№ 10(51) 2025
НАУКА і ТЕХНІКА СЬОГОДНІ
серії: право, економіка, педагогіка, техніка, фізико-математичні науки

УДК 004:519.1

https://doi.org/10.52058/2786-6025-2025-10(51)-2122-2134

Шевцова Наталія Вікторівна кандидат технічних наук, доцент кафедри інформаційних технологій та моделювання Рівненського державного гуманітарного університету, м. Рівне, https://orcid.org/0000-0002-3401-5468

Бабич Степанія Михайлівна кандидат технічних наук, доцент, доцент кафедри інформаційних технологій та моделювання Рівненського державного гуманітарного університету, м. Рівне, https://orcid.org/0000-0003-2145-6392

АЛГОРИТМИ ДИСКРЕТНОЇ МАТЕМАТИКИ В СУЧАСНИХ ІНФОРМАЦІЙНИХ СИСТЕМАХ

Анотація. У сучасному цифровому середовищі, що характеризується стрімким зростанням обсягів неструктурованих та напівструктурованих даних, особливої актуальності набуває дослідження математичних основ інформаційних систем. Ця стаття присвячена комплексному аналізу алгоритмів дискретної математики, які становлять теоретичний фундамент сучасних розподілених систем керування даними, зокрема NoSQL баз даних. У роботі детально досліджено ключові математичні моделі та алгоритмічні рішення, що забезпечують високу ефективність функціонування сучасних інформаційних систем.

Особливу увагу приділено консистентному хешуванню як основному механізму розподілу даних у системах на кшталт Amazon DynamoDB та Apache Cassandra, який забезпечує детерміноване відображення ключів партицій на фізичні вузли кластера.

Дослідження охоплює аналіз архітектурних особливостей різних типів NoSQL систем, включаючи системи «ключ-значення», документні, стовпчикові та графові бази даних, з акцентом на їхні унікальні властивості та сфери застосування.

У статті систематизовано зв'язок між теоретичними концепціями дискретної математики та практичними реалізаціями в сучасних хмарних технологіях. Доведено, що кожен клас NoSQL систем демонструє оптимальну ефективність для певних типів робочих навантажень: системи «ключ-значення» досягають найвищої продуктивності для операцій точкового доступу, стовпчикові системи ефективно обробляють аналітичні запити, а графові бази даних є незамінними для аналізу складних мережевих взаємозв'язків.

Особливістю дослідження є комплексний підхід до аналізу архітектурних переваг NoSQL рішень, включаючи здатність до горизонтального масштабування, високу доступність, відмовостійкість та гнучкість моделі даних.

Встановлено чіткий зв'язок між теоретичними аспектами дискретної математики та їх практичними реалізаціями в сучасних хмарних технологіях, що дозволяє оптимізувати вибір технологічних рішень для конкретних бізнес-завдань. Результати дослідження мають значну практичну цінність для розробників та архітекторів інформаційних систем, що працюють із великими обсягами даних у реальному часі.

Ключові слова: дискретна математика, NoSQL бази даних, розподілені інформаційні системи, хмарні технології, консистентне хешування.

Shevtsova Nataliia Viktorivna PhD, Associate Professor of the Department of Information Technologies and Modeling, Rivne State University of the Humanities, Rivne, https://orcid.org/0000-0002-3401-5468

Babych Stepaniia Mykhailivna PhD, Associate Professor, Associate Professor of the Department of Information Technologies and Modeling, Rivne State University of the Humanities, Rivne, https://orcid.org/0000-0003-2145-6392

ALGORITHMS OF DISCRETE MATHEMATICS IN MODERN INFORMATION SYSTEMS

Abstract. In the modern digital environment, characterized by the rapid growth of unstructured and semi-structured data volumes, the study of mathematical foundations of information systems becomes particularly relevant. This article is dedicated to a comprehensive analysis of discrete mathematics algorithms that form the theoretical foundation of modern distributed data management systems, particularly NoSQL databases. The work provides a detailed investigation of key mathematical models and algorithmic solutions that ensure the high efficiency of modern information systems.

