Технічні науки

УДК 004.852

Ловчинський Сергій Броніславович

студент

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Ловчинский Сергей Брониславович

студент

Национальный технический университет Украины

«Киевский политехнический институт имени Игоря Сикорского»

Lovchinsky S.

student

National Technical University of Ukraine

"Igor Sikorsky Kyiv Polytechnic Institute"

АНАЛІЗ ПІДХОДІВ ДО ЗБЕРІГАННЯ ВЕЛИКИХ ДАНИХ
АНАЛИЗ ПОДХОДОВ К ХРАНЕНИЮ БОЛЬШИХ ДАННЫХ
ANALYSIS OF APPROACHES TO STORING BIG DATA

Анотація: Технології великих даних відкривають принципово нові підходи до вирішення багатьох задач для підвищення ефективності різних процесів. Однак, для того щоб скористатися перевагами великих даних, крім аналітичного програмного забезпечення необхідна відповідна інфраструктура для зберігання, обробки і передачі даних.

У даній статті проводиться аналіз можливих способів збереження великих даних, досліджуються обмеження, які не дозволяють зробити це ефективно, а також наводиться огляд сучасного підходу до збереження великих даних.

Ключові слова: Великі дані, реляційні бази даних, СУБД, NoSQL.

Аннотация: Технологии больших данных открывают принципиально новые подходы к решению многих задач с целью увеличения производительности разных действий. Однако, для того чтобы использовать достоинства больших данных, помимо аналитического программного обеспечения нужна надлежащая инфраструктура с целью хранения, обрабатывания и передачи данных.

В этой статье проводится исследование возможных методов хранения больших данных, исследуются ограничения, которые не дают возможность сделать это продуктивно, а кроме того приводится обзор современного подхода к хранению больших данных.

Ключевые слова: Большие данные, реляционные базы данных, СУБД, NoSQL.

Abstract: The technology of big data reveals fundamentally new approaches to solving many problems for increasing the efficiency of various processes. However, in order to take advantage of the great data, in addition to analytical software, an appropriate infrastructure for storage, processing and data transmission is required.

This article analyzes possible methods for storing big data, examines limitations that do not allow it to be efficiently performed, and provides an overview of the modern approach to storing big data.

Keywords: Big data, relational databases, DBMS, NoSQL.

Постановка проблеми. Стрімко зростаючі обсяги інформації в умовах високонавантаженого режиму роботи сучасних інтернет-сервісів породжують нові складні завдання по організації її зберігання і обробки. Реляційні системи управління даними на сьогоднішній день є одним з найпоширеніших засобів представлення та маніпулювання інформацією в комп'ютерних системах. Однак реляційна модель не забезпечує обробку великої кількості неструктурованою інформації.

Виклад основного матеріалу дослідження. Великі дані — характеризуються обсягом, різноманітністю і швидкістю, з якою структуровані і неструктуровані дані надходять по мережах передачі в процесори і сховища, поряд з процесами перетворення цих даних в цінну для бізнесу інформацію. [1]

Як видно з цього визначення, великі дані мають чотири основні характеристики: обсяг, різноманітність, швидкість і цінність. Розглянемо їх докладніше:

- Обсяг. Наростаюча кількість даних, що створюється як людьми, так і машинами, та ставить до IT інфраструктурі нові вимоги щодо зберігання, обробки і надання доступу.
- Різноманітність. Дані містять різноманітну інформацію, представлену різними структурами. З усім цим, від логів доступу веб-сервера до операцій по кредитних картах, від результатів наукових експериментів до фотографій і відео, необхідно вміти працювати.
- Швидкість. Важливо усвідомлювати, що під швидкістю розуміється не тільки швидкість, з якою дані надходять в сховище, а й швидкість з якою важлива інформація з цих даних витягується.
- Цінність. Великі обсяги даних – це цінний ресурс. Але ще ціннішим він стає, якщо дозволяє відповідати на нагальні питання або питання, які можуть виникнути в майбутньому.

