються. Коли програма виконується, база даних виконує серію операцій, відомих як «транзакції». Усі транзакції в базі даних SQL залежать від властивостей ACID. У цьому випадку ACID відноситься до принципів атомізму, послідовності, ізоляції та довговічності. Однак розробники баз даних NoSQL дійшли висновку, що властивість ACID є марною перешкодою для захисту величезних даних. Отже, на початку 2000-х років професор Ерік Брюер запропонував нову теорію. Принципами CAP є послідовність, доступність і

толерантність до розділів. Теорема стверджує, що всі ці якості можуть бути отримані одночасно дизайнерами, які працюють у розподіленому середовищі. В якості компромісу для гарантованої узгодженості та доступності розробники можуть реалізувати толерантну до розділів базу даних за допомогою моделі CA. Якщо розробник бази даних більше цінує доступність і толерантність до розділів, ніж узгодженість, то не можна сказати, що база даних заснована на AP. Щоб досягти узгодженості та толерантності до розділів без шкоди для доступності, розгортається база даних на основі CP. Таблиця 2.1 описує основні функції, які підтримуються обома різновидами баз даних.

Таблиця 2.1 – Характеристики СКБД

СКБД	**Характеристики**						
	Схема	**Масштабованість**	**Відпо-відність**	**Архітек-тура**	**Мова запитів**	**Найкраще підходить для продуктивності**	
Реляційні СКБД	Фіксо-вана	Вертикальна	ACID	Централі-зована	SQL	Повільно	Банківська сфера
NoSQL	Дина-мічна	Горизонтальна	BASE	Розподі-лена	OO API, SQL Like	Швидко	Сховища даних

Стандартизація того, як класифікувати СКБД, призвела до кількох підкатегорій. Основою будь-якої системи керування базами даних є модель даних, на основі якої вона побудована. Багато сучасних комерційних СКБД значною мірою покладаються на реляційну модель даних. Модель об’єктних даних мало впроваджена, навіть якщо вона використовується в невеликій кількості комерційних систем. Багато застарілих програм продовжують працювати в системах баз даних на основі ієрархічних і мережевих моделей даних. Ми можемо класифікувати бізнес-модель NoSQL як одну з наступних.

1. Проблеми з надійністю та доступністю (CA): ця структура бази даних надає пріоритет узгодженості та доступності даних за допомогою підходу реплікації. Частини бази даних не піклуються про допуски фрагментів. Якщо

вузли розділені, дані не будуть синхронізовані. Vertica, Greenplum і системи управління реляційними базами даних є прикладами цього типу баз даних.

2. Є проблеми з узгодженістю та толерантністю до розділів (CP), які потрібно виправити. Основна мета такої системи керування базою даних – забезпечити цілісність даних, які вона зберігає. Однак висока доступність більше не підтримується. Дані зберігаються в різних вузлах, і коли вузол виходить з ладу, це призводить до того, що дані, які забезпечують узгодженість між ними, стають недоступними. Він підтримує толерантність до розділів, блокуючи повторну синхронізацію даних. Hypertable, BigTable і HBase – це лише кілька прикладів систем баз даних, які підтримують CP.

3. У цьому типі бази даних забезпечення доступності даних і доступу до розділів (AP) є головним пріоритетом. Якщо зв'язок між вузлами зривається, це не впливає на статус окремого вузла. Після вирішення розділу дані повторно синхронізуються, але узгодженість не забезпечується. Цих принципів дотримуються такі бази даних, як Riak, CouchDB.

4. Коли один елемент бази даних виходить з ладу, але інші продовжують працювати, цей розділ бази даних вважається «загалом доступним». У разі збою вузла операція продовжиться шляхом реплікації даних серед інших вузлів.

5. У програмному стані дані можуть змінюватися з часом залежно від таких факторів, як рівень участі користувача. Корисність такої інформації також може погіршитися після закінчення певного періоду. Тому необхідно або оновити, або отримати дані, щоб інформація була корисною в системі.

6. Відповідно до кінцевої узгодженості, після кожної зміни даних дані не стають узгодженими в усій системі, але вони стануть узгодженими з часом. Зазначається, що дані залишаться точними в осяжному майбутньому.

Системи баз даних NoSQL отримали ряд ознак, які можна пояснити тим, що структура реалізації не включає звичайний SQL. Кожна з кількох різних типів баз даних NoSQL використовує унікальний набір процедур запитів. Розробники програмного забезпечення часто несуть відповідальність за належне

проектування виконання запитів, а не залежать від декларативної мови запитів, яка поєднує запитання та забезпечує швидкі плани виконання запитів. Це тому, що декларативна мова запитів може об’єднувати запитання.

Користувачі системи відповідають за виконання додаткових завдань, таких як перевірка та інтерпретація зібраної інформації. Крім того, відповідальність за забезпечення узгодженості даних, реплікацію даних і доступність під час одночасних змін у спільних і реплікованих базах даних все більше і більше переходить до рук розробників. Поєднання масивних систем даних із базами даних NoSQL може мати декілька різних впливів на дизайн та архітектуру програмного забезпечення.

Брюер сформулював набір вимог до розподілених баз даних, використовуючи свій метод CAP [52]. Ці вимоги служать базовими. У випадку, коли є мережевий розділ (P: у кластері, коли інформація втрачається випадковим чином між вузлами), узгодженість (C: представлення ідентичних даних усім користувачам) має мати пріоритет над доступністю (доступністю для користувачів) (A: підтвердження має бути отримано кожним клієнтом як успіх або невдача). Якщо немає розділу (P), структура буде віддавати перевагу затримці (L) над узгодженістю. Таким чином отримуємо принцип PARCELS: незважаючи на це, коли є розділ (P), структура буде наголошувати на доступності (A), а не на узгодженості (C).

Сьогодні недостатньо покладатися виключно на структуровані дані, оскільки неструктурована природа даних вимагає швидкого аналізу даних і методів отримання інформації. Можливості бази даних SQL обмежені в їх здатності ефективно обробляти різні форми великих даних. Можливості баз даних NoSQL дозволяють ефективно обробляти масивні набори даних. Бази даних NoSQL перевершують у сферах розширення обсяги зберігання, гнучкості схеми, масштабованості та доступу в реальному часі. Бази даних NoSQL дотримуються атрибута BASE. Замість того, щоб віддавати пріоритет узгодженості та безпеці даних, бази даних NoSQL наголошують на покращенні ефективності читання та запису даних. Додаток відповідає за забезпечення

узгодженості даних. З цієї причини це чудовий варіант для обробки великих наборів даних. Крім того, до баз даних NoSQL не застосовуються обмеження на рівні даних, стовпців або таблиць, які використовуються в структурованих базах даних.

У цьому дослідженні проаналізовано приблизно ряд досліджень, які порівнювали корисність, продуктивність і надійність баз даних SQL і баз даних NoSQL. Це дослідження врахувало величезні обсяги даних. Згідно з нашими дослідженнями, бази даних NoSQL пропонують більший потенціал для розширення [43, 53]. Бази даних SQL краще підходять для транзакційних систем і використовують більше ресурсів для забезпечення цілісності та узгодженості даних, тоді як бази даних NoSQL краще вирішують задачі обробки великих і різноманітних наборів даних і використовують менше ресурсів для забезпечення цілісності та узгодженості даних.

З іншого боку, бази даних NoSQL відрізняються в тому сенсі, що вони надають більший пріоритет доступності даних. Результати нашого дослідження показують, що реляційні бази даних не можна ефективно замінити базами даних NoSQL. Оскільки обидві бази даних мають свої переваги та недоліки, база даних, яку організація вибере для використання, залежатиме від вимог, які є унікальними для цього бізнесу. Щоб навести лише одну ілюстрацію, бази даних NoSQL, які дозволяють використовувати модуль програмування MapReduce, краще підходять для паралельних обчислень, коли вони реалізовані в середовищі кластера.

Бази даних NoSQL побудовані на схемі, яка є більш динамічною, на відміну від реляційних баз даних, які сильно залежать від попередньо встановленої структури даних, яку називають «схемою» (таблична форма). Наприклад, щоб відстежувати дані студентів, слід, для прикладу, використовувати поля StdRegNo, StdName і StdAddress. Під час роботи з реляційними базами даних перше, що має відбутися, це побудова схеми, яка задовольняє всім необхідним вимогам домену та цілісності.

Після цього можна буде зберегти відповідні дані студента та дотримуватися необхідних обмежень. Розглядаючи розширення обсягу поточної бази даних шляхом включення двох нових стовпців, слід приділити цьому певну увагу. Ті, хто буде відповідати за внесення змін до існуючої схеми, також відповідатимуть за міграцію даних із попередньої схеми до нової.