Special attention is paid to consistent hashing as the primary data distribution mechanism in systems like Amazon DynamoDB and Apache Cassandra, which ensures deterministic mapping of partition keys to physical cluster nodes. The research covers the analysis of architectural features of various types of NoSQL systems, including key-value, document, columnar, and graph databases, with emphasis on their unique properties and application areas.

The article systematizes the connection between theoretical concepts of discrete mathematics and their practical implementations in modern cloud technologies. It is proven that each class of NoSQL systems demonstrates optimal

efficiency for specific types of workloads: key-value systems achieve the highest performance for point access operations, columnar systems effectively handle analytical queries, and graph databases are indispensable for analyzing complex network relationships.

The research features a comprehensive approach to analyzing the architectural advantages of NoSQL solutions, including horizontal scalability, high availability, fault tolerance, and data model flexibility. A clear connection has been established between theoretical aspects of discrete mathematics and their practical implementations in modern cloud technologies, enabling optimization of technological solutions for specific business tasks. The research results have significant practical value for developers and architects of information systems working with large volumes of real-time data.

Keywords: discrete mathematics, NoSQL databases, distributed information systems, cloud technologies, consistent hashing.

Постановка проблеми. Сучасні інформаційні системи стикаються з необхідністю обробки зростаючих обсягів неструктурованих та напів-структурованих даних, що призвело до активного впровадження NoSQL (Not Only SQL) баз даних. На противагу традиційним реляційним системам керування базами даних, побудованим на принципах жорсткої схеми та ACID-транзакцій, NoSQL-рішення демонструють переконливу ефективність у роботі з сучасними типами навантажень.

Фундаментальну основу функціонування NoSQL систем складають алгоритми дискретної математики, що забезпечують не лише ефективність виконання операцій та оптимальну організацію структур даних, але й підтримку консистентності в розподілених умовах. Ці алгоритми є ключовим компонентом, що забезпечує гнучкість моделювання даних, стійкість до збоїв та здатність до горизонтального масштабування.

Значення дискретної математики та її алгоритмічних реалізацій у розвитку NoSQL-систем продовжує зростати, що зумовлює необхідність поглибленого вивчення цих математичних дисциплін для фахівців у галузі інформаційних технологій. У міру збільшення потреб у обробці великих обсягів даних, саме цей математичний фундамент забезпечує стабільність, безпеку та надійність сучасних розподілених систем.

Аналіз останніх досліджень і публікацій. Сучасні дослідження в галузі розподілених систем керування даними демонструють посилену увагу до математичного фундаменту архітектурних рішень. Згідно з роботами Dong H. та ін. (2024) [1], cloud-native бази даних все більше покладаються на алгоритми дискретної математики для забезпечення горизонтального масштабування та відмовостійкості. Дослідження Anuyah S. ті ін. (2024) [2] підкреслюють

важливість графових алгоритмів для обробки складних зв'язків у соціальних мережах та рекомендаційних системах.

Публікації Kotiranta P. та ін. (2022) [3] показують переваги графових баз даних у виконанні складних запитів порівняно з традиційними реляційними системами. Огляди технічної документації AWS, Google Cloud та Azure [4, 5, 6] свідчать про активне впровадження консистентного хешування та інших алгоритмів дискретної математики в сучасних хмарних сервісах.

Особливу цінність представляють дослідження Wilson E.R. (2018) [7], які демонструють практичне застосування різних підходів до зберігання даних у сучасних додатках. Аналіз останніх публікацій свідчить про формування нових напрямів досліджень, пов'язаних з розвитком гібридних систем та вдосконаленням алгоритмів автономного управління розподіленими системами.

Мета статті: Дослідити роль алгоритмів дискретної математики в архітектурі сучасних інформаційних систем та проаналізувати їх практичну реалізацію в розподілених системах керування даними, зокрема в NoSQL базах даних.

