КІБЕРБЕЗПЕКА: освіта, наука, техніка
CYBERSECURITY: EDUCATION, SCIENCE, TECHNIQUE
№ 2 (30), 2025
Спеціальний випуск ISSN 2663 – 4023
Special issue
DOI 10.28925/2663-4023.2025.30.974
УДК 004.62 + 004.65

Цируль Олександр Олександрович
аспірант
Інститут телекомунікацій і глобального інформаційного простору НАН України, Київ, Україна
ORCID: 0009-0002-5945-5918
tsyruloleksandr@gmail.com

СТРУКТУРА ВИСОКОЕФЕКТИВНИХ СИСТЕМ УПРАВЛІННЯ ДАНИМИ NOSQL

Анотація. У статті досліджено архітектурні засади організації даних у високоефективних системах управління базами даних типу NoSQL, орієнтованих на документи. Автор аналізує структуру зберігання та обробки інформації від базових елементів – бітів і блоків – до сторінок, документів і колекцій, що формують багаторівневу модель управління даними. Показано, що документно-орієнтовані бази даних (MongoDB, RavenDB, LiteDB) забезпечують високу продуктивність і масштабованість завдяки напівструктурованому формату даних (JSON/BSON), гнучкій схемі та підтримці CRUD-операцій. Детально описано роль індексів, сторінкової організації та колекцій у підвищенні ефективності запитів і зменшенні витрат доступу до даних. Розглянуто моделі побудови індексів типу B-Tree та B-Tree+, механізми кешування, шардінгу, денормалізації та вибіркового індексування як основні методи оптимізації продуктивності. Підкреслено важливість балансу між вимогами швидкодії та надійності відповідно до принципів ACID і теореми CAP у розподілених середовищах. Порівняльний аналіз MongoDB, RavenDB і LiteDB продемонстрував відмінності у транзакційності, запитах, індексації та масштабованості, що дозволяє визначити оптимальне рішення залежно від типу застосунку. У висновках зазначено, що правильна архітектура організації даних у NoSQL-системах забезпечує високу ефективність управління великими обсягами інформації, гнучкість моделювання, надійність транзакцій і стійкість до навантажень, що є визначальними чинниками розвитку сучасних хмарних і розподілених систем зберігання даних.

Ключові слова: NoSQL; база даних; документ; колекція; сторінка; блок; біт; індекс; транзакція; оптимізація продуктивності.

ВСТУП

У складному за способами керування даними, бази даних NoSQL стали ключовою технологією, здатною впоратися з різноманітними вимогами сучасних програм [1]. Ці бази даних створені для забезпечення підвищеної гнучкості, масштабованості та продуктивності завдяки уникненню жорстких схем і реляційних структур традиційних баз даних. Бази даних NoSQL особливо вправні в управлінні великими обсягами різноманітних типів даних – неструктурованих, напівструктурованих або поліморфних за своєю природою.

Серед різних типів баз даних NoSQL системи на основі документів виділяються своєю здатністю зберігати дані в напівструктурованому форматі. Ці бази даних використовують документи як основну одиницю зберігання, яка може відрізнятися за структурою та зазвичай використовує JSON (об’єктна нотація JavaScript), BSON (двійковий JSON) або подібні формати. Зазначений формат дозволяє зберігати складні структури даних в одному документі, який можна легко індексувати та запитувати. На відміну від реляційних баз даних, які зберігають дані в рядках і стовпцях, бази даних документів зберігають дані у спосіб, який часто більше узгоджується зі структурами даних, які використовуються мовою програмування програми, забезпечуючи більш

інтуїтивно зрозумілу та плавну взаємодію з даними [2]. Бази даних NoSQL на основі документів, такі як MongoDB, CouchDB та інші, стали особливо популярними [3] для таких програм, як керування вмістом, каталоги та аналітика в реальному часі, де гнучкість схеми даних може призвести до значних переваг у продуктивності та масштабованості. Їм особливо сприяють сценарії, що вимагають швидкої розробки, частого внесення змін до структур даних і в середовищах зі змінним навантаженням [2]. Хоча гнучкість схеми прискорює початкову розробку, розумний вибір схем має вирішальне значення, оскільки вони суттєво впливають на продуктивність, впливаючи на надмірність даних, вартість навігації, вартість доступу до даних і ремонтопридатність. І хоча у низці робіт [3 – 5] підкреслюється важливість дослідження схем у сховищах документів NoSQL, та не в повній мірі враховується архітектура організації даних, хоча й робиться акцент на необхідності модернізації існуючих сховищ. Частково управління базами даних з врахуванням архітектури представлено у роботі [6], де зазначається, що саме різноманітність нереляційних баз даних міститься в їхній структурі зберігання даних, які дозволяють оперувати великою кількістю даних та проводити моделювання. Проте у зазначеній роботі розглянуті бази даних, орієнтовані на стовпці, документи та графіки.Проблема дослідження поглядає в тому, що постійне зростання соціальних мереж, нетрадиційних веб-технологій, мобільних додатків і пристроїв Інтернету речей (IoT) створює труднощі [7, 8] для хмарних систем даних у підтримці величезних наборів даних і дуже високих показників запитів. Бази даних NoSQL, такі як Cassandra та HBase, і реляційні бази даних SQL з реплікацією, такі як Citus/PostgreSQL, використовувалися для підвищення горизонтальної масштабованості та високої доступності систем зберігання даних. А при оцінці розподілених баз даних на кластері стандартних одноплатних комп’ютерів (SBC), таких, як реляційні бази даних Citus/PostgreSQL і NoSQL Cassandra і HBase, масштабування, еластичність і висока доступність стають пріоритетними, що вказує на важливість дослідження архітектури організації даних.