Маючи справу з великими наборами даних, процедура може стати непрактично повільною та довгою. Той факт, що бази даних NoSQL не оновлюються автоматично щоразу, коли в схему вносяться нові зміни, становить найважливішу проблему для гнучкої розробки програмного забезпечення [12]. Під час роботи з базами даних NoSQL необов'язково мати заздалегідь визначену схему для внесення будь-яких змін.

Нормалізація даних для контролю реляційної схеми (атрибутів), обмежень домену, обмежень перевірки, унікальних обмежень і обмежень Not NULL, серед іншого, сприяє безпечній інтеграції даних у реляційні бази даних на відміну від баз даних NoSQL. Реляційні бази даних забезпечують добре налагоджену систему безпеки та автентифікації для користувачів. Бази даних SQL перевершують бази даних NoSQL як з точки зору продуктивності, так і довговічності. SQL є стандартним інтерфейсом для всіх реляційних баз даних; однак бази даних NoSQL не використовують SQL. Натомість вони використовують інший інтерфейс.

Бази даних SQL і NoSQL використовують різноманітні моделі. Крім того, існує різниця в основній моделі даних, що використовується кожним типом бази даних NoSQL. Враховуючи поширеність кількох CSP, важко досягти переносимості даних. Швидке розширення комерційних баз даних створює ще одну проблему для хмарного зберігання інформації. Натомість AWS DBaaS пропонує дещо обмежені параметри розширення сховища для хмарних баз даних Azure. Категорії продуктів, архітектури, типи баз даних і мови наведено в таблиці 2.2.

Таблиця 2.2 – Категорія продукту бази даних, тип архітектури та мови

Ім'я бази даних	Категорія бази даних	Архітектура бази даних	Тип бази даних	Написано в
MongoDB	NoSQL-сховище документів	Розподілена мультимодель	Відкритий код	C++, Go, JavaScript, Python
MySQL	SQL		Відкритий код	C, C++
Оракул	SQL		ні	Мова асемблера, C, C++
SQL Server	SQL		ні	C, C++
Neo4j	Сімейство NoSQL-Graph		Відкритий код	Java
Диванна база	NoSQL-сховище документів	Розподілена мультимодель	Відкритий код	C++, Erlang, C, Go
CouchDB	NoSQL-сховище документів	Розподілена мультимодель	Відкритий код	Erlang, JavaScript, C, C++
Кассандра	На основі стовпців NoSQL	Розподілена мультимодель	Відкритий код	Java
Переосмислити	NoSQL-сховище документів	Розподілена мультимодель	Відкритий код	C++, Python, Java, JavaScript, Bash
MariaDB	SQL		Відкритий код	C, C++, Perl, Bash
RavenDB	NoSQL-сховище документів		Відкритий код	C#
BigTable	На основі стовпців NoSQL		ні	C++ (ядро), Java, Python, Go, Ruby
DynamoDB	NoSQL-Key/Value Store		ні	Java
SimpleDB	NoSQL-Key/Value Store	Розподілена база даних	ні	Ерланг
PostgreSQL	SQL		Відкритий код	C
PostGIS	SQL		Відкритий код	C
Hbase	На основі стовпців NoSQL	Розподілена мультимодель	Відкритий код	Java
GraphDB	Сімейство NoSQL-Graph	Розподілена база даних		
Sybase	SQL		ні	SQL
Redis	NoSQL - сховище ключів/значень		Відкритий код	ANSI C
Гіпертаблиця	На основі стовпців NoSQL		Відкритий код	C++
JenusGraph	Сімейство NoSQL-Graph		Відкритий код	Java
VoltDB	СКБД в пам'яті		Відкритий код	Java, C++
ВолдеМорт	NoSQL-Key/Value Store	Розподілене сховище даних	Відкритий код	Java
Інфогрід	Сімейство NoSQL-Graph		Відкритий код	Java
H2	SQL		Відкритий код	Java
ArangoDB	NoSQLDB		Відкритий код	C++, JavaScript
OrientDB	NoSQLDB		Відкритий код	Java
Azure SQL	SQL		Відкритий код	C, C++
Azure DocumentDB	NoSQL-сховище документів		Відкритий код	SQL (Core) API

3 ОСОБЛИВОСТІ РЕАЛІЗАЦІЇ НЕРЕЛЯЦІЙНИХ БАЗ ДАНИХ

3.1 Реалізація розподілу даних

Реалізація складних розподілених систем потрібна для досягнення значних рівнів як доступності, так і масштабованості. Крім того, сегментування та розділення відбуваються на базових рівнях архітектури програмного забезпечення, які називаються рівнем додатків, рівнем кешування та рівнем внутрішнього сховища відповідно. Однак досягти високої масштабованості може бути складно через атомарну абстракцію, представлену типовою структурою, яка використовує SQL як золотий стандарт непроцедурної мови.

Крім того, програмне забезпечення має бути інтелектуальним, щоб вирішувати проблеми, які виникають із копіями даних, і невідповідності, спричинені колізіями між змінами реплік, які відбуваються одночасно. Що стосується критеріїв якості, таких як масштабованість, послідовність, продуктивність і довговічність, кожен різновид бази даних NoSQL має власний унікальний набір недоліків і обмежень. Разом з тим архітектор принципово зобов’язаний провести дослідження характеристик номінальної бази даних, щоб виконати вимоги, висунуті постачальником під час вибору відповідної бази даних. Перерахуємо прогалини кожного різновиду баз даних NoSQL.

1. Якщо програмам просто потрібно зберігати та отримувати елементи даних, які прозорі для СКБД і можуть бути ідентифіковані за допомогою ключа, сховище ключ-значення може бути найкращим варіантом для. Навпаки, програмне забезпечення аварійно завершує роботу, якщо воно намагається виконати запит до бази даних на основі значення атрибута, відмінного від ключа. Крім того, неможливо оновити або отримати лише одне поле зі сховища ключ-значення документа.

2. Якщо додаткам потрібен детальний контроль над тим, які записи отримувати, які поля в записі змінювати та які поля отримувати на основі критеріїв, відмінних від первинного ключа, бази даних документів є чудовим

варіантом. Сховища даних документів забезпечують більшу гнучкість запитів, ніж сховища ключ-значення.

3. Коли програмам потрібно зберігати дані з сотнями чи тисячами полів, але в більшості запитів потрібно отримати доступ лише до підмножини цих полів, сховища даних із сімейства стовпців є ефективним рішенням. Такі сховища даних добре підходять для масивних наборів даних.

4. Графові бази даних ідеально підходять для випадків використання, які включають зберігання та аналіз даних про сутності зі складними зв'язками одна з одною. У графовій базі даних важливість сутностей та їхніх зв'язків розглядається однаково.

Термін «великі дані» часто використовується для позначення даних, отриманих із великомасштабних сенсорних мереж або створеної користувачами інформації з соціальних мереж. Однак, оскільки багато людей не мають необхідних навичок, вони не знають, як працювати з даними, щоб досягти бажаного результату для свого бізнесу. Це пояснюється тим, що значна кількість технологій і методів, що стосуються великих даних, є відносно недавніми розробками.

З іншого боку, реляційні сховища даних не в змозі обробити таку велику колекцію даних, що вимагає створення нових можливостей системи зберігання. Дані, що зберігаються в базі даних NoSQL, не потребують централізованої схеми. Завдяки більшій масштабованості, доступності даних, відмовостійкості та швидкій обробці величезних обсягів неструктурованих даних бази даних NoSQL підходять для обробки великих даних. Є багато питань, які ще не вирішені через відмінності між реляційними базами даних і базами даних NoSQL.

Можна зробити висновок, що NoSQL не є прийнятною альтернативою реляційним базам даних. На додаток до цього, NoSQL є чудовим варіантом для неоднорідних великих даних. Кожна з цих областей має численні незаповнені дослідницькі пустоти. Простота, масштабованість і продуктивність проектів баз даних NoSQL є областями, які потребують значного простору для вивчення. На відміну від суворої структури даних реляційної бази даних, бази даних NoSQL

використовують плоский файл або модель даних ключ/значення (таблична форма).

Бази даних NoSQL, завдяки своїй горизонтальній масштабованості, добре підходять для роботи з величезною кількістю та різноманітністю даних, які зараз використовуються. Однак це не стосується баз даних SQL, які можуть масштабуватися вертикально. При порівнянні NoSQL з реляційними базами даних масштабованість і продуктивність є двома найважливішими факторами. Існує також дослідницьке питання без відповіді про те, як інтегрувати функції [35] реляційних баз даних із функціями нереляційних баз даних.