Виклад основного матеріалу. Впровадження NoSQL баз даних як фундаменту інформаційних систем надає низку ключових архітектурних переваг. Перш за все, найважливішою перевагою є здатність до горизонтального масштабування, також відомого як масштабування «вшир». На противагу вертикальному масштабуванню, яке передбачає нарощування потужності окремого сервера, горизонтальне масштабування реалізується шляхом додавання стандартних серверних вузлів до кластера. Такі системи, як Apache Cassandra або Amazon DynamoDB, автоматично розподіляють дані та навантаження по цих вузлах, використовуючи складні алгоритми, серед яких ключову роль відіграє консистентне хешування. Цей підхід забезпечує практично необмежену масштабованість, дозволяючи системі обробляти пікові навантаження мільйонів запитів, що є критичним для глобальних соціальних мереж, платформ електронної комерції та IoT-рішень.

Крім того, архітектура NoSQL баз даних орієнтована на забезпечення високої доступності та відмовостійкості. Це досягається за рахунок вбудованих механізмів реплікації даних, коли кожен фрагмент інформації автоматично копіюється на кілька фізично розділених вузлів. У разі виходу з ладу одного або навіть кількох серверів, система продовжує обслуговувати запити, використовуючи доступні репліки, що мінімізує час простою та гарантує цілісність сервісу [1].

Важливим аспектом є гнучкість моделі даних, що реалізується через парадигму «схема при читанні». На відміну від реляційних баз, де структура таблиць має бути визначена строго до початку роботи, документні бази даних, такі як MongoDB, дозволяють зберігати напівструктуровані дані у формі JSON-

подібних документів без попередньої схеми. Це надає розробникам змогу швидко ітераціювати та модифікувати модель даних без проведення складних і ризикованих міграцій, пришвидшуючи темпи розробки в умовах динамічних вимог.

Також слід зазначити спеціалізацію NoSQL-рішень під конкретні типи навантажень і моделей даних. Наприклад, графові бази даних, такі як Neo4j, оптимізовані для ефективного виконання запитів, пов'язаних з аналізом зв'язків та обходом графів, що робить їх незамінними для систем виявлення шахрайства або соціальних мереж.

Таким чином, інтеграція NoSQL баз даних в архітектуру сучасних інформаційних систем є не просто трендом, а об'єктивною необхідністю, зумовленою потребою в обробці масивів даних нового типу. Їхні переваги — горизонтальна масштабованість, висока доступність, гнучка модель даних та предметна оптимізація — роблять їх фундаментальним компонентом для побудови високопродуктивних, відмовостійких та легко адаптованих інформаційних середовищ.

Класифікація NoSQL систем на основі моделей представлення даних та їх кротка характеристика [7]:

- *Системи типу «ключ-значення»* характеризуються організацією даних у вигляді асоціативних масивів, де кожному ключу відповідає певне значення. Архітектура цих систем ґрунтується на розподілених хеш-таблицях, що забезпечує детермінований доступ до даних. Серед представників цього класу виділяються Redis, Amazon DynamoDB та Riak. Основним перевагою є висока швидкодія при виконанні простих операцій читання та запису, що досягається завдяки мінімальній складності моделі даних. Обмеження пов'язані з відсутністю механізмів для виконання складних запитів та аналітичних операцій. Типові сценарії використання включають системи кешування, зберігання сесійних даних та параметрів конфігурації. Архітектура систем керування даними типу «ключ-значення» ґрунтується на фундаментальних принципах теорії хешування та методах комбінаторного аналізу.
- *Документо-орієнтовані системи* базуються на концепції документа як самодостатньої одиниці даних у форматі JSON, BSON або XML. Такі системи як MongoDB та Couchbase забезпечують гнучку схемну організацію, що дозволяє адаптувати структуру даних до мінливих вимог. Переваги включають ефективну роботу з ієрархічними даними та широкі можливості індексації. Недоліки проявляються при необхідності виконання складних міждокументних операцій, що пов'язано з відсутністю підтримки JOIN-механізмів у класичному розумінні. Застосовуються переважно для систем управління контентом, електронних каталогів та профілів користувачів. Математичний апарат цих систем включає теорію дерев та складних алгоритмів індексації. Основу

складають збалансовані дерева, зокрема B-дерева та їх модифікації, що забезпечують логарифмічну складність операцій пошуку та вставки. Для обробки складних вкладених структур використовуються рекурсивні алгоритми обходу дерев.

