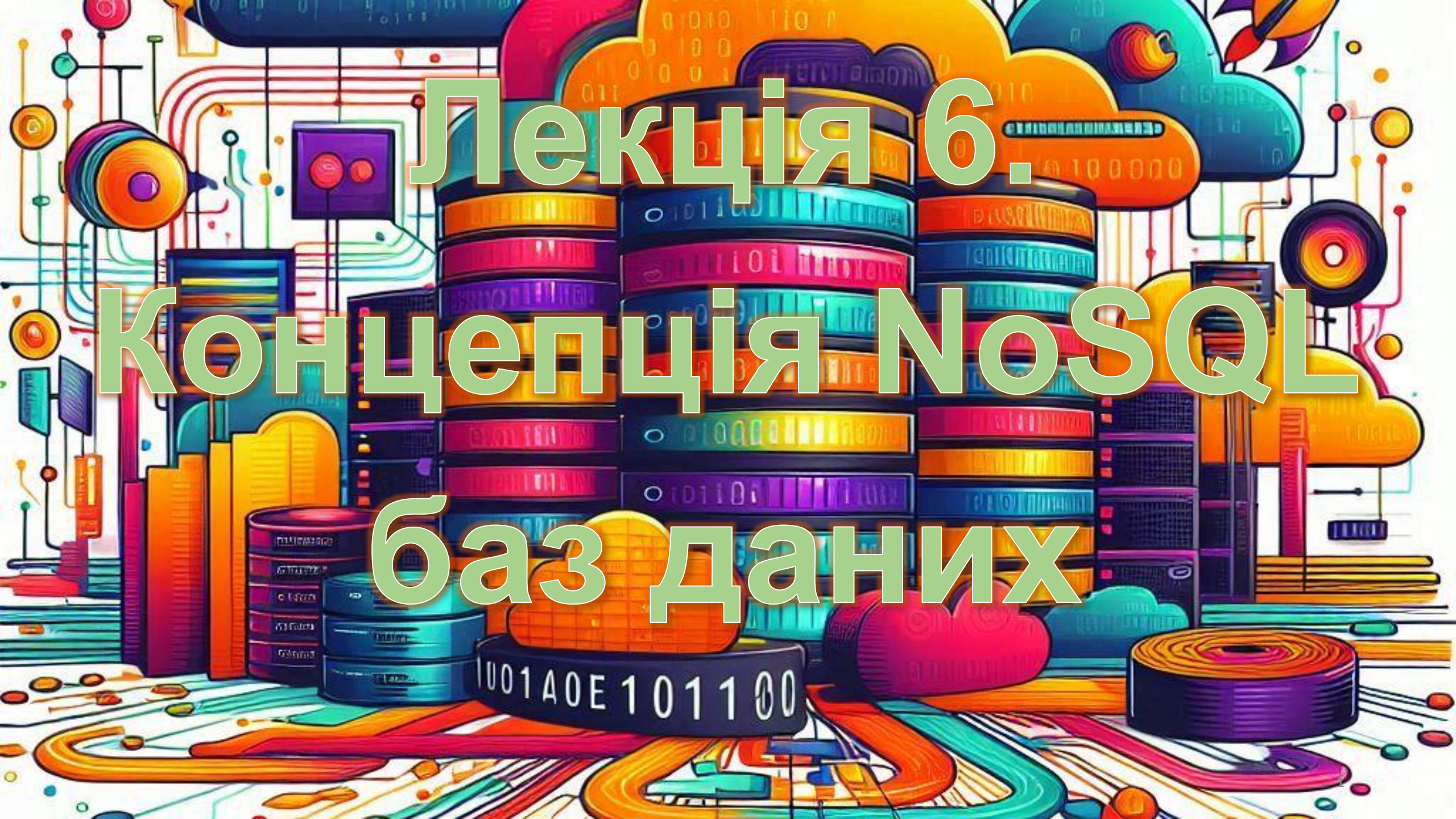
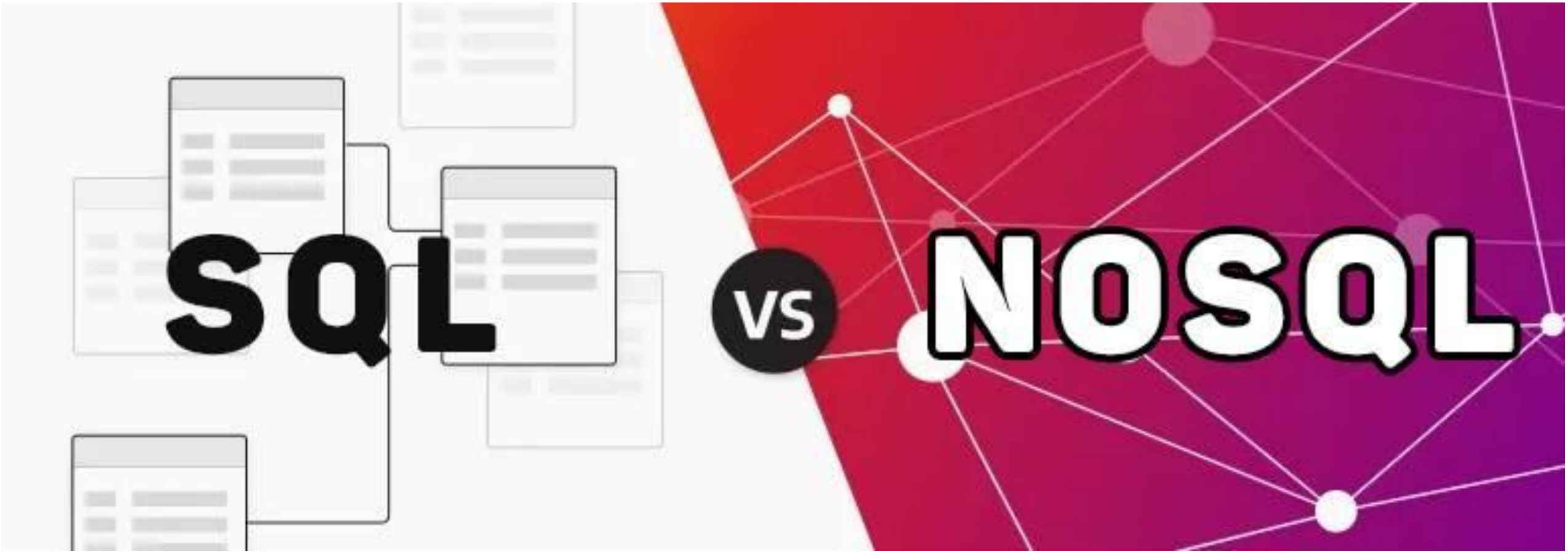