До певного моменту, практично єдиним способом для збереження та обробки даних була реляційна СУБД. Але зі збільшенням обсягів інформації з'явилися проблеми, з якими класична реляційна архітектура не справлялася. Якщо СУБД припиняє справлятися з об'ємом виконуваних операцій, можна застосувати наступні дії:

1. Найпростіший, при наявності фінансів, спосіб – це вертикальне масштабування, що являє собою здатність системи збільшувати продуктивність, використовуючи додаткове апаратне забезпечення на одному сервері. Однак нескінченно потужного сервера не існує, а значить вертикальний ріст кінцевий.
2. Більш витратний спосіб – це оптимізувати запити, проаналізувавши плани їх виконання, і створити додаткові індекси. Такий метод може принести тимчасове покращення, але додаткові індекси породжують додаткові операції, а з ростом обсягів оброблюваних даних ці додаткові операції призводять до деградації.
3. Наступним кроком може бути впровадження кеша на читання. При правильній організації такого рішення, ми можемо позбавити СУБД від значної частини операцій читання, але жертвуємо строгою консистентністю даних. Варто відзначити, що цей підхід призводить до ускладнення клієнтського програмного забезпечення.
4. Вибудовування операцій вставки та оновлення в черзі буде наступним можливим рішення, але розмір черги обмежений. До того ж, для забезпечення суворої консистентності, нам доведеться організувати персистентність самої черги, що являється досить складним завданням.
5. Нарешті, коли всі інші способи перестають працювати, настає момент переглянути спосіб організації самих даних. В першу чергу – провести денормалізацію схеми, щоб зменшити число нелокальних звернень. [2]

Підводячи підсумки, можна зробити висновок, що спроби пристосувати реляційну СУБД до роботи з великими даними призводять до наступного:

1. Відмова від строгої консистентності.
2. Догляду від нормалізації і впровадження надмірності.
3. Втрати виразності мови SQL і необхідності моделювати частину її функцій програмно.
4. Суттєвого ускладнення клієнтського програмного забезпечення.
5. Складнощі підтримки працездатності та відмовостійкості отриманого рішення.

Для вирішення проблем, які виникають під час збереження та читання великих даних було спроектовано архітектуру, здатну адаптуватися до зростаючих обсягів даних і ефективно їх обробляти. Така архітектура називається NoSQL. Цей термін позначає ряд підходів, спрямованих на реалізацію сховищ баз даних, які мають суттєві відмінності від моделей, що використовуються в традиційних реляційних СУБД з доступом до даних засобами мови SQL. Застосовується до баз даних, в яких робиться спроба вирішити проблеми масштабованості і доступності за рахунок атомарності та узгодженості даних. [3] Розглянемо основні особливості NoSQL підходу:

Виключення зайвого ускладнення. Реляційні бази даних виконують величезну кількість різних функцій і забезпечують сувору консистентність даних. Однак для багатьох додатків подібний набір функцій, а також задоволення вимог ACID є зайвими.

Висока пропускна здатність. Багато NoSQL рішень забезпечують набагато більш високу пропускну здатність даних ніж традиційні СУБД.

Необмежене горизонтальне масштабування. На противагу реляційним СУБД, NoSQL рішення проектуються для необмеженого горизонтального масштабування. При цьому додавання і видалення вузлів в кластері ніяк не позначається на працездатності системи. Додатковою перевагою подібної архітектури є те, що NoSQL кластер може бути

розгорнутий на звичайному апаратному забезпеченні, істотно знижуючи вартість всієї системи.

Продуктивність в обмін на консистентність. При описі підходу NoSQL не можна не згадати теорему CAP. Дотримуючись цієї теореми, багато NoSQL бази даних реалізують доступність даних і стійкість до поділу, жертвуючи консистентністю в перевагу високої продуктивності. І дійсно, для багатьох класів додатків сувора консистентність даних – це те, від чого цілком можна відмовитися. [4]

На сьогоднішній день створено велику кількість NoSQL рішень, які можна класифікувати наступним чином:

Сховище ключ-значення. Відмінною особливістю є проста модель даних – асоціативний масив або словник, що дозволяє працювати з даними по ключу. Приклади використання:

- Кешування – швидке і часте збереження даних для майбутнього використання.
- Черга – деякі бази даних типу ключ-значення підтримують списки, набори і черги.
- Розподіл інформації / завдань – використовується для реалізації паттерна Pub / Sub.
- Оновлення інформації в реальному часі. [5]

Основні представники цього типу моделі збереження даних:

- Redis – база даних в оперативній пам'яті з додатковою відмовостійкістю.
- Riak – розподілене, реплікаційне сховище.
- Memcached – розподілена БД в оперативній пам'яті.