- *Стовпчикові системи* відрізняються організацією даних у вигляді стовпців, а не рядків, що дозволяє ефективно виконувати операції агрегації та аналітичні запити. Представниками цього класу є Apache Cassandra, HBase та ScyllaDB. Архітектура цих систем оптимізована для операцій масового запису та читання великих обсягів даних. Алгоритми управління розподіленими даними та реплікації в цих системах базуються на принципах теорії ймовірностей та методах оцінки розподілу даних. Переваги включають високий рівень стиснення даних та ефективне використання ресурсів системи зберігання. Проєктування моделі даних вимагає ретельного аналізу шаблонів доступу, що становить певну складність для розробників. Типові сценарії використання охоплюють системи аналітики реального часу, платформи збору телеметричних даних та рішення для IoT (Internet of Things).
- *Графові системи* спеціалізуються на обробці даних зі складними мережевими взаємозв'язками та реалізують формальну модель теорії графів як концептуальну основу своєї архітектури. Математичний базис цих систем включає розвинений апарат алгоритмів пошуку шляхів, серед яких центральне місце посідають алгоритм Дейкстри для знаходження найкоротших шляхів у зважених графах та алгоритм Флойда-Уоршелла для обчислення найкоротших відстаней між усіма парами вершин, методи обходу графових структур, такі як пошук в ширину (BFS) та пошук в глибину (DFS), утворюють фундамент для реалізації основних операцій навігації по графах. Теоретико-графова основа цих систем поширюється на теорію мережевих потоків, яка забезпечує математичний апарат для аналізу оптимізаційних задач у транспортних мережах, системах розподілу ресурсів та комунікаційних мережах. Серед представників виділяються Neo4j, Amazon Neptune та JanusGraph. Головною перевагою є ефективна реалізація алгоритмів пошуку шляхів, що робить ці системи незамінними для аналізу соціальних мереж, рекомендаційних систем та виявлення аномалій. Специфічність моделі даних та складність розподіленого масштабування становлять основні обмеження для широкого впровадження.

Дослідження показують, що кожен клас NoSQL систем демонструє оптимальну ефективність для певних типів робочих навантажень. Системи «ключ-значення» досягають найвищої продуктивності для операцій точкового доступу, тоді як стовпчикові системи ефективніше обробляють аналітичні запити. Документо-орієнтовані системи займають проміжне положення, поєднуючи гнучкість структури з достатньою продуктивністю [2].

Сучасні інформаційні системи все частіше базуються на поєднанні розподілених архітектур і хмарних технологій, що дозволяє одночасно вирішувати кілька критичних завдань. З одного боку, розподілена архітектура забезпечує високу доступність і стійкість системи шляхом географічного розподілу обчислювальних ресурсів, що зменшує мережеві затримки та захищає від регіональних збоїв. З іншого боку, інтеграція з хмарними технологіями надає гнучке масштабування, дозволяючи динамічно адаптувати потужність системи до мінливого навантаження без необхідності значних капітальних інвестицій.

Така конвергенція технологій створює синергетичний ефект, де розподілена архітектура забезпечує базову надійність і продуктивність, а хмарні компоненти додають еластичність і економічну ефективність. Ця комбінація особливо важлива в умовах сучасного цифрового бізнес-середо-вища, де системи повинні одночасно забезпечувати високу доступність, оброб-ляти великі обсяги даних і залишатися економічно ефективними при мінливому навантаженні.

Архітектура паралельної обробки запитів у системах на кшталт ScyllaDB та Apache Cassandra ґрунтується на алгоритмах балансування навантаження, які використовують принципи теорії графів та комбінаторної оптимізації для динамічного розподілу обчислювального навантаження. Ці алгоритми врахо-вують топологію мережі, географічне розташування даних та поточне наванта-ження вузлів, забезпечуючи мінімізацію затримок та максимізацію пропускної здатності в хмарних середовищах Amazon Web Servises, Google Cloud та Azure.