Розробка гнучких структур міграції даних з реляційної бази даних до сховища даних NoSQL є ще одним напрямком досліджень, який отримав підвищений інтерес в останні роки. Це надзвичайно важливо, оскільки будь-якій компанії необхідно отримувати корисні відомості зі своїх власних даних, наприклад про використання системних ресурсів і оцінки продуктивності співробітників.

Крім того, NoSQL є найкращим варіантом порівняно з реляційними базами даних у хмарі через розподілену природу хмарної платформи. Для того, щоб зменшити зусилля споживачів хмари під час переміщення даних між багатьма хмарними платформами та здійснювати контроль над сумісністю та портативністю даних, сучасний рівень вимагає уніфікованих інфраструктур API.

3.2 Приклад порівняння продуктивності реляційної і нереляційної СКБД

SQL або мова структурованих запитів є найбільш широко використовуваним виразом у світі для виконання команд у реляційних базах даних на основі таблиць. За допомогою SQL можна створювати бази даних, таблиці, стовпці та індекси, у яких він гарантує та скасовує відмінності користувачів шляхом запиту та зберігання даних.

Таким чином, SQL є надійною мовою, розділеною на набори команд: мова визначення даних (DDL), мова маніпулювання даними (DML), мова керування даними (DCL), мова керування транзакціями (TCL) і мова запитів даних (DQL). SQL є одним із найпоширеніших варіантів, що робить його безпечним і вигідним варіантом для складних служб. Однак його можна обмежити.

Усі дані у БД мають мати однакову структуру. Це може вимагати тривалої початкової підготовки, і одна помилка може призвести до зупинки всієї системи. Сьогодні база даних SQL призначена для непродуктивних даних і надається як упорядковані дані. Її гнучкість дозволяє кожному документу мати свою структуру, дозволяючи різні синтаксиси бази даних для інших баз даних. У міру того, як розміри набору даних зростають, архітектури великих даних стають необхідними для ефективної та своєчасної їх обробки. У свою чергу, це передбачає використання ефективних алгоритмів для оптимізації розміру кластера та вартості, а також для вибору масштабованих функцій і зменшення розмірності.

З іншого боку, бази даних NoSQL є масштабованими. Цей базовий об’єкт підтримує більше трафіку шляхом фрагментації (розділу даних), тобто додавання більше серверів до своєї бази даних. Це робить NoSQL потужнішим і найкращим для проектування великих наборів, дозволяючи постійні зміни.

База даних NoSQL складається з файлів, організованих у форматі ієрархічної бази даних, і використовує оболонку UNIX Shell як інструмент взаємодії. Моделі баз даних NoSQL мають зовсім іншу структуру, кожна з яких використовує несумісну форму запиту. Бази даних NoSQL визначають чотири типи структурованих даних, такі як документи, графи, пари ключ-значення та великі сховища стовпців. Один із найзагальніших типів у світі NoSQL, зі схемами, визначеними як гнучкі, дозволяє записам і документам мати різні типи та кількість полів без потреби у фіксованій схемі.

Такі документи визначаються у форматі JSON через концепції, суміжні з об’єктно-орієнтованою моделлю, де вони еквівалентні документам. Крім того,

можна додавати нові поля, що дозволяє швидко розробляти програми, як у випадку MongoDB і CouchDB.

Забезпечення якості викликає серйозне занепокоєння в індустрії розробки програмного забезпечення, оскільки сьогодні більшість компаній використовують цю програму для керування своїм бізнесом, продуктами та стосунками з клієнтами, що вимагає кращої надійності та якості. Тому заходи забезпечення якості мають вирішальне значення для успіху будь-якої програми. Одним з найпростіших і дешевих є моніторинг. Моніторинг програмного забезпечення допомагає в прийнятті рішень, надаючи кількісні дані, щоб повідомити, які аспекти продукту відповідають або не відповідають встановленим стандартам якості, і дозволяючи оцінити переваги нових інженерних методів і інструментів.

Наприклад, розуміння та вдосконалення виробничих процесів, оцінка рентабельності інвестицій та управління проектом на основі фактів, а не «здогадок».

Для моніторингу програмного забезпечення використовуються різні показники, як-от типи вимірювань, що використовуються в програмній системі, документації чи пов’язаному процесі. За допомогою цих показників можна визначити зусилля чи час, необхідні для виконання завдання, або розмір продукту. Крім того, програмні показники легко обчислюються, розуміються, перевіряються та не залежать від спостерігача, який їх застосовує. Вони також є хорошим джерелом для статистичних досліджень життєвого циклу програмного забезпечення.

Метрики можна і потрібно застосовувати на етапі розробки програмного забезпечення, забезпечуючи їх позитивний вплив на кінцевий продукт. Однак, на думку деяких експертів, вимірювання необхідно визначати відповідно до конкретних цілей для вимірювання артефактів програмного забезпечення за допомогою значущих показників. Крім того, моніторинг необхідно визначати відповідно до конкретних цілей для вимірювання артефактів програмного забезпечення за допомогою значущих показників. У цьому відношенні GQM

(ціль/запитання/метрика) – це підхід до застосування метрик для покращення процесу розробки програмного забезпечення (і, відповідно, згенерованих програмних продуктів) при збереженні бізнес-цілей організації та технічних цілей. Запис і читання великих даних вимагає ретельного аналізу шаблонів використання, щоб знайти відповідні конструкції ключів рядків, щоб дані були розділені, а навантаження рівномірно розподілялося між вузлами.

Це низхідний підхід до встановлення систематичних вимірювань цілей, пов'язаних із процесом розробки. Команда встановлює організаційні цілі, встановлює цілі вимірювання, вводить запитання для досягнення конкретних цілей і визначає показники, які дають відповіді.

Дослідження, зосереджене на програмі у віртуалізованому середовищі, дає змогу статично визначати коригування розподілу ресурсів, віддаючи пріоритет підвищенню продуктивності різних машин, які складають цю програму. Тому, оскільки організації все частіше використовують віртуалізацію для зменшення операційних витрат на фізичні сервери в центрі обробки даних і консолідації додатків, виникають занепокоєння щодо паралельності ресурсів у консолідованому віртуалізованому середовищі. Як наслідок, критичні програми більш вразливі до вузьких місць у продуктивності, таких як транзакції бази даних.

Використання мобільних додатків і Інтернету зростає, а генерація масивних неструктурованих даних призвела до винаходу різноманітних сховищ даних NoSQL. У результаті вимоги до веб-масштабу зростають щодня, а бази даних NoSQL розвиваються, щоб відповідати галузевим вимогам.

Бази даних, орієнтовані на стовпці, також відомі як розширювані сховища записів і великі стовпчасті сховища. Усі сховища створені за мотивами Bigtable від Google, розподіленої системи зберігання для керування структурованими даними, призначені для масштабування до величезного розміру.

На відміну від реляційних баз даних, сховища стовпців у NoSQL є гібридними сховищами рядків/стовпців. Хоча сховища стовпців використовують масово розподілену архітектуру для зберігання даних замість таблиць, вони

поділяють ідею зберігання по стовпцях зі стовпчастими базами даних і стовпцевими розширеннями для баз даних на основі рядків. Кожен ключ у сховищі стовпців пов'язаний з одним або кількома атрибутами. Сховище стовпців зберігає свої дані, щоб їх можна було швидко агрегувати з меншою активністю введення-виведення. Він пропонує високу масштабованість зберігання даних (див. рис. 3.1).

Parameter	Column Databases	Relational Databases
Type	It is a column oriented data store	It is a row oriented data store
Compression Rates	These type of data stores basically permits high compression rates due to little distinct or unique values in columns	Typical compression mechanisms which provide less efficient result Then what we achieve from column-oriented data stores
Best suitable For	Column Oriented databases best suitable for OLAP (Online Analytical Processing) System	Relational databases best suitable for OLTP (Online Transaction Processing) System
Scalability	Columnar Databases can be horizontally scalable by data partitioning.	Relational Databases can be vertically scalable by upgrading hardware.
Performance	When massive amount of data comes every second, than high performance and high availability characteristic help column oriented database.	When massive amount of data comes every second, Then due to full ACID property support, join and locks adverse performance can be seen.

Рисунок 3.1 – Порівняння реляційних та стовпчикових СКБД

Дані, що зберігаються в базі даних, базуються на порядку сортування сімейства стовпців. Стовпці можна розділити на сімейства стовпців, що має вирішальне значення для поділу та впорядкування даних. Під час виконання стовпці та рядки можна гнучко додавати. Тим не менш, сімейства стовпців часто потрібно визначати заздалегідь, що призводить до меншої гнучкості, ніж ключ-значення або сховища документів. Сімейство сховищ даних стовпців включає Hbase , Hypertable і Cassandra.