Лекція 6.
Концепція NoSQL
баз даних

Актуальність баз даних NoSQL у сучасному світі



База даних – це те місце, де ви зберігаєте та систематизуєте всі дані, які збирає ваш додаток, а система управління базою даних (СУБД) – це програмне забезпечення для зручного управління базою даних.

Виникнення та розвиток NoSQL баз даних

Зростання обсягів даних. Наприкінці 1990-х років обсяг даних, які обробляли організації, почав зростати експоненціально.

Різноманітність даних. Дані більше не були виключно структурованими. Веб-сторінки, мультимедійні файли, соціальні мережі та інші джерела генерували напівструктуровану та неструктуровану інформацію, яку було складно інтегрувати в реляційну модель.

Вимоги до швидкості. У новій економіці часу затримка навіть у кілька секунд могла призвести до втрати клієнтів або фінансових втрат. реляційні бази даних, побудовані для забезпечення транзакційної цілісності, не завжди могли забезпечити необхідну продуктивність.

CAP-теорема. Концепція, сформульована у 2000 році, вказувала на неможливість одночасного досягнення узгодженості, доступності та стійкості до поділу у розподілених системах.

Виникнення та розвиток NoSQL баз даних

Перші практичні реалізації NoSQL баз з'явилися наприкінці 2000-х років у великих технологічних компаніях:

Google Bigtable (2004) – стовпцева база даних, розроблена для роботи з величезними обсягами даних у розподіленому середовищі.

Amazon Dynamo (2007) – база даних "ключ-значення", яка стала основою для багатьох сучасних NoSQL рішень.

Apache Cassandra (2008) – стовпцева база даних із високою доступністю та масштабованістю.

MongoDB (2009) – документно-орієнтована база даних, яка стала однією з найпопулярніших у своєму класі.

Виникнення та розвиток NoSQL баз даних

NoSQL бази даних пройшли кілька етапів розвитку:

Перший етап: експериментальний. У цей період (2000-2010) основними користувачами NoSQL баз були великі корпорації, які розробляли власні рішення для вирішення специфічних завдань.