Важливим компонентом розподілених інформаційних систем є прото-коли консенсусу Paxos та Raft, являють собою математично строгі рішення для забезпечення узгодженості в асинхронних розподілених системах. Алгоритм Paxos ґрунтується на теорії виборів у розподілених системах та формальній логіці.

Протокол Raft, створений як більш зрозуміла альтернатива Paxos, використовує формальну модель детермінованих автоматів. У сучасних хмарних сервісах ці протоколи реалізуються з урахуванням специфіки мережевої інфраструктури. Google Cloud Spanner використовує модифіковану версію Paxos з глобальною синхронізацією часу через атомні годинники, що дозволяє досягати глобальної консистентності при збереженні доступності. Azure Cosmos DB реалізує мультирегіональну архітектуру на основі власного протоколу консенсусу, що поєднує елементи Paxos з механізмами кворумів для оптимізації затримок між регіонами [7, 8].

Географічне розподілення даних реалізується через алгоритми розміщення даних (Data Placement Algorithms) та вибору регіонів (Region Selection Algorithms), що базуються на теорії графів та методах оптимізації,

враховуючи шаблони доступу, правові вимоги та економічні фактори. Ці алгоритми призначені для автоматичного визначення оптимального розташування даних з урахуванням вартості передачі, мережевих затримок та вимог захисту даних.

Сучасні розподілені системи керування даними отримують свою ефективність із строгих формальних математичних моделей, що забезпечують передбачувану продуктивність та надійність у складних умовах експлуатації. Глибоке дослідження алгоритмічних інструментів NoSQL систем виявляє їхню органічну зв'язність із фундаментальними розділами дискретної математики, що становить теоретичну основу для їх функціонування.

Консистентне хешування, як центральний механізм розподілу даних, ґрунтується на принципах теорії кілець та апараті модульної арифметики. Цей підхід знайшов широке застосування в таких системах як Amazon DynamoDB, Cassandra та Riak, де простір хеш-ключів та простір серверних вузлів взаємно відображаються на дискретне кільце через систему детермінованих хеш-функцій. Дискретна природа цього підходу проявляється у використанні скінченної множини точок на кільці, що контрастує з неперервними інтервалами традиційних підходів і забезпечує мінімальний перерозподіл даних при динамічних змінах конфігурації кластера [8].

Деревні структури індексів, що використовуються в сучасних системах, реалізують принципи теорії графів, зокрема концепцію спрямованих ациклічних графів. B-дерева та B+-дерева як збалансовані ієрархічні структури зберігають суворі інваріантні властивості в реляційних базах даних таких як PostgreSQL та MySQL, а також в документних базах MongoDB. Підтримка цих інваріантів забезпечується через систему дискретних операцій розбиття та об'єднання вузлів, а також алгоритмів перерозподілу ключів. Аналіз складності основних операцій, який демонструє логарифмічну залежність від кількості елементів, ґрунтується на формальних методах дискретної математики та комбінаторного аналізу.

Архітектура LSM-дерев реалізується на послідовності дискретних подій, що включають запис у журнал у пам'яті, створення відсортованих файлів та операції злиття. Ця модель реалізована в таких системах як RocksDB та Apache Cassandra. Алгоритм злиття відсортованих послідовностей представляє собою канонічну задачу дискретної математики, яка знайшла ефективне застосування в сучасних системах зберігання даних. Застосування амортизаційного аналізу дозволяє математично довести, що середня вартість операцій запису зберігається на оптимальному рівні, незважаючи на значні витрати окремих операцій злиття.

Ймовірнісні структури даних, такі як фільтри Блума, реалізують принципи теорії ймовірностей для дискретних подій у поєднанні з апаратом

теорії множин. Бітовий масив як елементарна дискретна структура даних стає основою для побудови ефективних механізмів фільтрації в базах даних HBase та Redis. Математична модель функціонування цих структур ґрунтується на аналізі незалежних подій обчислення хеш-функцій, причому ймовірність хибнопозитивних результатів описується комбінаторними формулами, що створює класичну задачу багатокритеріальної оптимізації з компромісом між обсягом пам'яті, точністю та обчислювальною складністю.