Метою дослідження є аналіз архітектури організації даних, спеціально розроблену для баз даних NoSQL на основі документів з побудовою відповідної моделі представлення даних. Основна увага націлена на критичні концепції: біта, блоку, документа, сторінки та колекції, кожна з яких відіграє значну роль у загальній моделі керування даними. Аналіз цих основоположних елементів не тільки покращить розуміння того, як бази даних NoSQL на основі документів ефективно керують даними, але й висвітлить їхні функціональні переваги та операційну механіку. Основною задачею роботи виступає деталізація рівнів структурування даних аж до самих бітів з поширенням на організаційні моделі з представленням у вигляді схем для керування великими кластерами даних.

ДОСЛІДЖЕННЯ СТРУКТУРИ ОРГАНІЗАЦІЇ ДАНИХ

Найфундаментальнішою одиницею даних в обчислювальній техніці є біт, скорочення від «двійкової цифри». Біти є будівельними блоками всіх даних і представлені двома станами, які зазвичай позначаються як 0 і 1. Кожен фрагмент інформації, який обробляється комп’ютером, на найнижчому рівні, є послідовністю бітів. Ці біти згруповані для формування складних структур даних, що забезпечує представлення та маніпулювання на вищих рівнях обчислювальної архітектури.

У контексті документних баз даних NoSQL біти залишаються остаточним показником обсягу даних. Коли дані зберігаються, незалежно від їхньої вищої структури

в документах чи колекціях, основний носій інформації зрештою записує цю інформацію у вигляді послідовностей бітів.

Піднімаючись на один рівень в ієрархії даних від бітів, ми зустрічаємо блоки. Блок — це безперервний набір бітів, який діє як фундаментальна одиниця зберігання даних у цифрових системах. Ці одиниці можуть коливатися від 512 байт до 4 КБ. Блоки служать основним засобом організації даних для запам'ятовуючих пристроїв; бази даних використовують ці блоки для читання та запису даних на диск.

Якщо взяти за основу розмір блоку 512 байт, тобто 4096 бітів то файл, що займає 3 блоки матиме розмір 12288 бітів, при конвертації в кілобайти дорівнюватиме 1.5 кілобайт.

Таблиця 1

Використання блоків для збереження файлів

Назва файлу	Початок	Кількість блоків	Блоки	Розмір файлу
file.txt	0	3	0,1,2	1.536 кілобайт
image.jpg	4	3	4,5,6	1.536 кілобайт
video.mp4	8	4	8,9,10,11	2.048 кілобайт

У цифрових системах зберігання, сторінка — це блок даних фіксованої довжини, який є основною одиницею керування даними та їх передачі між диском і пам’яттю в системі баз даних.. Підхід зі сторінками широко використовується як для реляційних баз даних та і не реляційних. Розмір сторінок для кожної бази даних може мати різне значення, для прикладу база даних Postgres використовує сторінки розміром 8192 байта, а MySql для прикладу 16384 байт.

Кожна сторінка має свій індекс, від якого залежить її місцезнаходження в файловій системі. При необхідності отримати дані однієї сторінки, відбувається запит до операційної системи на зчитування з файлу зі зміщенням, що читає 8192 байти, і відображає запитані байти на блоки файлової системи за допомогою вузла індексу файлової системи або inode, який містить метадані про файл. Кожна адреса блоку логічної файлової системи (LBA) відображається на певному фізичному блоці в пристрої зберігання даних. Якщо читати один байт, все одно відбудеться читання всього блок з диска. Для прочитання 8 сторінки, потрібно прочитати біти зі зсувом 8 * 8192 + 1 і отримати наступні 8198 байти (Рис. 1).

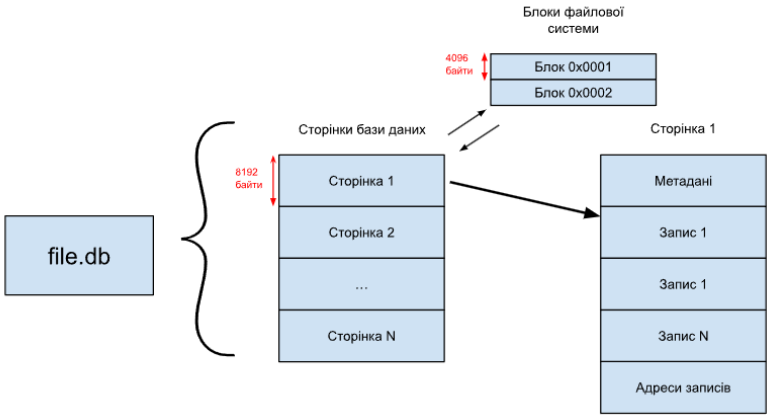

Рис 1. Схематичне зображення сторінок баз даних

Кожна сторінка складається з трьох основних компонентів: метадані сторінки, записи (документи) та адреси записів.

Метадані сторінки включають всю довідкову інформацію про сторінку, яка допомагає як у внутрішньому управлінні, так і в пошуковій оптимізації. Поля метаданих, зазвичай мають статичні розміри, відповідно можна завжди дізнатись де закінчується секція з метаданими і починається наступний блок.

Секція з документами займає основну частину сторінки бази даних та займається збереженням самих даних. Як правило, кожен документ в базі даних має унікальний ідентифікатор, який використовується для доступу та управління даними. Більш детально структуру документів описано у відповідному розділі. Записи можуть бути збережені в сторінці по порядку їх створення або можуть бути організовані за певними критеріями, наприклад за ключем або індексами, що оптимізують процеси пошуку та доступу до даних.