Графічне представлення вище описаного сховища показано на рисунку 1.

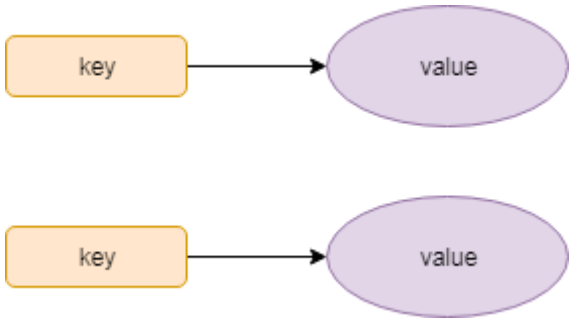

Рисунок 1 – Сховище ключ-значення

Документне сховище. Модель даних подібних сховищ дозволяє об'єднувати безліч пар ключ-значення в абстракцію, звану «документ». Документи можуть мати вкладену структуру і об'єднуватися в колекції. Приклади використання:

- Вкладена інформація – документо-орієнтовані сховища відмінно працюють з глибокою вкладеністю складної інформації.
- Підтримка JavaScript – одна з відмінних рис документо-орієнтованих сховищ це те, як вони працюють з іншими додатками: підтримка JSON. [6]

Представниками такого сховища являються:

- Couchbase – документо-орієнтована система управління базами даних заснована на JSON.
- MongoDB – документо-орієнтована система управління базами даних з відкритим вихідним кодом, яка не потребує опису схеми даних.

На рисунку 2 продемонстровано графічне представлення документного сховища.

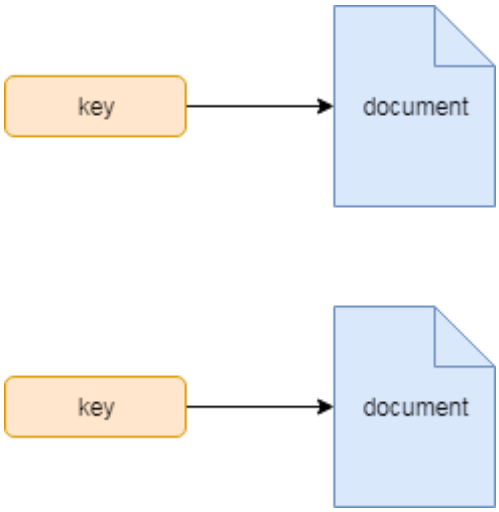

Рисунок 2 – Документне сховище

Колонкове сховище. Цей тип здається найбільш схожим з традиційними реляційними СУБД. Модель даних сховищ подібного типу має на увазі зберігання значень як неінтерпретованих байтових масивів, адресованих кортежами. Приклади застосування:

- Зберігання неструктурованих, що не руйнуються даних – якщо вам необхідно зберігати великі обсяги даних протягом довгого часу, то такі БД дуже добре впораються з завданням.
- Масштабування – за задумом такі бази даних легко масштабуються. Вони легко справляються з будь-яким обсягом даних.

Представники цього типу сховища:

- Cassandra – структура даних заснована на BigTable і DynamoDB.
- HBase – сховище для Apache Hadoop засноване на принципах BigTable.

На рисунку 3 показується графічне представлення колонкового сховища.