Другий етап: комерціалізація. У 2010-х роках багато NoSQL баз стали доступними як комерційні продукти з підтримкою спільнот розробників. Наприклад, MongoDB, Couchbase та Redis отримали широку популярність.

Третій етап: інтеграція з SQL. Розробники почали додавати до NoSQL баз можливість використання SQL-подібних мов запитів (наприклад, SQL-подібний синтаксис у Cassandra).

Четвертий етап: гібридні системи. На сьогодні багато СУБД поєднують риси як реляційних, так і NoSQL баз. Наприклад, PostgreSQL інтегрує підтримку JSON, а Microsoft Azure Cosmos DB підтримує кілька моделей даних одночасно.

Виникнення та розвиток NoSQL баз даних

NoSQL бази мають низку **переваг**, які зробили їх популярними:

Гнучкість. Відсутність жорстких схем дозволяє працювати з напівструктурованими та неструктурованими даними.

Масштабованість. Горизонтальне масштабування забезпечує обробку великих обсягів даних.

Швидкість. Висока продуктивність при виконанні операцій читання та запису.

Різноманіття моделей. Бази "ключ-значення", документно-орієнтовані, стовпцеві та графові бази дозволяють вибрати оптимальне рішення для конкретного завдання.

Виникнення та розвиток NoSQL баз даних

Попри свої переваги, NoSQL бази даних мають і **недоліки**:

Відсутність стандартизації. У кожної бази свої API, що ускладнює міграцію.

Обмежена підтримка транзакцій. Багато NoSQL баз не гарантують повну відповідність принципам ACID.

Складність у проектуванні. Для досягнення оптимальної продуктивності потрібні глибокі знання архітектури конкретної СУБД.

Основні характеристики NoSQL баз даних

Відсутність жорстких схем (schema-less). Однією з ключових особливостей NoSQL баз даних є відсутність суворої схеми даних.

Горизонтальна масштабованість. На відміну від реляційних баз даних, які здебільшого підтримують вертикальну масштабованість (збільшення потужності одного сервера), NoSQL бази даних орієнтовані на горизонтальну масштабованість.

Висока продуктивність при роботі з великими обсягами даних. NoSQL бази даних розроблені для роботи з великими обсягами даних і забезпечення високої швидкості операцій читання та запису.

Гнучкість при роботі з напівструктурованими та неструктурованими даними. Реляційні бази даних найкраще працюють із структурованими даними, тоді як NoSQL бази даних можуть обробляти напівструктуровані та неструктуровані дані

Основні характеристики NoSQL баз даних

CAP-теорема та її роль у NoSQL базах даних. CAP-теорема стверджує, що розподілені бази даних не можуть одночасно забезпечувати три властивості:

- **Consistency (узгодженість)**: усі клієнти бачать однакові дані одночасно.
- **Availability (доступність)**: кожен запит отримує відповідь, навіть якщо деякі вузли недоступні.
- **Partition tolerance (стійкість до розділення)**: система продовжує працювати, навіть якщо частина вузлів не може обмінюватися даними.