Інструменти порівняльного аналізу допомагають генерувати синтетичні дані, рядок випадкових символів, індексованих основним ідентифікатором. Крім того, він має виконавець робочого навантаження для обробки кількох клієнтських потоків і виконує певні операції CRUD.

Виконаємо порівняння реляційної СКБД MySQL та нереляційної документо-орієнтованої СКБД MongoDB. Для обох моделей розглядалися різні профілі навантаження. Це:

– профіль A, 50% читання, 50% запис;
– профіль B, 100% запис;
– профіль C, 100 читання;
– час виконання та час передачі (пропускна здатність).

Тести проводилися на одному комп'ютері з ЦП Intel i9 9880H, 2,3 ГГц, 32 ГБ пам'яті та 500 ГБ SDD. Розглянемо результати тестування.

На рисунку 3.2 показано результати визначення пропускної здатності для кожного з трьох профілів.

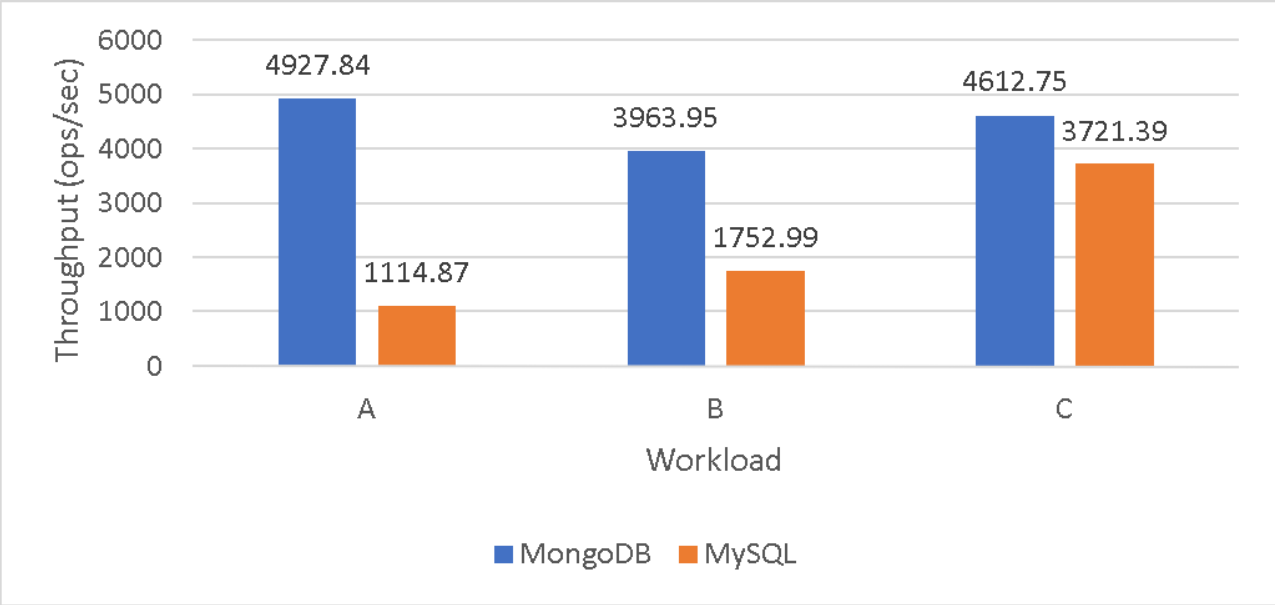

Рисунок 3.2 – Порівняння пропускної здатності двох СКБД для кожного профілю

Рисунок 3.3 ілюструє показники часу виконання операцій читання/запису.

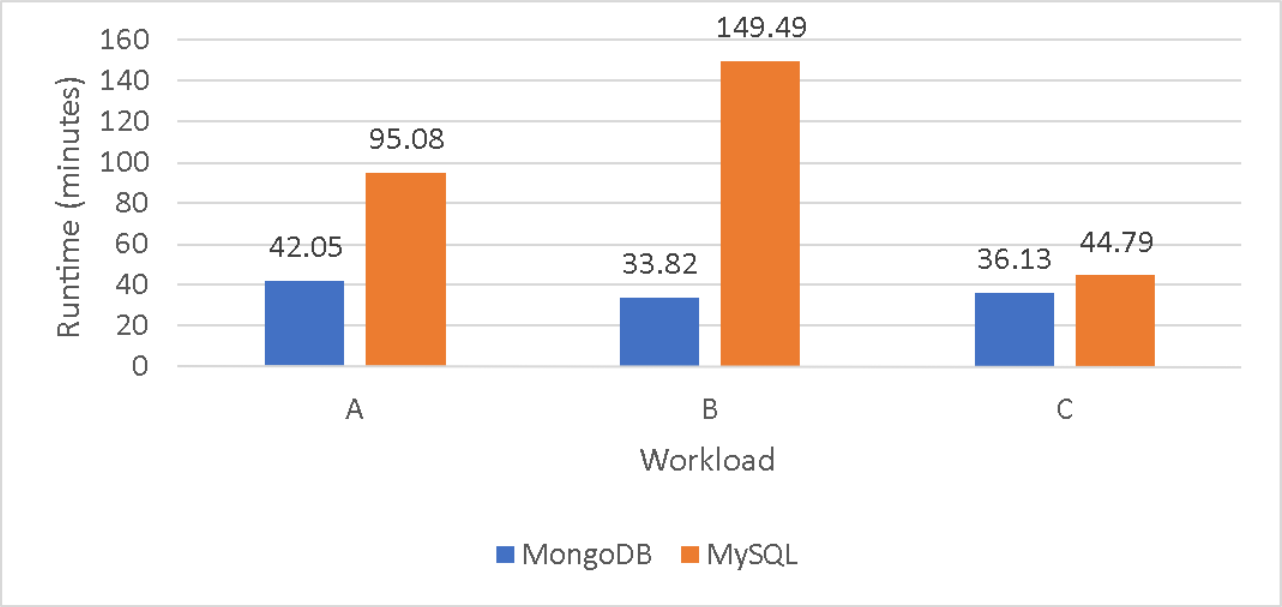

Рисунок 3.3 – Порівняння часу виконання

Для профілю навантаження A маємо 50% читання та 50% операцій запису, а також 10000000 записів. Бази даних MySQL і MongoDB протестовано, і MongoDB працює на 55,77% краще. Однак MySQL демонструє нижчу продуктивність – 95 хвилин і 8 секунд, тоді як MongoDB працює краще – 42 хвилини і 5 секунд.

Як бачимо, для великого обсягу даних показники нереляційної СКБД переважають показники реляційної.

4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1 Охорона праці та її актуальність в ІТ-сфері

Для підвищення ефективності системи управління охорони праці (СУОП) дуже важлива роль належить формуванню і розвитку інформаційної культури фахівців ІТ-технологій, яка впливає на удосконалення інформаційного контуру сучасних підприємств, дозволяє створювати надійні прогнози щодо стану умов праці, показників здоров'я та працездатності, виробничого травматизму і професійної захворюваності, визначати політику розвитку підприємств, установ та організацій на основі різноманітних стратегій охорони праці (інноваційні, маркетингові, інвестиційні, фінансові, технологічні, диверсифікаційні). Поряд з інформаційною культурою важливо використовувати в рамках СУОП «трикутник» її складових: правову, організаційну, управлінську.

В управлінні охороною праці потрібно реалізувати основні положення, окремі теоретико-методологічні підходи інформаційного менеджменту. Головну роль та відповідальність за стан СУОП мають нести фахівці служби охорони праці сучасного підприємства.

Сучасне суспільство називають постіндустріальним, постекономічним, інформаційним, оскільки йдеться про багатосторонні і кардинальні зміни у розвитку цивілізації.

Інформаційне суспільство передбачає докорінну зміну, яка полягає у перетворенні інформації і знань у головний професійно-виробничий потенціал особистості, соціуму і держави.

На постіндустріальному етапі розвитку суспільства вирішальним фактором стає інформація. Її домінування ініціювала науково-технічна революція, яку ще іменують інформаційною, оскільки нею охоплена будь-яка інтелектуальна діяльність, починаючи з інформаційних образів штучного

інтелекту у нових технологіях, економіки, і продовжуючи інформатизацією суспільства в умова світової глобалізації науки й освіти тощо.

Інформаційні технології розглядаються як потужний важіль економічного зростання України. Для цього необхідні значні стратегічні інвестиції у комп'ютерну та комунікаційну інфраструктуру, програми досліджень і розробок, освітню галузь.

Під інформаційною культурою розуміють сукупність, складову НІТ (новітні інформаційні технології), технологічну, правову, психологічну, соціологічну та ергономічну підсистеми, що сприяють спрямованому впливу на протікання соціальних процесів у суспільстві, колективі і вихованню свідомого відношення людини до праці, виконання прав та обов'язків.