Секція адрес документів чи секцією вказівників, містить метадані, що вказують на позицію кожного документа в межах сторінки та його розмір. Це дозволяє системі баз даних швидко знаходити і читати записи за потреби, без необхідності читання всієї сторінки.

Інформація про позицію документа зазвичай включає зміщення від початку сторінки, де починається документ, в той час як інформація про розмір вказує, скільки байтів займає даний запис. Ця секція критично важлива для забезпечення ефективності доступу до даних, особливо в системах з великими обсягами даних, де оптимізація часу доступу є ключовою (Рис. 2).

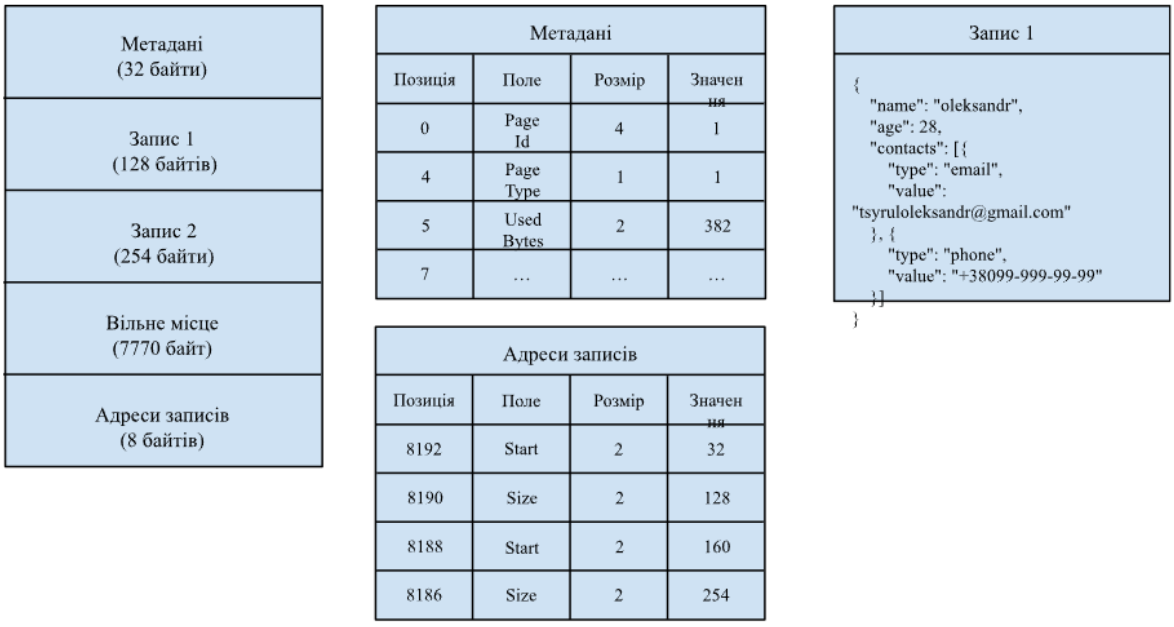

Рис. 2 Наочна візуалізація структури документа

Документ у контексті бази даних NoSQL на основі документів насамперед відноситься до структурованого, але гнучкого формату, який інкапсулює дані напівструктурованим способом. Зазвичай структуровані у форматі JSON (об’єктна нотація JavaScript), BSON (двійковий JSON) або подібних форматах, документи забезпечують динамічну схему для зберігання даних. На відміну від реляційних баз

даних, які вимагають, щоб дані відповідали попередньо визначеним моделям, кожен документ у базі даних NoSQL може мати свою унікальну структуру (Рис. 3).

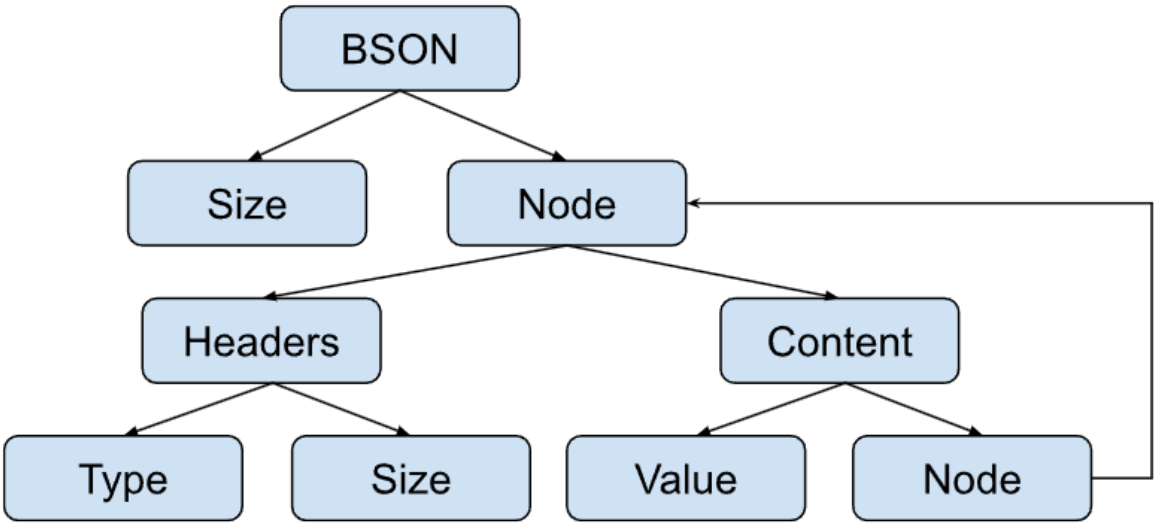

Рис. 3. Структура формату BSON

У базах даних документів дані інкапсулюються в парах ключ-значення, де «ключ» представляє назву атрибута, а «значення» — його дані. Цей формат дозволяє створювати вкладені документи та масиви, пропонуючи значну гнучкість і глибину.