Відмінності NoSQL баз даних від реляційних баз даних

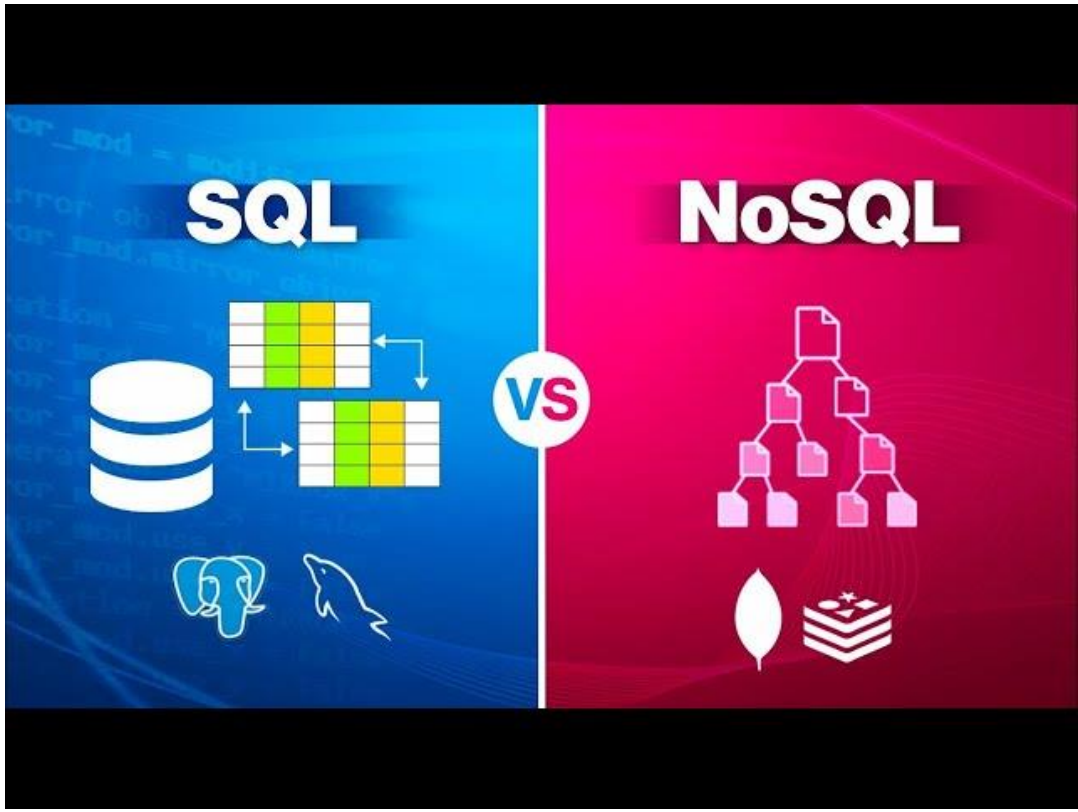
Структура даних: реляційна (таблиці) vs нереляційна (гнучкі формати).

Мова запитів: SQL vs API чи спеціалізовані мови запитів.

Масштабованість: вертикальна (SQL) vs горизонтальна (NoSQL).

Типи даних: фіксовані схеми (SQL) vs довільні формати (JSON, XML тощо).

Узгодженість і доступність: Strong Consistency (SQL) vs Eventual Consistency (NoSQL).

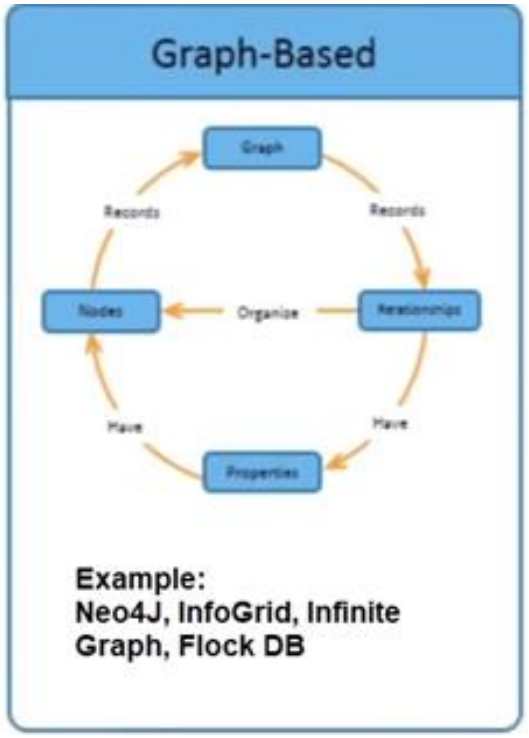


Типи NoSQL баз даних

Бази даних “ключ-значення”

Документно-орієнтовані бази даних

Стовпцеві бази даних



Графові бази даних

Типи NoSQL баз даних: Бази даних “ключ-значення”

Бази даних “ключ-значення” (Key-Value Databases) — це прості за структурою сховища, у яких дані зберігаються у форматі пара “ключ”-“значення”. Ключ є унікальним ідентифікатором, за яким можна швидко отримати значення.

Основні операції баз даних “ключ-значення”:

- *Додавання даних*: створення нової пари ключ-значення.
- *Читання даних*: отримання значення за ключем.
- *Оновлення даних*: зміна значення для наявного ключа.
- *Видалення даних*: видалення пари ключ-значення.

Приклади баз даних “ключ-значення”

Redis — це популярна відкрита база даних “ключ-значення”, яка працює в оперативній пам'яті.
DynamoDB — це керована база даних від Amazon Web Services, яка також використовує модель “ключ-значення”.