Поняття інформаційної культури виникло в процесі активізації дослідницької уваги до механізмів інформаційного обміну у зв'язку зі значним підвищення ролі інформації в соціокультурних процесах суспільства, яке розглядають як інформаційне суспільство знань, де в центрі знаходяться інформаційні технології.

Робота з інформацією та інформаційна культура в цілому є одним з найважливіших компонентів спроб компанії управляти змінами. Є три принципові причини, в силу яких сьогодні необхідно дбати про інформаційну культуру компанії.

По-перше, вона все більше і більше стає найважливішою частиною загальної організаційної (корпоративної) культури компанії. Все більше компаній розуміють необхідність перетворень, орієнтованих на задоволення очікувань споживача. Щоб сьогодні впливати на майбутнє, потрібно уявляти собі на що вона буде схожа. А для цього потрібно працювати з різноманітною діловою, професійною, технологічною, соціальною, ринковою та політичною інформацією.

По-друге, інформаційні технології роблять можливим створення в компаніях комп'ютерних мереж, за допомогою яких йде спілкування між менеджерами, але важливо знати, як люди використовують цю інформацію. Саме

по собі створення такої мережі з усіма її робочими станціями і мультимедійними можливостями не гарантує того, що інформація буде використовуватися більш розумно і більш ефективно.

По-третє, для різних функціональних служб, підрозділів та робочих груп сучасних підприємств в сфері охорони праці інформаційна культура різна, а це означає відмінність методологічних підходів до процесів усвідомлення, збору, організації, обробки, поширення і використання інформації. Тому багато менеджерів погодяться з тим, що корпоративна інформаційна культура важлива для вироблення різних стратегій охорони праці та запровадження відповідних заходів з її вдосконалення.

Для деяких галузей, таких як розробка програмного забезпечення, інформаційна культура є необхідною умовою виживання, тому що зміна технологій в розробці програмного забезпечення відбувається кожні 6-8 місяців, а інвестиції на підготовку персоналу і освоєння нової технології величезні і у великих компаніях варіюються від 1,5 до 2 млрд. доларів на рік.

Аналіз свідчить, що інформатизація та інтеграція комунікаційного простору України сприяє різкому підвищенню інформаційної та професійної компетентності, ділової активності, стимулюванню конкуренції, створенню інноваційних підприємств та організацій, нових робочих місць, зниженню витрат на утримання управлінського апарату.

Поряд із задачами і здобутками окреслилися негативи використання інформаційних технологій:

1) надмірне інформаційне навантаження, суть якого полягає у тому, що кількість корисної інформації, яка надходить до мережі, перевищує психофізіологічні можливості її сприйняття людиною;

2) велика кількість інформації, яка сприймається, але не є корисною для фахівців в даний момент;

3) інформаційний голод, причиною якого є саме надлишок інформації, викликаний інформаційним перенавантаженням;

4) «інформоманія» як хвороба людини, яка робить останню знеособленою, залежною від перебування в інформаційному просторі і роботи з комп’ютером і чому вона віддає перевагу, уникаючи «живого» спілкування з людьми;

5) поява «кіберспільнот», що за своїми соціокультурними характеристиками набагато ближчі до представників інших культур у глобальному інформаційному просторі, ніж до своєї етнонаціональної спільноти чи решти населення, не охопленого Інтернетом;

6) індивідуалізм і дегуманізація способу життя «мешканців» Інтернету – відсутність готовності ділитися своїми знаннями.

Слід розуміти, що комп'ютерні технології, а особливо їх мережі істотно впливають на життєдіяльність людини, припускаючи глобалізацію і технократизацію суспільства. Але в ще більшій мірі цей вплив поширюється безпосередньо на центральну нервову систему, яка звикає працювати в дуже інтенсивному режимі багатозадачності, де вже переважають не тривалі логічні роздуми, а інтуїтивно-реактивні ланцюжки розумових формулювань у зв'язку з величезним обсягом оброблюваної щодня інформації, кількість якої зростає за експоненціальною швидкістю. Виникає припущення, що саме збільшення обсягу інформації та прискорення її обробки людиною може згубно вплинути на розвиток розумових здібностей людини.

Аналіз продуктивності розумової праці в найбільших за чисельністю фахівців IT-фірм показав, що велике значення з точки зору впливу на її результати має організаційна (корпоративна) культура. В цьому напрямі влаштовуються різні тімбілдинги, заходи, тренінги для розвитку персоналу. Також кожен керівник повинен добре розуміти свого співробітника, що саме для нього важливо, що його мотивує. Важливо відвести потрібну роль відповідному співробітнику, щоб він виконував ті завдання, які йому цікаві.

На подібних тренінгах в тому числі повинна розглядатися інформаційна культура працівника, в освоєнні, володінні, мотивуванні, застосуванні, перетворенні інформації із застосуванням сучасних інформаційних технологій і

використанням цих умінь в навчанні з охорони праці і в подальшій професійній діяльності. Особливо вони будуть корисні, як доповнення до існуючих інструктажів з охорони праці на підприємстві, або як контроль психологічного стану та взаємовідносин у колективі.

Інформаційна культура як інтегративне утворення абсолютно не зводиться до розрізнених знань, вмінь та навичок роботи за комп'ютером. Вона передбачає інформативну спрямованість цілісної особистості, яка володіє мотивацією до застосування і засвоєння нових даних. Інформаційну культуру можна розглядати, як одну з граней особистісного розвитку промислових робітників. Це шлях універсалізації якостей людини.

Оволодіння інформаційною культурою сприяє реальному розумінню особистістю свого місця, себе і своєї ролі у виробничому колективі. Вона має сприяти формуванню нового покоління фахівців інформаційного суспільства, який повинен володіти наступними навичками: виділення релевантної, значущої інформації, диференціації вихідних даних, розробки інформативних критеріїв її оцінки інформації, вміло використовувати її в рамках СУОП.

Сьогодні продовжує діяти стратегічне правило «Можливості комп’ютерної техніки обмежені тільки нашими уявленнями».

4.2 Шкідлива дія шуту та вібрації і захист від неї

Для запобігання шкідливої дії шуму і вібрації на організм працюючих проводяться технічні, організаційні і медикопрофілактичні заходи.

Одним з основних технічних заходів є зменшення при експлуатації та на стадії проектування, конструювання обладнання причин шуму і вібрації в самому джерелі утворення. Досягають цього завдяки використанню раціональної конструкції обладнання, заміни ударної дії деталей і машин коливальною, з’єднання елементів гнучкими зв’язками, врівноважування обертових частин механізмів, заміни металевих деталей пластмасовими, забезпечення різних власних частот коливань механізму з частотою збуджуючої сили.

Аеродинамічний шум може бути зменшений застосуванням глушників та повітропроводів зі змінним перерізом. Шум трансформаторів (електромагнітний шум) знижується, якщо застосувати листи заліза як складових осердя трансформатора з малою магнітострикцією, серцевини.

Якщо неможливо ізолювати чи знизити шум і вібрацію самого джерела, потрібно:

– ізолювати джерело шуму або вібрації від навколишнього середовища засобами вібро- та звукоізоляції

– раціонально планувати виробничі приміщення, що мають інтенсивні джерела шуму;

– збільшувати звукопоглинання внутрішніх поверхонь приміщення шляхом звукопоглинальних покриттів.

Принцип роботи звукоізоляційних екранів оснований на відбиванні звукової хвилі від різних екранів, стін, кожухів обладнання. Шумливі агрегати слід закривати звукоізоляційними кожухами з виводом назовні органів керування та контрольних приладів. Звукоізоляційні екрани виготовляють з металу, деревини, пластмаси та інших щільних матеріалів. Екрани зсередини покривають звукопоглинаючими матеріалами (скловатою пінополіуретаном), а по периметру кожуха – віброізоляційними підкладками (гума).

Вихідними даними для розрахунку параметрів необхідного екрану є спектр шуму, який необхідно ослабити, кількість екранів, через які проходить шум, їх площа, акустичні характеристики приміщення.

За розрахованими значеннями необхідної звукової ізоляційної здатності екрану підбирається матеріал конструкції й екрану.

Принцип звукопоглинання оснований на явищі трансформації коливальної енергії звуку в теплову через втрати при терті. Найбільші втрати при терті мають пористі, волокнисті і перфоровані матеріали: поролон, пемзолітові і деревоволокнисті плити тощо.

Енергія звукової хвилі переходить у теплову енергію, причому, ефект звукоізоляції збільшується з ростом частоти звукової хвилі. Звукопоглинаючими

матеріалами оббивають стелі, стіни. Щоб одержати ефективну звукоізоляцію, найбільш доцільно застосовувати багатошарові огородження з м'якими прошарками (мінеральна вата).