Ця структура документа є інтуїтивно зрозумілою для розробників, оскільки вона часто відповідає структурі об'єктів в коді програми, зменшуючи невідповідність структурі, яка часто зустрічається в реляційних базах даних. Використання структури моделі документа має ряд переваг:

- Гнучкість: документи можуть часто змінювати структуру в міру розвитку вимог програми.
- Ефективність читання/запису: дані, пов'язані з однією сутністю (наприклад, користувачем або продуктом), зберігаються в одному документі, зменшуючи кількість читань, необхідних для отримання або оновлення сутності.

Колекція в документній базі даних NoSQL функціонує подібно до таблиці в системі реляційної бази даних, але без жорстких обмежень схеми. Кожна колекція призначена для зберігання масиву документів, потенційно кожен із власною унікальною структурою, адаптованою до даних, які він зберігає. Цей архітектурний вибір сприяє створенню надзвичайно гнучкого середовища, підтримуючи динамічні та часто непередбачувані потреби сучасних програм. Документи, що містяться в колекції, зазвичай пов'язані контекстом програми, що робить логічне розділення інтуїтивно зрозумілим. Наприклад, колекція «клієнти» може містити всі документи, пов'язані з окремими клієнтами. Для розподілу даних між колекціями в сторінках з даними в метадані зберігають індекс колекції до якої належить сторінка. Це допомагає в покращити час взаємодії з даними, оскільки при виконанні запиту, всі сутності сторінки, належать до одного набору даних. Але даний підхід призводить до незначного виділення додаткової пам'яті, шляхом резервування та збереження сторінок власних сторінок для кожної колекції (Рис. 4).

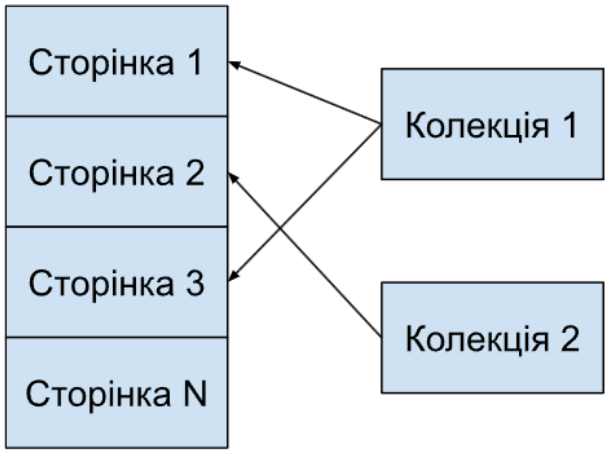

Рис. 4. Схематичне зображення взаємодії сторінок та колекцій

Управління колекціями включає кілька ключових операцій, спрямованих на підтримку точності, доступності та цілісності даних. Ці операції зазвичай включають створення, читання, оновлення та видалення документів, які зазвичай називають операціями CRUD.

- Створити (Create): додавання нових документів до колекції. Кожна операція може передбачати вставлення одного або кількох документів.
- Читання (Read): отримання документів із колекції. Цей процес може передбачати простий пошук за ідентифікатором документа або складні запити з використанням різних полів усередині документів.
- Оновлення (Update): зміна існуючих документів. Оновлення в NoSQL можуть бути вибірковими, тобто оновлюватимуться лише визначені поля, залишаючи решту документа без змін.
- Видалити (Delete): видалення документів із колекції за певними критеріями або повне видалення.

Кожну з цих операцій можна налаштувати та оптимізувати на основі базової структури документа, загальної моделі даних і міркувань продуктивності запиту.

Ефективне надсилання запитів має вирішальне значення для продуктивності бази даних. Бази даних NoSQL на основі документів використовують різні методи індексування для прискорення пошуку даних.

- Первинні та вторинні індекси: первинні індекси, як правило, обертаються навколо унікального ідентифікатора за замовчуванням у кожному документі, тоді як вторинні індекси можуть бути побудовані на будь-якому доступному для вибору полі в документах.
- Складені індекси та повнотекстовий пошук: можна використовувати складені індекси, щоб забезпечити ефективне надсилання запитів до кількох полів. Крім того, деякі системи NoSQL підтримують можливості повнотекстового пошуку, індексуючи цілі документи для операцій текстового пошуку.

МОДЕЛЮВАННЯ ВЗАЄМОДІЇ ДАНИХ В ЗАЛЕЖНОСТІ ВІД ОБРАНОЇ СТРУКТУРИ БАЗИ ДАНИХ

Ефективний збір даних і керування ними прямо пропорційно впливають на швидкодію взаємодії з даними. При виконанні запиту до бази даних, потрібно отримати

з файлової системи сторінку, перевірити чи відповідає тип сторінки за збереження даних, якщо так, перевірити чи відповідає колекції для пошуку. Для вирішення даної проблеми використовують первинні індекси не тільки для пошуку значень за ключем, а і для внутрішніх операцій, щоб отримати повний набір даних що належить колекції, наприклад при додаванні нового індексу. Для не реляційних баз даних і деяких реляційних також, самими популярними індексами є B-Tree та B-Tree+. Різниця між даними індексами є в збереженні значень. Якщо B-Tree зберігає значення в кожному елементі (Node) то B-Tree+ зберігає значення тільки в останньому елементі дерева (Leaf), що призводить до дублювання ключа, але зменшення розміру кожного елементу дерева, що можна представити моделлю (Рис. 5).