Типи NoSQL баз даних: Бази даних “ключ-значення”

Сфери використання

Кешування. Однією з головних сфер використання баз даних “ключ-значення” є кешування. Завдяки високій швидкості доступу до даних, вони дозволяють зберігати часто запитувану інформацію, що зменшує навантаження на основну базу даних і підвищує продуктивність системи.

Сесії користувачів. Бази даних “ключ-значення” ідеально підходять для зберігання інформації про сесії користувачів. Ключем може бути ідентифікатор сесії, а значенням — відповідні дані (наприклад, налаштування, стан авторизації тощо).

Системи повідомлень. Системи обміну повідомленнями також активно використовують бази даних “ключ-значення” для зберігання повідомлень, черг задач та іншої інформації.

Типи NoSQL баз даних: Бази даних “ключ-значення”

Переваги баз даних “ключ-значення”

Простота: проста модель даних забезпечує легкість у використанні й адмініструванні.

Швидкість: висока продуктивність при операціях читання й запису.

Масштабованість: можливість горизонтального масштабування для роботи з великими обсягами даних.

Типи NoSQL баз даних: Бази даних “ключ-значення”

Недоліки баз даних “ключ-значення”

Обмежена функціональність: відсутність складних запитів і обробки даних, як у реляційних базах даних.

Проблеми із взаємозв'язками: відсутність природної підтримки зв'язків між даними.

Споживання пам'яті: через зберігання даних у пам'яті, як у Redis, можуть виникати обмеження на обсяг даних.

Типи NoSQL баз даних: Документно-орієнтовані бази даних

Визначення та особливості

Формати зберігання даних: Найпоширенішими форматами є JSON (JavaScript Object Notation) та BSON (Binary JSON).

Гнучкість структури: Відсутність суворої схеми дозволяє зберігати документи з різними структурами в одній колекції.

Швидкість доступу: Завдяки оптимізованим механізмам індексації та можливості зберігати дані без суворої нормалізації, документно-орієнтовані бази даних забезпечують високу продуктивність.

Підтримка вкладених структур: Документи можуть містити вкладені об'єкти та масиви, що спрощує моделювання складних структур даних без необхідності використовувати зв'язки між таблицями.

API для роботи з даними: Для доступу до даних використовуються спеціалізовані API, які підтримують операції CRUD (Create, Read, Update, Delete).

Типи NoSQL баз даних: Документно-орієнтовані бази даних

Приклади документно-орієнтованих баз даних

MongoDB: Найпопулярніша документно-орієнтована база даних, яка використовує формат BSON для зберігання даних. MongoDB підтримує складні запити, агрегації, транзакції та горизонтальну масштабованість через шардинг. Її часто використовують для розробки веб-додатків, мобільних застосунків, зберігання мультимедійного контента.

CouchDB: Ця база даних використовує JSON для зберігання документів і HTTP для доступу до даних через RESTful API. CouchDB забезпечує простоту використання та високу стійкість завдяки механізму реплікації та можливості працювати в офлайн-режимі з подальшою синхронізацією.

Типи NoSQL баз даних: Документно-орієнтовані бази даних

Сфери використання

Управління контентом: Завдяки можливості зберігати документи з різною структурою, ці бази даних ідеально підходять для управління великими обсягами контенту, включаючи статті, блоги, медіафайли тощо.

Аналітика: Документно-орієнтовані бази даних ефективно працюють з великими обсягами напівструктурованих даних, що робить їх популярними для збору, обробки та аналізу логів, даних про поведінку користувачів, подій тощо.

Електронна комерція: У системах онлайн-торгівлі документно-орієнтовані бази даних дозволяють зберігати інформацію про товари, замовлення, користувачів та їхні сесії.

Інтернет речей (IoT): У системах IoT, де дані надходять із багатьох пристроїв у реальному часі, документно-орієнтовані бази даних забезпечують високу продуктивність і можливість зберігати нестандартні структури даних.

Типи NoSQL баз даних: Стовпцеві бази даних

Принцип організації даних у стовпцях

Основна ідея стовпцевих баз даних полягає у зберіганні даних не за рядками, як у реляційних базах даних, а за стовпцями.

Ця модель забезпечує кілька ключових переваг:

Ефективність читання. Завдяки зберіганню даних у стовпцях, системи можуть швидко виконувати запити, які стосуються лише певних полів.