Важливим технічним рішенням у забезпеченні виробничих умов є вдосконалення ручних віброінструментів. Для цього використовують віброгасіння, змінюють ударний вузол, проводять балансування частин, що обертаються.

Послаблення локальної вібрації і передачі вібрації на підлогу і сидіння досягається засобами віброізоляції і вібропоглинання, застосуванням пружинних і гумових амортизаторів, прокладок тощо. Для обмеження поширення вібрацій через ґрунт, між фундаментом і ґрунтом залишають повітряні проміжки, які називаються акустичними розривами.

В останні роки знаходять застосування динамічні віброгасники, в яких створюються вібрації, що співпадають по частоті і протилежні по фазі вібрації машини, коливання якої необхідно зменшити.

До організаційних заходів по боротьбі з шумом та вібрацією на виробництві відносяться: впровадження раціонального режиму праці і відпочинку, обмеження часу роботи при використанні ручного інструменту, який створює вібрацію.

Глушники звуку застосовуються для зменшення шуму аеродинамічних установок (вентиляторів, пневмоінструментів, газотурбінних, дизельних, компресорних установок). Вони поділяються на активні, які поглинають звукову енергію, що на них поступила, і реактивні, які відбивають цю енергію. Потужні джерела шуму як правило розміщують в окремих приміщеннях, які віддалені від постійних робочих місць.

Ізоляційні кабіни або екрани застосовують як екрани робочих місць для зменшення зовнішніх шумів.

Якщо не вдається зменшити рівень шуму і вібрації на робочому місці до нормативних значень та необхідно використовувати засоби індивідуального

захисту: рукавиці, взуття, навушники, м’які шоломи, які зменшують рівень звукового тиску на 40-50 дБ.

У процесі виробництв, експлуатації і зберігання радіоелектронних засобів можуть виникати механічні і динамічні дії, що характеризуються широким діапазоном частот коливань, а також амплітудою, прискоренням і часом дії. Рівень механічних дій визначається умовами транспортування й експлуатації.

Необхідно розрізняти два види механічних дій: удари і вібрації. Удар виникає, коли апаратура отримує швидку зміну прискорення (піддаються удару входи кабелів, джгути, резистори, конденсатори, напівпровідникові діоди і тріоди, силові трансформатори, дроселі тощо). Вібрації – довготривалі знакозмінні процеси, які впливають на роботу апаратури при безпосередньому контакті з джерелом коливань або через повітряне середовище.

У результаті дії вібрацій і удару можуть бути наступні ушкодження апаратури: порушення герметичності через псування паяльних, зварних і клеєних швів і появи тріщин у метало-скляних спаях; повне руйнування корпусів або окремих їх частин через механічний резонанс або циклічну втому; обривання монтажних зв’язків, відшарування багатошарових друкованих плат, руйнування підставок; вихід з ладу електричних контактів; модуляція розмірів хвилеводних трактів; коаксіальних кабелів, конденсаторів змінної ємності, коливальних контурів, електровакуумних приладів, зміщення положення органів настроювання і управління.

Під впливом вібрацій може статись зміна параметрів напівпровідникових приладів, вольт амперних характеристик діодів, транзисторів. Все це призводить до руйнування конструкцій за рахунок явищ втоми.

Радіоелектронна апаратура (РЕА) повинна мати віброміцність, вібростійкість, ударостійкість.

Захист РЕА здійснюється наступними групами методів:

– зменшується інтенсивність джерел вібрації шляхом балансування, зменшення зазорів, віброізоляції джерела вібрацій;

– зменшується величина дій, що передається апаратом шляхом віброізоляції, демпфірування, виключення резонансів, активного віброзахисту за допомогою ексцентриків, маятників, гіроскопів;

– використання найбільш добротні і жорсткі компоненти і вузли;

– застосовуються амортизатори.

Захист часом, захист віддалю, усунення джерела тепловиділення, теплоізоляція, охолодження гарячої поверхні, забезпечення тепловіддачі тіла людини та індивідуальні засоби захисту.

Захист часом передбачає обмеження часу перебування робітника в зоні дії інфрачервоного випромінювання. Потужність випромінювання можна знизити за рахунок конструкторських і технологічних рішень (зміною нагрівання виробів у нагрівальних пічках індукційним нагріванням та ін.) і за рахунок покриття поверхні, яка нагрівається, тепло ізолювальним матеріалом.

Якщо теплоізоляція неможлива, тоді захист від прямої дії інфрачервоного випромінювання здійснюється екрануванням.

Екрани можуть бути прозорими, напівпрозорими і непрозорими.

У свою чергу вони поділяються на тепловідбивальні, тепловідвідні та теплопоглинальні; стаціонарні і нестаціонарні.

Застосовують також прозору водяну завісу у вигляді суцільної тонкої водяної плівки. Вода є активним поглиначем інфрачервоного випромінювання.

Перегрівання людини попереджують раціональним режимом пиття, режимом праці та гідро процедурами. Спецодяг виготовляється з незаймистого, стійкого до інфрачервоного випромінювання, м’якого і повітронепроникного матеріалу (тканина з металевим покриттям відбиває 90 % інфрачервоного випромінювання).

Для захисту очей застосовують світлофільтри зі спеціального жовто-зеленого або синього скла.

Першочергові заходи – це конструкторські і технологічні рішення, які виключають генерацію або понижують інтенсивність випромінювання. Спеціальні засоби захисту (екранування джерел випромінювання, фарбування

стін у світлі кольори) попереджують розповсюдження і знижують інтенсивність цих випромінювань у виробничих приміщеннях. Очі захищають окулярами або щитками зі склом – світлофільтром. Для захисту шкіри використовують мазі з речовинами – світлофільтрами для цих променів (салол, саліцилово-метиловий ефір та ін.), а також спецодяг з бавовняних тканин і грубововняного сукна. Руки захищають рукавицями.

ВИСНОВКИ

Результати показують, що немає необхідності переходити від реляційних баз даних до баз даних NoSQL. Оскільки обидва типи баз даних мають переваги та недоліки, компанія може вибрати СКБД, яка найкраще відповідає її вимогам. Організація може використовувати базу даних SQL, якщо вона надає пріоритет стандартизації та узгодженості даних. NoSQL слід використовувати, коли бізнес має велику кількість неструктурованих даних і доступність даних є високою вимогою. Для агрегації невеликих наборів даних можна віддати перевагу реляційній базі даних перед базою даних NoSQL і навпаки для аналітики великих даних.

MapReduce найбільше підходить для використання в кластерах завдяки своїй розподіленій природі. Хоча MapReduce працює повільно під час процесу агрегації, він більш ефективний у паралельних обчисленнях і розроблений для обробки величезних обсягів неструктурованих даних. Бази даних NoSQL, завдяки своїй розсіяній та високомасштабованій природі, добре підходять для програм, які генерують об’ємні дані з різноманітними функціями. Коли залучаються геопросторові дані, масштабованість реляційних баз даних вища, ніж у баз даних NoSQL. Бази даних NoSQL мають швидший час відповіді на дані, ніж реляційні бази даних, особливо при обробці величезних обсягів геопросторових даних.

Сховища даних NoSQL пропонують альтернативу звичайним RDBMS, але ймовірно, що організації не можуть відразу вирішити, який з них використовувати. Оптимальна стратегія вибору відповідної бази даних NoSQL полягає у визначенні того, що вимагає програма, а система керування реляційною базою даних не може забезпечити. Якщо система керування реляційною базою даних (RDBMS) може ефективно керувати даними, система зберігання NoSQL може не знадобитися.

Крім того, загальновизнано, що база даних NoSQL є новою розробкою в просторі баз даних. Однак вони розробляються з використанням

загальновизнаних теорій. Незважаючи на переваги, системи на основі NoSQL не позбавлені недоліків, таких як відсутність загальноприйнятих стандартів або добре відомої мови запитів для використання з базами даних NoSQL. Кожна база даних має свої особливості та способи роботи. Навпаки, ці набори даних є новими та динамічними. Немає гарантії, що всі дані будуть правильно записані в сховище даних, оскільки бази даних NoSQL не пропонують строгих характеристик ACID.

Крім того, бази даних NoSQL спрощують швидку розробку завдяки своїй динамічній/гнучкій схемі. На відміну від жорсткої структури реляційних баз даних, моделі та архітектури баз даних NoSQL є більш адаптивними.

Перехід від однієї моделі зберігання до іншої є складним для розробників через різні моделі, які використовуються базами даних NoSQL. Різні CSP використовують несумісні протоколи та інтерфейси, що робить їх несумісними один з одним. Оскільки кожен CSP розробив власні API для своїх конкретних служб, стає важко обмінюватися даними між ними. Для ефективного керування портативністю та сумісністю даних бази даних NoSQL потребують єдиного стандартизованого хмарного рішення.