Конфліктно-вільні репліковані типи даних (CRDT) ґрунтуються на формалізмах теорії частково впорядкованих множин та абстрактної алгебри. Ці структури знайшли застосування в розподілених системах таких як Redis, Riak та Akka. Математичні властивості комутативності та асоціативності операцій забезпечують можливість безконфліктного злиття реплік у розподілених умовах. Векторні годинники як складні дискретні структури даних реалізують механізм відстеження причинно-наслідкових зв'язків у розподілених системах, що є критично важливим для забезпечення цілісності даних.

Графові бази даних, такі як Neo4j, Amazon Neptune та JanusGraph, використовують потужний формальний апарат теорії графів, де граф визначається як дискретна структура з скінченними множинами вершин та ребер. Алгоритми пошуку в ширину та глибину оперують дискретними поняттями сусідств вершин, відстаней між ними та списків суміжності. Коректність цих алгоритмів доводиться строгими методами теорії графів, що підтверджує їх властивості знаходження найкоротших шляхів та інші важливі характеристики [3].

Для прикладу детальніше розглянемо алгоритм консистентного хешування, який використовується в хмарній системі керування даними NoSQL типу «ключ-значення» Amazon DynamoDB. Він становить основу її архітектури та забезпечує високу масштабованість і відмовостійкість [4, 8].

Кожна таблиця DynamoDB використовує первинний ключ, який може бути представлений у двох формах: простий ключ, що складається з одного атрибута, або складений ключ (Composite Key), який включає ключ партиції (Partition Key) та ключ сортування (Sort Key).

Саме значення ключа партиції визначає фізичне розташування даних у кластері. Система застосовує спеціалізовану внутрішню хеш-функцію до значення ключа партиції. Ця функція, розроблена компанією Amazon, перетворює вхідне значення на велике ціле число (зазвичай 128-бітове), що забезпечує рівномірний розподіл даних та підвищену безпеку системи. DynamoDB використовує модифіковану версію консистентного хешування, що включає наступні компоненти:

Адресний простір: Весь діапазон хеш-значень ($0..2^{128}-1$) представляється у вигляді логічного кільця.

Віртуальні вузли: Кожен фізичний сервер відображається на кільце через множину віртуальних вузлів, що забезпечує більш рівномірний розподіл навантаження порівняно з базовою реалізацією консистентного хешування.

Процес локалізації даних: Для визначення цільового сервера система виконує послідовність операцій (схема розподілу зображена на рисунку1):

1. хеш-значення ключа партиції розміщується на віртуальному кільці;
2. виконується пошук першого віртуального вузла сервера за годинниковою стрілкою;
3. дані направляються на фізичний сервер, пов'язаний із знайденим віртуальним вузлом.