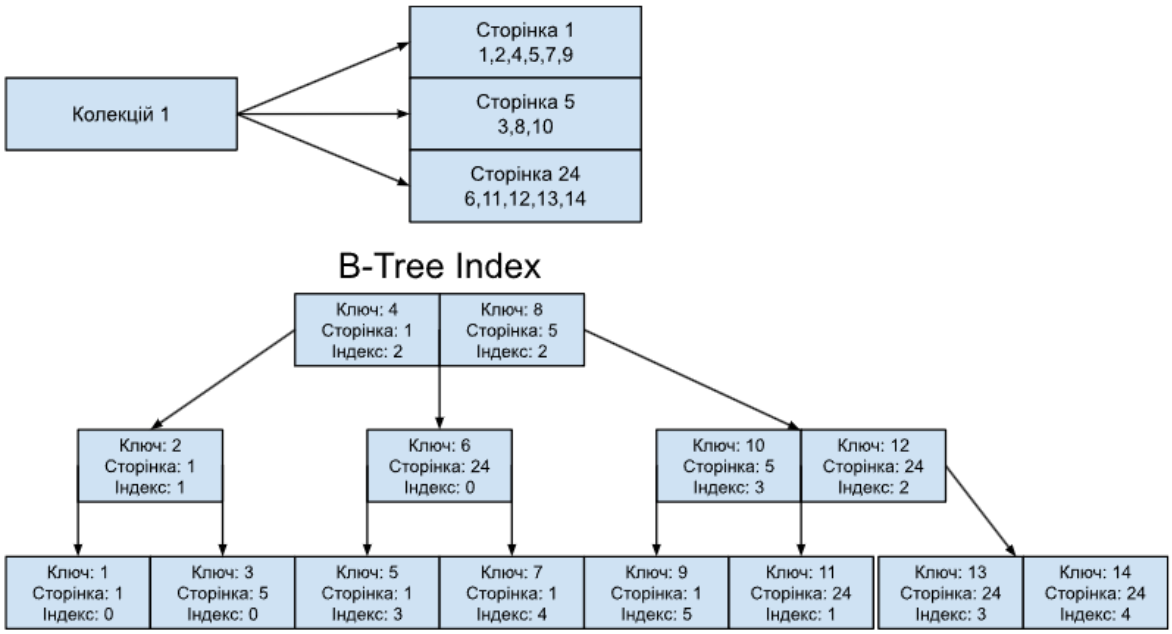

Рис. 5. Побудова B-Tree індексу на основі заданих сторінок

У сучасних системах NoSQL керування транзакціями [9], особливо в розподілених середовищах, передбачає розгляд узгодженості, атомарності та довговічності. Транзакції бази даних у системах NoSQL на основі документів, які є складними через їх розподілену природу, можуть включати механізми для забезпечення атомарних операцій між кількома документами або цілими колекціями.

Для створення надійної системи є різні підходи до відповідної надійності, але більшість з них суттєво впливають на швидкодію Бази даних NoSQL на основі документів часто дозволяють налаштування конфігурації, які схиляють терези на користь узгодженості або доступності, залежно від конкретних вимог програми. Коли швидкодія не є основною вимогою, а більш важливо відповідати максимальним вимогам надійності, для даної цілі користуються принципом ACID (атомарність, узгодженість, ізольованість, довговічність) для підтримки сценаріїв, що вимагають високої узгодженості та надійності. В свою чергу, коли швидкодія стоїть на першому місці, користуються теоремою CAP: для розподілених баз даних теорема CAP стверджує [10], що система не може одночасно гарантувати узгодженість, доступність і толерантність.

ОРГАНІЗАЦІЯ ДАНИХ І ПРОДУКТИВНІСТЬ

Організації даних у базі даних NoSQL значно впливає на її загальну продуктивність. Ефективна організація даних сприяє скороченню часу відповіді, зменшенню накладних витрат на зберігання та покращеній масштабованості. Основними аспектами продуктивності, на які впливає організація даних:

- Швидкість читання та запису: організація даних у межах оптимізованих блоків і використання методів індексування, значно зменшити кількість операцій введення та виведення на диску, необхідних для читання та запису даних, тим самим прискорюючи дані процеси.
- Ефективність запитів: належне індексування, продумана структура документа та структурування даних за колекціями, можуть зменшити навантаження під час виконання запиту. Це призводить до покращення часу відповіді на запит та зменшення навантаження на ресурси бази даних.
- Локальність даних: зберігання пов’язаних даних в одному блоці пам’яті чи сусідніх блоках може зменшити потребу в тривалому використанню диска, а також може прискорити операції з’єднання, що можуть бути особливо ресурсо затратними в розподілених базах даних.
- Кешування: ефективна організація даних сприяє покращенню ефективності систем кешування, більш точно узгоджуючи кешовані дані з шаблонами запитів, тим самим покращуючи час доступу.
- Вибіркове індексування: реалізація індексів лише для полів, які часто використовуються або ті які передбачають операції сортування. Збільшення кількості індексів може призвести до непотрібних накладних витрат на зберігання та продуктивність.
- Вибір типу індексу: залежно від запиту важливою складовою і використання відповідні типи індексу, наприклад складені індекси, геопросторові індекси або повнотекстові індекси, покращують продуктивність запитів.
- Денормалізація: на відміну від реляційних баз даних, NoSQL часто виграє від денормалізованих структур даних, які об’єднують пов’язані дані в окремі документи, зменшуючи потребу в дорогих операціях об’єднання документів.
- Шардинг: розподіл даних між декількома серверами для підвищення масштабованості та відмовостійкості. Ефективне сегментування залежить від вибору правильного ключа фрагмента, який максимізує розподіл даних, одночасно підтримуючи переважаючі шаблони запитів.
- Поділ: логічне поділ даних на одному сервері або між кластерами може зменшити час отримання даних і ефективніше керувати великими наборами даних.
- Кешування бази даних: використання кешування сторінок даних може значно пришвидшити взаємодію з даними шляхом меншого навантаження дискового сховища.
- Дефрагментація даних: регулярно дефрагментувати сховище даних, щоб забезпечити безперервне зберігання блоків, що може покращити операції читання.
- Фонові оновлення: оновлення в не пікові години, щоб мінімізувати вплив на продуктивність бази даних, гарантуючи, що важкі операції запису не впливатимуть на роботу інтерфейсу користувача.