Стиснення даних. Дані в межах одного стовпця часто мають схожий тип і значення, що спрощує їх стиснення.

Масштабованість. Стовпцеві бази даних легко розширювати горизонтально, додаючи нові вузли для обробки даних.

Типи NoSQL баз даних: Стовпцеві бази даних

Приклади стовпцевих баз даних

Apache Cassandra є однією з найвідоміших стовпцевих баз даних. Вона була розроблена для забезпечення високої доступності та масштабованості без компромісів щодо продуктивності. Cassandra використовує розподілену архітектуру, що дозволяє ефективно працювати з величезними обсягами даних.

HBase є ще одним популярним рішенням, створеним на базі розподіленої файлової системи Hadoop. Ця база даних оптимізована для роботи з великими обсягами напівструктурованих і неструктурованих даних. HBase часто використовується для обробки часових рядів, логів і даних IoT.

Типи NoSQL баз даних: Стовпцеві бази даних

Сфери використання стовпцевих баз даних

Обробка великих обсягів даних. Стовпцеві бази даних є ідеальним рішенням для сценаріїв, де необхідно зберігати та обробляти величезні обсяги інформації,

Аналітичні запити. Завдяки своїй архітектурі стовпцеві бази даних ефективно виконують аналітичні запити, які стосуються обробки даних лише у вибраних стовпцях.

Системи рекомендацій. Стовпцеві бази даних, такі як Apache Cassandra, можуть використовуватися для створення персоналізованих систем рекомендацій.

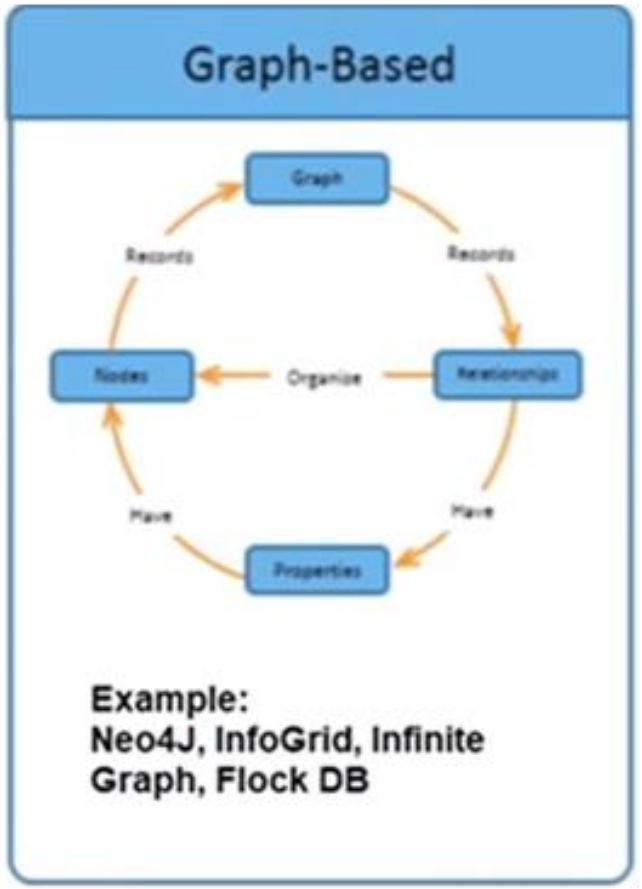
Складні часові ряди. HBase та подібні бази даних є відмінним вибором для зберігання та обробки часових рядів.

Машинне навчання та обробка даних. У контексті машинного навчання стовпцеві бази даних часто використовуються для підготовки великих масивів даних.

Типи NoSQL баз даних: Графові бази даних

Основна ідея: збереження даних у вигляді вузлів та зв'язків (графів).

Графові бази даних базуються на теорії графів, яка є розділом математики, що вивчає властивості та взаємозв'язки між об'єктами. У графових базах даних інформація зберігається у вигляді двох основних елементів: вузлів (nodes) та зв'язків (edges). Вузли представляють об'єкти або сутності, тоді як зв'язки описують відносини між цими об'єктами. Наприклад, у соціальній мережі вузлами можуть бути користувачі, а зв'язками — їхні дружні зв'язки або взаємодії.