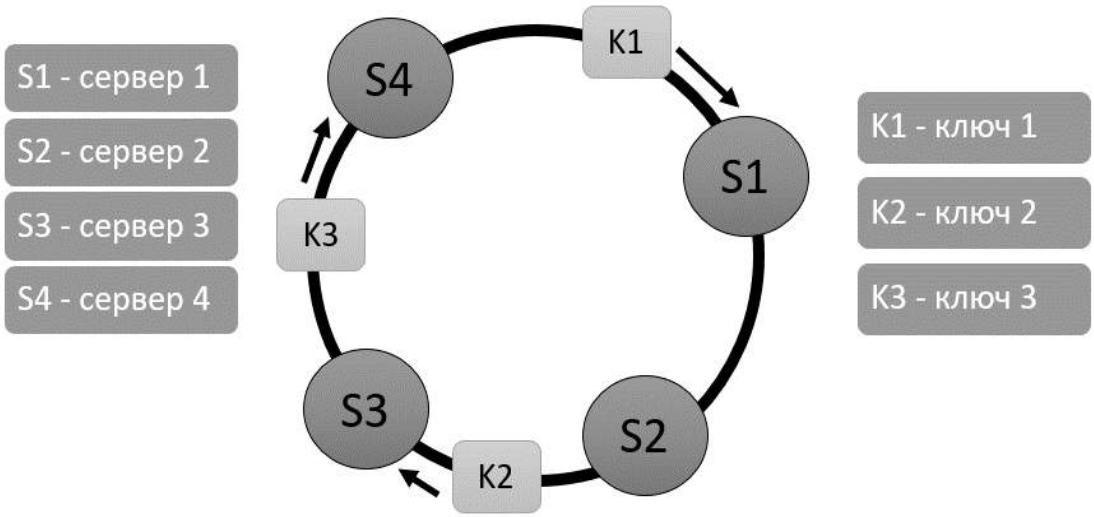

Рис. 1. Схема розподілу даних на віртуальному кільці.

Дана архітектура забезпечує адаптивне масштабування, при якому додавання або видалення серверів кластера призводить до мінімального перерозподілу даних, коли лише невелика частина записів, що знаходяться між зміненими вузлами, потребує переміщення, забезпечуючи практично безперервну роботу системи під час масштабування.

Ефективне балансування навантаження досягається завдяки комбінації детермінованої хеш-функції та віртуальних вузлів, що забезпечує рівномірний розподіл даних по всіх серверах кластера. Відмовостійкість системи реалізується через автоматичну реплікацію даних на кілька послідовних вузлів кільця, де типовий коефіцієнт реплікації становить K=3. Цей механізм гарантує доступність даних навіть при одночасному збої кількох серверів, оскільки система здатна автоматично перенаправляти запити до доступних реплік, зберігаючи цілісність та консистентність даних у разі часткових відмов обладнання.

Архітектура DynamoDB демонструє високу продуктивність операцій читання/запису та простих запитів. Основним алгоритмом пошуку в DynamoDB є розподілене хешування з використанням консистентного хешування для локалізації даних.

Цей підхід забезпечує складність O(1) для точкових пошуків за ключем партиції, що досягається завдяки прямому відображенню ключа на фізичну партицію через детерміновану хеш-функцію. На противагу цьому, SQL-системи використовують B+-дерева як основну структуру індексів, що забезпечує складність $O(\log_m N)$ для пошуку за первинним ключем, де m – ступінь дерева, а N – кількість записів [8].

Операції повного сканування в DynamoDB мають лінійну складність O(N) і вимагають паралельного опитування всіх партицій, що призводить до мережевих витрат порядку O(P), де P – кількість партицій. У SQL-системах повне сканування також має складність O(N), однак важливою перевагою SQL-систем є наявність оптимізатора запитів, здатного вибирати оптимальний алгоритм з'єднань, тоді як DynamoDB взагалі не підтримує JOIN-операції.

Розуміння алгоритмічної складності основних операцій є критично важливим для проєктування ефективних систем керування даними, що відповідають конкретним вимогам до продуктивності та масштабованості. Аналіз показує оптимальну ефективність DynamoDB для операцій точкового пошуку та діапазонних запитів у межах партиції, тоді як SQL-системи залишаються незмінно ефективними для складних аналітичних запитів та операцій, що вимагають гарантованої цілісності даних.

Висновки. Проведений аналіз демонструє, що сучасні NoSQL системи являють собою практичну реалізацію абстрактних концепцій дискретної математики. Формальні моделі теорії множин, графів, алгебри та ймовірностей трансформуються у високоефективні механізми вирішення складних інженерних проблем масштабування, продуктивності та надійності. Без цього математичного фундаменту створення сучасних систем керування даними було б неможливим, що підтверджує глибокий взаємозв'язок між теоретичною математикою та практичними реалізаціями в галузі інформаційних технологій.

Ця синергія продовжує визначати напрямки розвитку сучасних систем керування даними, відкриваючи нові можливості для подальшого вдосконалення архітектурних рішень.