Завдяки продуманій і стратегічній оптимізації організації даних бази даних NoSQL на основі документів можуть значно підвищити продуктивність. Ці оптимізації не тільки підвищують операційну ефективність, але й забезпечують ефективне масштабування бази даних із зростанням вимог до даних програми. Баланс між оптимізованими

методами зберігання даних і ефективними операціями з даними в режимі реального часу становить суть управління високопродуктивними системами баз даних NoSQL.

ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

В епоху великих даних, вибір відповідної системи управління базами даних (СУБД) стає життєво важливим для успішної реалізації обчислювальних задач. Документо-орієнтовані NoSQL бази даних забезпечують гнучкість та масштабованість, яка необхідна для сучасних додатків. Ця стаття оглядає три популярні документо-орієнтовані бази даних: MongoDB, RavenDB і LiteDB, оцінюючи їх архітектурні властивості, транзакційні можливості, запити та індексацію. Кожна з представлених баз даних має відкритий вихідний код, що надало можливість більш глибоко зануритись в механізми їх роботи.

- MongoDB - Широко відома своєю високою продуктивністю і гнучкістю, MongoDB використовує формат BSON (Binary JSON) для збереження даних, що оптимізує її для великих обсягів документів. Підтримка складних запитів, агрегації та індексації робить MongoDB ідеальним варіантом для масштабованих застосунків.
- RavenDB - Розроблена на платформі .NET, RavenDB використовує Linq і RQL (Raven Query Language) для запитів, що спрощує розробку в середовищі Windows. RavenDB підтримує сильні транзакційні гарантії з самого свого створення, що є відмінним для додатків, які вимагають надійної консистенції даних.
- LiteDB - Ця легка база даних ідеально підходить для використання у десктопних додатках, малих веб-додатках та утилітах, оскільки не вимагає великого розгортання і може бути легко вбудована в застосунки.

В розгляді легких рішень для систем управління базами даних, особлива увага приділяється LiteDB, оскільки вона вирізняється своєю доступністю та ефективністю для певних категорій застосувань. є вбудованою, легкою в застосуванні базою даних, що підтримує велику частину функцій без необхідності використання власної інфраструктури. Це дає їй значні переваги в динамічних умовах використання з обмеженими ресурсами, що робить її подібною до не менш популярного представника серед реляційних баз даних, а саме SQLite. Незважаючи на багатий функціонал, LiteDB має обмеження, що стають помітними при роботі з великими обсягами даних або при здійсненні складних операцій. В таких сценаріях LiteDB може виявитись менш ефективною, ніж більш спеціалізовані системи баз даних, що проектовані саме для великомасштабної роботи. У зв’язку з різними сферами використання, більш доречним є порівняння MongoDB та RavenDB.

Можливість масштабування, безперечно, є одним з найбільших переваг використання документно орієнтованих баз даних, а саме як база даних поводиться під час роботи на кількох серверах, що включає в себе здатність працювати та не втрачати дані, коли вузол у кластері виходить з ладу та розподілити навантаження на кілька машин. У даній сфері MongoDB є лідером, з високою пропускною здатністю і підтримкою великих кластерів даних. RavenDB також здатний до кластеризації, але підхід до її реалізації, є дуже відмінним, що може бути більш доцільним в залежності від задачі.

Концепція транзакцій в базах даних, що забезпечує атомарність, послідовність, ізоляцію та стійкість (ACID), є визначальною для багатьох систем управління базами даних. RavenDB був спроектований з підтримкою ACID-транзакцій з самого початку. Ця особливість робить RavenDB відмінним рішенням для розробників, яким потрібна

гарантія цілісності та надійності даних, відповідно до класичної транзакційної моделі. RavenDB використовує власну модель зберігання даних, яка оптимізована для забезпечення транзакційної сумісності на рівні документів. Тим часом, MongoDB, яка починалася як СУБД без підтримки традиційних ACID-транзакцій, розвивалася у бік їх інтеграції, що відбулося починаючи з версії 4.0. Це додавання було значною зміною для MongoDB, яка традиційно славилась своєю високою продуктивністю та горизонтальною масштабованістю в обмін на певні транзакційні гарантії. З часом команда розробників MongoDB досягла того, що стало можливим використання ACID-транзакцій для складних дій з даними на множині документів.