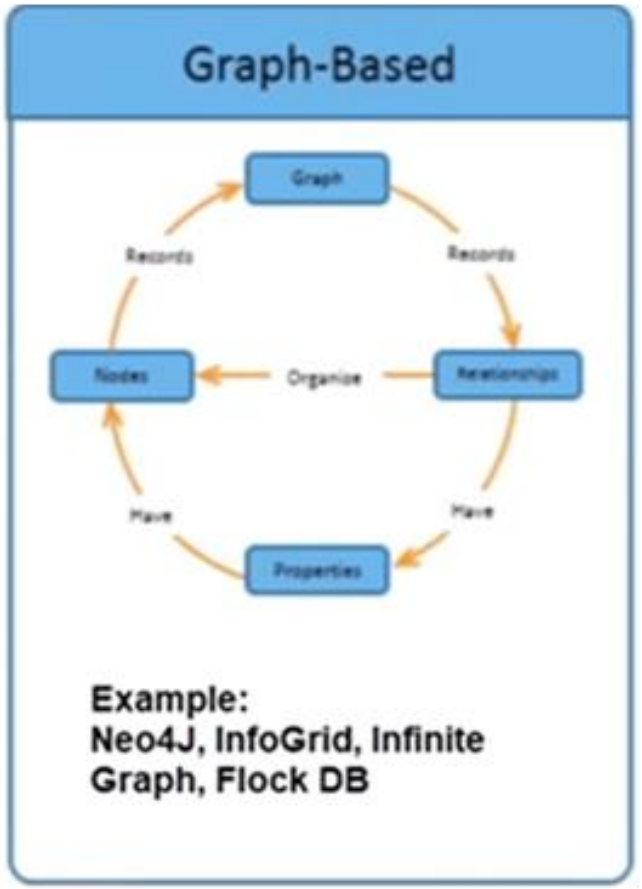


Типи NoSQL баз даних: Графові бази даних

Приклади графових баз даних.

Neo4j — це одна з найвідоміших графових баз даних, яка використовує власну модель зберігання даних, оптимізовану для роботи з графами. Вона підтримує мову запитів Cypher, яка спеціально розроблена для роботи з графовими даними. Cypher дозволяє легко формулювати складні запити, такі як пошук шляхів між вузлами або виявлення шаблонів у взаємозв'язках.

ArangoDB — це мультимодельна база даних, яка підтримує не лише графову модель, але й документну та ключ-значення. Це робить її гнучким рішенням для різних типів застосувань. У контексті графових даних ArangoDB використовує модель, яка дозволяє зберігати вузли та зв'язки, а також виконувати складні графові запити.



Типи NoSQL баз даних: Графові бази даних

Сфери використання графових баз даних

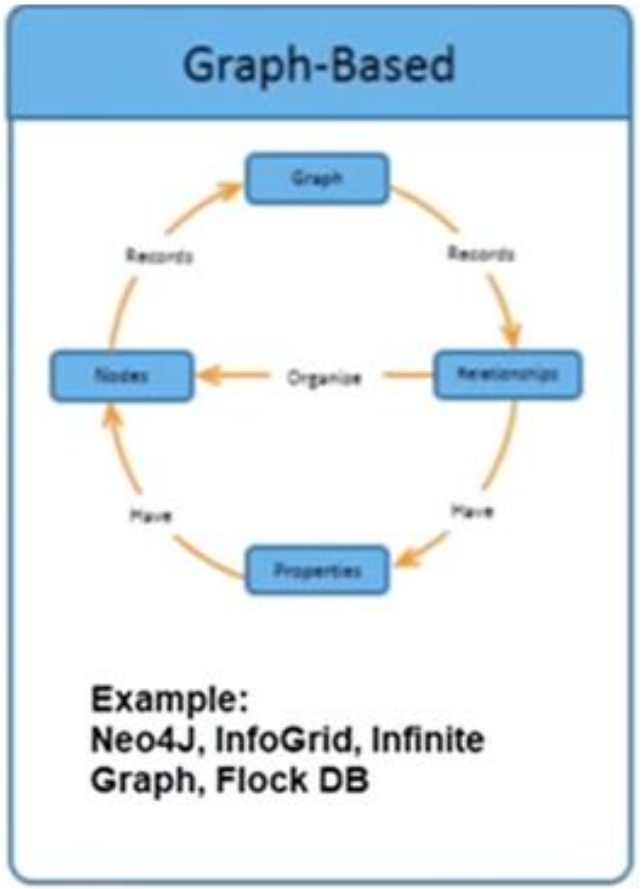
Соціальні мережі: У соціальних мережах графові бази даних використовуються для моделювання зв'язків між користувачами, групами та контентом.