Проведене дослідження підтвердило фундаментальну роль алгоритмів дискретної математики в архітектурі сучасних інформаційних систем.

Доведено, що такі розділи дискретної математики, як теорія графів, комбінаторний аналіз, абстрактна алгебра, становлять теоретичну основу для побудови ефективних NoSQL систем. Конкретні алгоритмічні рішення, такі як консистентне хешування в DynamoDB та алгоритми на графах в Neo4j,

демонструють практичну реалізацію математичних моделей для забезпечення масштабованості, відмовостійкості та високої продуктивності.

Особливо варто відзначити, що кожен тип NoSQL систем оптимізований для певного класу завдань: системи "ключ-значення" досягають максимальної ефективності для точкових запитів, документні бази даних – для роботи з ієрархічними структурами, стовпчикові – для аналітичних робочих навантажень, а графічні – для аналізу складних взаємозв'язків. Ця спеціалізація ґрунтується на глибокому математичному аналізі характеристик різних типів робочих навантажень.

Майбутній розвиток NoSQL технологій пов'язаний з вдосконаленням гібридних систем, що поєднують переваги різних моделей даних, зокрема інтеграція можливостей реляційних та NoSQL підходів. Це вимагатиме розробки нових алгоритмів для забезпечення консистентності в гетерогенних середовищах.

Література:

1. H. Dong, C. Zhang, G. Li, H. Zhang. Cloud-Native Databases: A Survey. *IEEE Transactions on Knowledge and Data Engineering*, vol. 36, no. 12, Dec. 2024. pp. 7772-7791, DOI: https://doi.org/10.1109/TKDE.2024.3397508.

2. Anuyah S., Bolade E., Agbaakin O. Understanding Graph Databases: A Comprehensive Tutorial and Survey. Nov 2024, 29 p. URL: https://arxiv.org/pdf/2411.09999 (date of access: 14.10.2025). https://doi.org/10.48550/arXiv.2411.09999

3. Kotiranta P., Junkkari M., Nummenmaa J. Performance of Graph and Relational Databases in Complex Queries. *Appl. Sci.* 2022, *12*, 6490. https://doi.org/10.3390/app12136490

4. AWS Database Blog. URL: https://aws.amazon.com/blogs/database (date of access: 14.10.2025).

5. Google Cloud Database Updates. URL: https://cloud.google.com/blog/products/databases (date of access: 14.10.2025).

6. Databases on Azure. URL: https://azure.microsoft.com/en-us/products/category/databases#Explore-all-products (date of access: 14.10.2025).

7. Wilson E. R. Seven databases in seven weeks: A guide to modern databases and the NoSQL movement (2nd ed.). USA : Pragmatic Bookshelf, 2018. 354 p.

8. Amazon DynamoDB Documentation. URL: https://docs.aws.amazon.com/dynamodb (date of access: 14.10.2025).

References:

1. Dong H., Zhang C., L, G., & Zhang H. (2024). Cloud-native databases: A survey. *IEEE Transactions on Knowledge and Data Engineering, 36*(12), 7772–7791. https://doi.org/10.1109/TKDE.2024.3397508

2. Anuyah S., Bolade E., & Agbaakin O. (2024). Understanding graph databases: A comprehensive tutorial and survey. *arXiv*. https://doi.org/10.48550/arXiv.2411.09999

3. Kotiranta P., Junkkari M., & Nummenmaa J. (2022). Performance of graph and relational databases in complex queries. *Applied Sciences, 12*(13), 6490. https://doi.org/10.3390/app12136490

4. AWS database blog. Retrieved from https://aws.amazon.com/blogs/database.

5. Databases on Azure. Retrieved from https://azure.microsoft.com/en-us/products/category/databases#Explore-all-products.

6. Google Cloud database updates. Retrieved from https://cloud.google.com/blog/products/databases.

7. Wilson E. R. (2018). *Seven databases in seven weeks: A guide to modern databases and the NoSQL movement* (2nd ed.). Pragmatic Bookshelf.

8. Amazon DynamoDB documentation. Retrieved from https://docs.aws.amazon.com/dynamodb.