Мови запитів є ключовим інструментом взаємодії з базами даних. Вони дозволяють розробникам формулювати вимоги до даних та керувати ними ефективно. MongoDB використовує JSON-подібний формат для написання запитів, що має назву BSON (Binary JSON). Ця мова запитів заснована на структурі даних, що сприймається формально та вимагає від розробників розуміння структур JSON. RavenDB, в свою чергу, використовує запити RQL (Raven Query Language), яка є адаптацією SQL для документ-орієнтованих баз даних, доповнену здатністю використання LINQ (Language Integrated Query) - технології, що інтегрується безпосередньо у C# та інші мови .NET платформи. RQL та LINQ спільно забезпечують більш звичний досвід для розробників, які вже мають досвід роботи з SQL або .NET, знижуючи поріг входження та спрощуючи реалізацію запитів

MongoDB використовує стандартний та надійний набір індексів, для прикладу первинні, вторинні, на основі декількох полів, просторові, повнотекстові. в свою чергу RavenDB має принципово інший підхід до розробки індексів. Вони представлені та оголошені як визначені користувачем функції. Така функція дають можливість використання обчислення або операції над індексами, перетворення даних або їх агрегація під час індексації.

Вибір між MongoDB, RavenDB і LiteDB залежить від конкретних потреб проекту, включаючи масштабність, треби до транзакційності, а також ресурсів і середовища розробки. MongoDB є найбільш універсальним варіантом для великих, масштабованих застосунків, RavenDB оптимізована для серйозних .NET додатків, а LiteDB відмінно підходить для невеликих застосунків або як вбудована база даних.

ВИСНОВОК ТА ПЕРСПЕКТИВИ ПОДАЛЬШИХ ДОЛІДЖЕНЬ

Правильне розуміння архітектури організації даних у документних базах даних NoSQL має першорядне значення для використання їх повного потенціалу та ефективного керування сучасними додатками для великих даних. Протягом цього обговорення ми глибоко заглибилися в основні компоненти, які складають основу баз даних NoSQL: біти, блоки, документи, сторінки та колекції. Кожен компонент відіграє ключову роль у ефективності та масштабованості операцій з даними, безпосередньо впливаючи на загальну продуктивність системи.

Динамічний характер баз даних NoSQL, особливо тих, які орієнтовані на документи, пропонує значну перевагу в обробці різноманітних типів даних і моделей даних, що швидко змінюються. Гнучкість документів у поєднанні з надійною обробкою колекцій дозволяє компаніям розвивати свої програми без суворих обмежень, які накладають традиційні реляційні бази даних.

Оскільки дані продовжують зростати в обсязі, різноманітності та швидкості, роль баз даних ставатиме все більш критичною в ландшафті управління даними. Постійні дослідження та розробки в цій галузі, ймовірно, запропонують передові методи обробки

даних, підтримки транзакцій та оптимізації продуктивності, допомагаючи організаціям залишатися гнучкими в умовах зміни потреб у даних.

Підсумовуючи, опанувавши архітектуру організації даних у базах даних, зацікавлені сторони можуть не лише забезпечити ефективне керування даними, але й створювати масштабовані, високопродуктивні програми, які використовують увесь спектр можливостей, пропонованих сучасною технологією NoSQL.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Atzeni, P., Bugiotti, F., Cabibbo, L., & Torlone, R. (2020). Data modeling in the NoSQL world. *Computer Standards & Interfaces*, *67*. https://doi.org/10.1016/j.csi.2016.10.003
2. Liu, L. (2023). RETRACTED ARTICLE: Design of NoSQL database in oral English teaching based on 5G network and AI recognition. *Soft Computing : A Fusion of Foundations, Methodologies and Applications*, *27*(14), 10337–10345. https://doi.org/10.1007/s00500-023-08306-6
3. Gomes, C., de O. Junior, M. N., Nogueira, B., Maciel, P., & Tavares, E. (2023). NoSQL-based storage systems: influence of consistency on performance, availability and energy consumption. *The Journal of Supercomputing : An International Journal of High-Performance Computer Design, Analysis, and Use*, *79*(18), 21424–21448. https://doi.org/10.1007/s11227-023-05488-6
4. Gomes, C., Tavares, E., Junior, M. N. d. O., & Nogueira, B. (2021). Cloud storage availability and performance assessment: a study based on NoSQL DBMS. *The Journal of Supercomputing : An International Journal of High-Performance Computer Design, Analysis, and Use*, *78*(2), 2819–2839. https://doi.org/10.1007/s11227-021-03976-1
5. Maté, A., Peral, J., Trujillo, J., Blanco, C., García-Saiz, D., & Fernández-Medina, E. (2021). Improving security in NoSQL document databases through model-driven modernization. *Knowledge and Information Systems : An International Journal*, *63*(8), 2209–2230. https://doi.org/10.1007/s10115-021-01589-x
6. Mozaffari, M., Nazemi, E., & Eftekhari-Moghadam, A. M. (2020). Feedback control loop design for workload change detection in self-tuning NoSQL wide column stores. *Expert Systems with Applications*, *142*. https://doi.org/10.1016/j.eswa.2019.112973
7. da Silva, L. F., & Lima, J. V. F. (2023). An evaluation of relational and NoSQL distributed databases on a low-power cluster. *The Journal of Supercomputing : An International Journal of High-Performance Computer Design, Analysis, and Use*, *79*(12), 13402–13420. https://doi.org/10.1007/s11227-023-05166-7
8. Abramov, V., Astafieva, M., Boiko, M., Bodnenko, D., Bushma, A., Vember, V., Hlushak, O., Zhyltsov, O., Ilich, L., Kobets, N., Kovaliuk, T., Kuchakovska, H., Lytvyn, O., Lytvyn, P., Mashkina, I., Morze, N., Nosenko, T., Proshkin, V., Radchenko, S., & Yaskevych, V. (2021). Theoretical and practical aspects of the use of mathematical methods and information technology in education and science. https://doi.org/10.28925/9720213284km
9. Blanco, C., García-Saiz, D., Rosado, D. G., Santos-Olmo, A., Peral, J., Maté, A., Trujillo, J., & Fernández-Medina, E. (2022). Security policies by design in NoSQL document databases. *Journal of Information Security and Applications*, *65*. https://doi.org/10.1016/j.jisa.2022.103120
10. Brewer E. (2012). CAP twelve years later: How the 'rules' have changed. *Computer*, Volume 45, Issue 2, pg. 23–29. DOI:10.1109/MC.2012.37.