Майбутні напрямки роботи щодо структурованих даних включають прийняття денормалізованої стратегії для SQL RDBMS і порівняння результатів вставки, оновлення та отримання даних у MongoDB з даними в іншій системі. Натомість ми можемо порівняти продуктивність MongoDB і SQL RDBMS у конкретному сценарії, стежачи за нормалізованим підходом, використаним першим, а потім оцінюючи результати вставки, оновлення та отримання даних. Паралельні геопросторові підходи, що використовуються базами даних NoSQL, вимагають більше уваги, якщо вони хочуть ефективно обслуговувати велику кількість користувачів. Комп’ютерний зір, ідентифікація об’єктів, класифікація сигналів та різноманітні інші програми глибинного навчання – це лише деякі з програм, заснованих на штучному інтелекті, які можуть отримати користь від розширення методології великих даних.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ihor, Bodnarchuk, et al. Multicriteria choice of software architecture using dynamic correction of quality attributes. In: International Conference on Computer Science, Engineering and Education Applications. Cham: Springer International Publishing, 2019. p. 419-427.

2. Bodnarchuk, I., Duda, O., Kharchenko, A., Kunanets, N., Matsiuk, O., & Pasichnyk, V. (2020). Choice Method of Analytical Platform for Smart City (No. 4374). EasyChair.

3. Kharchenko, A., Raichev, I., Bodnarchuk, I., & Matsiuk, O. (2021, October). The Survey of Global Software Design Processes. In 2021 IEEE 8th International Conference on Problems of Infocommunications, Science and Technology (PIC S&T) (pp. 291-294). IEEE.

4. Siddiqa, A.; Hashem, I.A.T.; Yaqoob, I.; Marjani, M.; Shamshirband, S.; Gani, A.; Nasaruddin, F. A survey of big data management: Taxonomy and state-of-the-art. J. Netw. Comput. Appl. 2016, 71, 151–166.

5. Kong, X.; Shi, Y.; Yu, S.; Liu, J.; Xia, F. Academic social networks: Modeling, analysis, mining and applications. J. Netw. Comput. Appl. 2019, 132, 86–103.

6. Ordonez, C. (2009). Optimization of linear recursive queries in SQL. IEEE Transactions on knowledge and Data Engineering, 22(2), 264-277.

7. Obasanjo, D. Building scalable Databases: Denormalization, the NoSQL movement and Digg. 2009.

8. Strozzi, C. (1998). NoSQL: A relational database management system. Lainattu, 5, 2014.

9. George, S. NoSQL—NOT ONLY SQL. Int. J. Enterp. Comput. Bus. Syst. 2013, 2.

10. Alsolai, H.; Roper, M. A systematic literature review of machine learning techniques for software maintainability prediction. Inf. Softw. Technol. 2020, 119, 106214.

11. Hou, B.; Qian, K.; Li, L.; Shi, Y.; Tao, L.; Liu, J. MongoDB NoSQL injection analysis and detection. In Proceedings of the 2016 IEEE 3rd International Conference on Cyber Security and Cloud Computing (CSCloud), Beijing, China, 25–27 June 2016; pp. 75–78.

12. Padhy, R.P.; Patra, M.R.; Satapathy, S.C. RDBMS to NoSQL: Reviewing some next-generation non-relational Database's. Int. J. Adv. Eng. Sci. Technol. 2011, 11, 15–30.

13. Gyo″rödi, C.; Gyo″rödi, R.; Pecherle, G.; Olah, A. A comparative study: MongoDB vs MySQL. In Proceedings of the 2015 13th International Conference on Engineering of Modern Electric Systems (EMES), Oradea, Romania, 11–12 June 2015; pp. 1–6.

14. Băzăr, C.; Iosif, C.S. The transition from rdbms to nosql. a comparative analysis of three popular non-relational solutions: Cassandra, mongodb and couchbase. Database Syst. J. 2014, 5, 49–59.

15. Mukherjee, S. The Battle between NoSQL Databases and RDBMS; University of the Cumberlands: Williamsburg, KY, USA, 2019.

16. Chopade, R.; Pachghare, V.K. Ten years of critical review on database forensics research. Digit. Investig. 2019, 29, 180–197.

17. McColl, R.C.; Ediger, D.; Poovey, J.; Campbell, D.; Bader, D.A. A performance evaluation of open source graph Databases. In Proceedings of the First Workshop on Parallel Programming for Analytics Applications, Orlando, FL, USA, 16 February 2014; pp. 11–18.

18. Anikin, D.; Borisenko, O.; Nedumov, Y. Labeled Property Graphs: SQL or NoSQL? In Proceedings of the 2019 Ivannikov Memorial Workshop (IVMEM), Velikiy Novgorod, Russia, 13–14 September 2019; pp. 7–13.

19. Jung, M.-G.; Youn, S.-A.; Bae, J.; Choi, Y.-L. A study on data input and output performance comparison of MongoDB and PostgreSQL in the big data environment. In Proceedings of the 2015 8th International Conference on Database Theory and Application (DTA), Jeju Island, Republic of Korea, 28–25 November 2015; pp. 14–17.

20. Wei-Ping, Z.; Ming-Xin, L.I.; Huan, C. Using MongoDB to implement textbook management system instead of MySQL. In Proceedings of the 2011 IEEE 3rd International Conference on Communication Software and Networks, Xian, China, 27–29 May 2011; pp. 303–305.

21. Aghi, R.; Mehta, S.; Chauhan, R.; Chaudhary, S.; Bohra, N. A comprehensive comparison of SQL and MongoDB Databases. Int. J. Sci. Res. Publ. 2015, 5, 1–3.

22. Faraj, A.; Rashid, B.; Shareef, T. Comparative study of relational and non-relations Database performances using Oracle and MongoDB systems. J. Impact Factor 2014, 5, 11–22.

23. Lawrence, R. Integration and virtualization of relational SQL and NoSQL systems including MySQL and MongoDB. In Proceedings of the 2014 International Conference on Computational Science and Computational Intelligence, Kunming, China, 15–16 November 2014; Volume 1, pp. 285–290.

24. Zhao, G.; Huang, W.; Liang, S.; Tang, Y. Modeling MongoDB with relational model. In Proceedings of the 2013 Fourth International Conference on Emerging Intelligent Data and Web Technologies, Washington, DC, USA, 9–11 September 2013; pp. 115–121.

25. Stanescu, L.; Brezovan, M.; Burdescu, D.D. Automatic mapping of MySQL Databases to NoSQL MongoDB. In Proceedings of the 2016 Federated Conference on Computer Science and Information Systems (FedCSIS), Gdansk, Poland, 11–14 September 2016; pp. 837–840.

26. Chandra, D.G. BASE analysis of NoSQL database. Futur. Gener. Comput. Syst. 2015, 52, 13–21.

27. Lee, C.-H.; Zheng, Y.-L. Automatic SQL-to-NoSQL schema transformation over the MySQL and HBase Databases. In Proceedings of the 2015 IEEE International Conference on Consumer Electronics-Taiwan, Taipei, Taiwan, 6–8 June 2015; pp. 426–427.

28. Pereira, D.A.; de Morais, W.O.; Pignaton de Freitas, E. NoSQL real-time Database performance comparison. Int. J. Parallel Emergent Distrib. Syst. 2018, 33, 144–156.

29. Kabakus, A.T.; Kara, R. A performance evaluation of in-memory databases. J. King Saud Univ. Inf. Sci. 2017, 29, 520–525.

30. Simanjuntak, H.T.A.; Simanjuntak, L.; Situmorang, G.; Saragih, A. Query Response Time Comparison NOSQLDB MONGODB with SQLDB Oracle. JUTI J. Ilm. Teknol. Inf. 2015, 13, 95–105.

31. Pokorný, J. Database technologies in the world of big data. In Proceedings of the 16th International Conference on Computer Systems and Technologies, Dublin, Ireland, 25–26 June 2015; pp. 1–12.

32. Moniruzzaman, A.B.M.; Hossain, S.A. Nosql Database: New era of Databases for big data analytics-classification, characteristics and comparison. arXiv 2013, arXiv:1307.0191.

33. Zeng, N.; Zhang, G.-Q.; Li, X.; Cui, L. Evaluation of relational and NoSQL approaches for patient cohort identification from heterogeneous data sources. In Proceedings of the 2017 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), Kansas City, MO, USA, 13–16 November 2017; pp. 1135–1140.