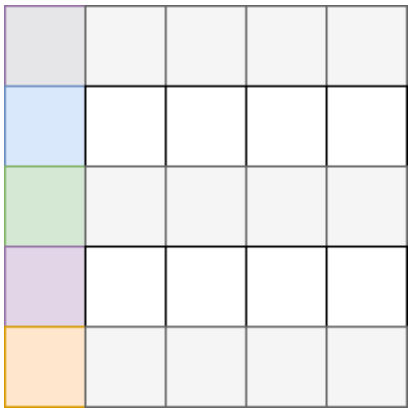

Рисунок 3 – Колонкове сховище

Сховища на графах. Подібні сховища застосовуються для роботи з даними, які природним чином представляються графами (наприклад, соціальна мережа). Модель даних складається з вершин, ребер і властивостей. [7] Основні представники графового сховища:

- OrientDB – відкрита СУБД, яка об'єднує в собі можливості документо-орієнтованої і графо-орієнтованої БД.
- Neo4J – графовая система управління базами даних з відкритим вихідним кодом, реалізована на Java.

Графічне представлення графового сховища показано на рисунку 4.

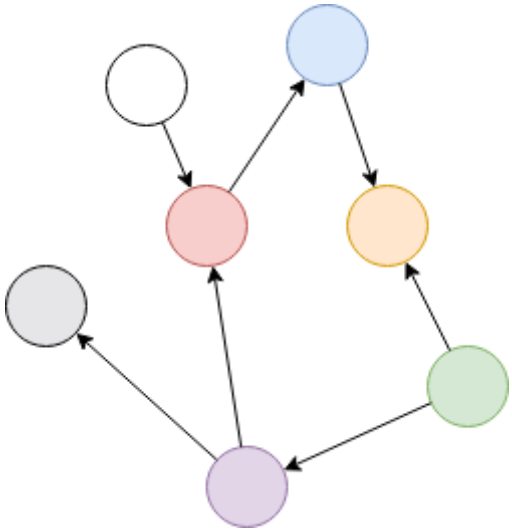

Рисунок 4 – Графове сховище

Наведені схеми даних в разі використання NoSQL-рішень може здійснюватися через використання різних структур даних: хеш-таблиць, дерев та інших. Оцінка продуктивності таких рішень наведена в таблиці 1.

Таблиця 1 – Порівняльна таблиця сховищ за моделями даних

Модель даних	**Продуктивність**	**Масштабованість**	**Гнучкість**	**Функціональність**
Key–Value Store	high	high	high	variable (none)
Column-Oriented Store	high	high	moderate	minimal
Document-Oriented Store	high	variable (high)	high	variable (low)
Graph Database	variable	variable	high	graph theory
Relational Database	variable	variable	low	relational algebra

Висновки. В даній статті були розглянуті проблеми, які поставили перед реляційними СУБД великі дані. Проаналізувавши можливі шляхи вирішення цих проблем, було вказано на ті концептуальні обмеження, які не дозволяють класичної реляційної архітектурі справлятися зі стрімко зростаючим обсягом інформації. Далі були розглянуті основні рішення NoSQL. Проаналізовані архітектурні особливості, які дозволяють кожному з них ефективно вирішувати поставлену задачу. Важливо відзначити, що жоден з представлених підходів не пропонує рішення всіх можливих завдань, які виникли в контексті великих даних. Кожен з них ефективно вирішує свій клас завдань, дозволяючи користувачеві вибрати найбільш підходящий для нього інструмент.

Література

1. Min Chen. Big Data. Related Technologies, Challenges, and Future Prospects / Min Chen, Shiwen Mao, Yin Zhang, Victor C.M. Leung – Spinger, 2014. – 100 с.
2. О. Бунин. Разработка высоконагруженых систем / О. Бунин – Д. «Издательство Олега Бунина», 2011 – 403 с.
3. М. Фаулер. NoSQL: новая методология разработки нереляционных баз данных / М. Фаулер, Прамодкумар Дж. Садаладж – М.: «Вильямс», 2013. – 192 с.
4. Л. Черняк. Смутное время СУБД / Л. Черняк. – М.: Открытые системы, 2012. – 106 с.
5. Shashank Tiwari. Professional NoSQL / Shashank Tiwari – Packt Publishing, 2011. – 384 с.
6. NoSQL – коротко о главном. – Режим доступа: https://habrahabr.ru/company/oleg-bunin/blog/319052/. – Дата доступа: 20.06.2017
7. Сравнение NoSQL систем управления базами данны. – Режим доступа: http://devacademy.ru/posts/nosql/. – Дата доступа: 20.06.2017