КІБЕРБЕЗПЕКА: освіта, наука, техніка
CYBERSECURITY: EDUCATION, SCIENCE, TECHNIQUE
№ 2 (30), 2025
Спеціальний випуск ISSN 2663 – 4023
Special issue

Oleksandr Tsyrul
postgraduate
Institute of Telecommunications and Global Information Space of the NAS of Ukraine, Kyiv, Ukraine
ORCID :0009-0002-5945-5918
tsyruloleksandr@gmail.com

STRUCTURE OF HIGH-PERFORMANCE NOSQL DATA MANAGEMENT SYSTEMS

Abstract. The article explores the architectural foundations of data organization in high-performance NoSQL database management systems, particularly document-oriented ones. It examines the data structure at all levels—from bits and blocks to pages, documents, and collections—forming a unified model of data control. Document-based databases (MongoDB, RavenDB, LiteDB) are shown to provide flexibility, scalability, and high performance through semi-structured storage formats (JSON, BSON), schema-free design, and CRUD operation support. The paper analyzes how data organization, indexing, caching, denormalization, and sharding affect system efficiency. Special attention is paid to index structures such as B-Tree and B-Tree+, transactional mechanisms aligned with ACID principles, and the application of the CAP theorem in distributed environments. A comparative analysis of MongoDB, RavenDB, and LiteDB architectures is presented regarding transaction processing, indexing, scalability, and query handling. The study concludes that a well-designed data organization architecture in NoSQL systems is a key factor ensuring flexibility, performance, reliability, and resilience in modern cloud and distributed data storage infrastructures.

Keywords: NoSQL; database; document; collection; page; block; bit; index; transaction; performance optimization.

REFERENCES (TRANSLATED AND TRANSLITERATED)

1. Atzeni, P., Bugiotti, F., Cabibbo, L., & Torlone, R. (2020). Data modeling in the NoSQL world. *Computer Standards & Interfaces*, *67*. https://doi.org/10.1016/j.csi.2016.10.003
2. Liu, L. (2023). RETRACTED ARTICLE: Design of NoSQL database in oral English teaching based on 5G network and AI recognition. *Soft Computing : A Fusion of Foundations, Methodologies and Applications*, *27*(14), 10337–10345. https://doi.org/10.1007/s00500-023-08306-6
3. Gomes, C., de O. Junior, M. N., Nogueira, B., Maciel, P., & Tavares, E. (2023). NoSQL-based storage systems: influence of consistency on performance, availability and energy consumption. *The Journal of Supercomputing : An International Journal of High-Performance Computer Design, Analysis, and Use*, *79*(18), 21424–21448. https://doi.org/10.1007/s11227-023-05488-6
4. Gomes, C., Tavares, E., Junior, M. N. d. O., & Nogueira, B. (2021). Cloud storage availability and performance assessment: a study based on NoSQL DBMS. *The Journal of Supercomputing : An International Journal of High-Performance Computer Design, Analysis, and Use*, *78*(2), 2819–2839. https://doi.org/10.1007/s11227-021-03976-1
5. Maté, A., Peral, J., Trujillo, J., Blanco, C., García-Saiz, D., & Fernández-Medina, E. (2021). Improving security in NoSQL document databases through model-driven modernization. *Knowledge and Information Systems : An International Journal*, *63*(8), 2209–2230. https://doi.org/10.1007/s10115-021-01589-x
6. Mozaffari, M., Nazemi, E., & Eftekhari-Moghadam, A. M. (2020). Feedback control loop design for workload change detection in self-tuning NoSQL wide column stores. *Expert Systems with Applications*, *142*. https://doi.org/10.1016/j.eswa.2019.112973
7. da Silva, L. F., & Lima, J. V. F. (2023). An evaluation of relational and NoSQL distributed databases on a low-power cluster. *The Journal of Supercomputing : An International Journal of High-Performance Computer Design, Analysis, and Use*, *79*(12), 13402–13420. https://doi.org/10.1007/s11227-023-05166-7
8. Abramov, V., Astafieva, M., Boiko, M., Bodnenko, D., Bushma, A., Vember, V., Hlushak, O., Zhyltsov, O., Ilich, L., Kobets, N., Kovaliuk, T., Kuchakovska, H., Lytvyn, O., Lytvyn, P., Mashkina, I., Morze, N., Nosenko, T., Proshkin, V., Radchenko, S., & Yaskevych, V. (2021). Theoretical and practical aspects of

the use of mathematical methods and information technology in education and science. https://doi.org/10.28925/9720213284km

9. Blanco, C., García-Saiz, D., Rosado, D. G., Santos-Olmo, A., Peral, J., Maté, A., Trujillo, J., & Fernández-Medina, E. (2022). Security policies by design in NoSQL document databases. *Journal of Information Security and Applications*, *65*. https://doi.org/10.1016/j.jisa.2022.103120
10. Brewer E. (2012). CAP twelve years later: How the 'rules' have changed. *Computer*, Volume 45, Issue 2, pg. 23–29. DOI:10.1109/MC.2012.37.