34. Lee, J.-G.; Kang, M. Geospatial Big Data: Challenges and Opportunities. Big Data Res. 2015, 2, 74–81.

35. Liu, Z.; Guo, H.; Wang, C. Considerations on Geospatial Big Data. IOP Conf. Ser. Earth Environ. Sci. 2016, 46, 012058.

36. Fatima, H.; Wasnik, K. Comparison of SQL, NoSQL and NewSQL Databases for internet of things. In Proceedings of the 2016 IEEE Bombay Section Symposium (IBSS), Maharashtra, India, 21–22 December 2016; pp. 1–6.

37. Fraczek, K.; Plechawska-Wojcik, M. Comparative analysis of relational and non-relational Databases in the context of performance in web applications. In Proceedings of the International Conference: Beyond Databases, Architectures and Structures, Ustron´, Poland, 30 May–2 June 2017; pp. 153–164.

38. Hricov, R.; Šenk, A.; Kroha, P.; Valenta, M. Evaluation of XPath queries over XML documents using SparkSQL framework. In Proceedings of the International Conference: Beyond Databases, Architectures and Structures, Ustron´, Poland, 30 May–2 June 2017; pp. 28–41.

39. Yue, P.; Tan, Z. 1.06 GIS Databases and NoSQL Databases. Compr. Geogr. Inf. Syst. 2017, 50.

40. Bartoszewski, D.; Piorkowski, A.; Lupa, M. The comparison of processing efficiency of spatial data for PostGIS and MongoDB databases. In Proceedings of the Beyond Databases, Architectures and Structures. Paving the Road to Smart Data Processing and Analysis: 15th International Conference, BDAS 2019, Ustron´, Poland, 28–31 May 2019; pp. 291–302.

41. Kothuri, R.; Godfrind, A.; Beinat, E. Pro Oracle Spatial for Oracle Database 11g; Dreamtech Press: New Delhi, India, 2008.

42. Baralis, E.; Dalla Valle, A.; Garza, P.; Rossi, C.; Scullino, F. SQL versus NoSQL Databases for geospatial applications. In Proceedings of the 2017 IEEE International Conference on Big Data (Big Data), Boston, MA, USA, 11–14 December 2017; pp. 3388–3397.

43. Rodrigues, M.; Santos, M.Y.; Bernardino, J. Big data processing tools: An experimental performance evaluation. Wiley Interdiscip. Rev. Data Min. Knowl. Discov. 2019, 9, e1297.

44. Shirazi, M.N.; Kuan, H.C.; Dolatabadi, H. Design patterns to enable data portability between clouds' Databases. In Proceedings of the 2012 12th International Conference on Computational Science and Its Applications, Salvador, Bahia, 18–21 June 2012; pp. 117–120.

45. Zhou, L.; Fu, A.; Yu, S.; Su, M.; Kuang, B. Data integrity verification of the outsourced big data in the cloud environment: A survey. J. Netw. Comput. Appl. 2018, 122, 1–15.

46. Strozzi, C. (1998). NoSQL: A relational database management system. Lainattu, 5, 2014.

47. George, S. (2013). Nosql–not only sql. International Journal of Enterprise Computing and Business Systems, 2(2), 2-5.

48. Shirazi, M.N.; Kuan, H.C.; Dolatabadi, H. Design patterns to enable data portability between clouds' Databases. In Proceedings of the 2012 12th International Conference on Computational Science and Its Applications, Salvador, Bahia, 18–21 June 2012; pp. 117–120.

49. Silva, L.A.B.; Costa, C.; Oliveira, J.L. A common API for delivering services over multi-vendor cloud resources. J. Syst. Softw. 2013, 86, 2309–2317.

50. Liao, C.-S.; Shih, J.-M.; Chang, R.-S. Simplifying MapReduce data processing. Int. J. Comput. Sci. Eng. 2013, 8, 219–226. [CrossRef] 153. Silva, L.A.B.; Costa, C.; Oliveira, J.L. A common API for delivering services over multi-vendor cloud resources. J. Syst. Softw. 2013, 86, 2309–2317.

51. Alomari, E.; Noaman, A. SeCloudDB: A Unified API for Secure SQL and NoSQL Cloud Databases. In Proceedings of the 2019 3rd International Conference on Cloud and Big Data Computing, Oxford, UK, 28–30 August 2019; pp. 38–42.

52. Brewer, E. A. (2000, July). Towards robust distributed systems. In PODC (Vol. 7, No. 10.1145, pp. 343-477).

53. Solanke, G.B.; Rajeswari, K. SQL to NoSQL transformation system using data adapter and analytics. In Proceedings of the 2017 IEEE International Conference on Technological Innovations in Communication, Control and Automation (TICCA), Chennai, India, 6 April 2017; pp. 59–63.

ДОДАТКИ

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Тернопільський національний технічний університет імені Івана Пулюя (Україна)
Університет імені П'єра і Марії Кюрі (Франція)
Маріборський університет (Словенія)
Технічний університет у Кошице (Словаччина)
Вільнюський технічний університет ім. Гедимінаса (Литва)
Наукове товариство ім. Т.Шевченка

АКТУАЛЬНІ ЗАДАЧІ СУЧАСНИХ ТЕХНОЛОГІЙ

Збірник
тез доповідей

XIII Міжнародної науково-практичної конференції молодих учених та студентів
11-12 грудня 2024 року

УКРАЇНА
ТЕРНОПІЛЬ – 2024

Матеріали XIII Міжнародної науково-практичної конференції молодих учених та студентів «АКТУАЛЬНІ ЗАДАЧІ СУЧАСНИХ ТЕХНОЛОГІЙ» – Тернопіль, 11-12 грудня 2024 року

УДК 004.652; 004.652.4

Святослав Мельничук
Тернопільський національний технічний університет імені Івана Пулюя

ПІДХОДИ ДО ПОРІВНЯННЯ ПРОДУКТИВНОСТІ РЕЛЯЦІЙНИХ ТА НЕРЕЛЯЦІЙНИХ БАЗ ДАНИХ

Sviatoslav Melnychuk

APPROACHES TO THE COMPARISON OF PERFORMANCE FOR SQL AND NOSQL DATABASES

Порівнюючи продуктивності реляційних та нереляційних баз даних, слід оцінити кілька ключових аспектів [1]. По-перше, слід взяти до уваги складність запитів та робоче навантаження. Реляційні (SQL) бази даних, такі як PostgreSQL або MySQL, відмінно справляються зі складними багатотабличними з'єднаннями, які є звичайними для реляційних моделей. Однак ці операції можуть призвести до зниження продуктивності під час великих навантажень. Нереляційні (NoSQL) бази даних, такі як MongoDB або Cassandra, оптимізовані для більш простих, великих операцій читання та запису, часто перевершують SQL для випадків опрацювання великомасштабних, неструктурованих або напівструктурованих даних.

Вирішальну роль для продуктивності також відіграє здатність СКБД до масштабованості. Бази даних SQL в основному вертикально масштабовані, тобто підвищення їх продуктивності залежить від оновлення апаратного забезпечення. На противагу їм NoSQL бази даних розроблені для горизонтального масштабування, що дозволяє розподіляти дані між кількома серверами для обробки зростаючих навантажень. Ця властивість робить їх більш придатними для розподілених хмарних систем.

Іншим фактором для оцінки продуктивності систем керування базами даних різних типів є шаблони доступу до даних та індексування. Бази даних SQL часто використовують складні стратегії індексування для оптимізації продуктивності запитів, але це може призвести до росту затрат обчислювальних потужностей під час інтенсивних операцій запису. Бази даних NoSQL надають гнучкі параметри індексування, які можна адаптувати до конкретних випадків використання, таких як геопросторові запити або дані часових рядів, хоча ця гнучкість може відбуватися за рахунок оптимізації запитів для більш складних операцій.

Бази даних SQL дотримуються строгих принципів ACID, забезпечуючи надійну узгодженість, яка є важливою для таких додатків, як банківська справа чи електронна комерція. Це може викликати затримку, особливо в розподілених середовищах. Бази даних NoSQL, розроблені на основі кінцевої узгодженості (BASE), надають перевагу продуктивності та доступності, що робить їх більш ефективними в таких сценаріях, як канали соціальних мереж.

Також варто розглянути вплив складності моделі даних на продуктивність. Бази даних SQL з їх жорсткими вимогами до схеми можуть спричинити додаткові затрати під час адаптації до мінливих структур даних. Оскільки бази даних NoSQL не містять схем, вони можуть обробляти подібні зміни більш динамічно, підвищуючи продуктивність у гнучких середовищах.

Література

1. Ковтун, Б. В., Манич, А. М., & Романюк, О. В. (2020). Порівняльна характеристика реляційних та NoSQL баз даних (Doctoral dissertation, ВНТУ).