Рекомендаційні системи: Графові бази даних є ідеальним інструментом для побудови рекомендаційних систем.

Управління мережами: У сфері управління мережами (наприклад, телекомунікаційними або комп'ютерними) графові бази даних використовуються для моделювання топології мережі, аналізу шляхів передачі даних та виявлення потенційних проблем.

Біоінформатика: У біоінформатиці графові бази даних використовуються для аналізу біологічних мереж, таких як генні взаємодії або метаболічні шляхи.

Фінанси: У фінансовій сфері графові бази даних допомагають виявляти шахрайство, аналізувати транзакційні зв'язки та моделювати фінансові мережі.



Порівняння популярних NoSQL баз даних:
Redis vs MongoDB

Основні відмінності між Redis та MongoDB

Модель даних:

- **Redis**: Redis є базою даних типу “ключ-значення”, де дані зберігаються у вигляді пар ключ-значення.
- **MongoDB**: MongoDB є документно-орієнтованою базою даних, яка зберігає дані у форматах JSON або BSON.

Швидкість і продуктивність:

- **Redis**: Завдяки роботі в оперативній пам’яті, Redis забезпечує мілісекундний час доступу до даних, що робить його ідеальним для високошвидкісних додатків, таких як кешування, обробка черг та керування сесіями.
- **MongoDB**: MongoDB використовує зберігання даних на диску, але має індексацію та підтримує виконання складних запитів. Вона підходить для програм, які потребують роботи з великими обсягами даних, але не потребують надшвидкої обробки.

Порівняння популярних NoSQL баз даних:
Redis vs MongoDB

Основні відмінності між Redis та MongoDB

Масштабованість:

- **Redis**: Redis добре масштабується горизонтально, але здебільшого використовується для специфічних задач, що вимагають високої продуктивності.
- **MongoDB**: MongoDB підтримує горизонтальне шардінг, що дозволяє ефективно розподіляти дані між кількома серверами. Вона підходить для роботи з великими базами даних і забезпечує відмовостійкість.

Підтримка транзакцій:

- **Redis**: Redis підтримує транзакції, але у вигляді простих послідовних команд. Це обмежує можливість роботи зі складними операціями.
- **MongoDB**: MongoDB пропонує повноцінну підтримку ACID-транзакцій з версії 4.0, що робить її більш придатною для критично важливих систем, де необхідна узгодженість даних.

Порівняння популярних NoSQL баз даних:
Redis vs MongoDB

Основні відмінності між Redis та MongoDB

Гнучкість запитів:

- **Redis**: Redis не підтримує складні запити. Його функціональність зосереджена на швидкому доступі до даних за ключами.
- **MongoDB**: MongoDB підтримує складні запити, агрегації, фільтрацію та індексацію, що дозволяє ефективно працювати зі структурованими та напівструктурованими даними.

Порівняння популярних NoSQL баз даних:
Redis vs MongoDB

Області застосування Redis

Redis ідеально підходить для сценаріїв, де потрібна висока швидкість обробки даних. Основні області використання Redis:

- **Кешування**: Redis широко використовується як кеш через свою здатність швидко обробляти дані.
- **Управління сесіями**: Завдяки швидкому запису та зчитуванню, Redis є популярним вибором для зберігання інформації про сесії користувачів.
- **Обробка черг**: Redis може використовуватися для створення черг повідомлень завдяки своїй підтримці списків і множин.
- **Аналітика в реальному часі**: Завдяки швидкій обробці даних, Redis підходить для аналізу даних у реальному часі.

Порівняння популярних NoSQL баз даних:
Redis vs MongoDB

Області застосування MongoDB

MongoDB підходить для систем, які потребують роботи зі складними даними. Основні сценарії використання MongoDB:

- **Управління контентом**: MongoDB дозволяє зберігати складні структури контенту, такі як блоги, статті або мультимедійні об’єкти.
- **Електронна комерція**: Завдяки своїй гнучкості, MongoDB використовується для зберігання інформації про товари, замовлення та клієнтів.
- **Аналітика**: MongoDB підтримує складні запити, що робить її придатною для бізнес-аналітики.
- **Інтернет речей (IoT)**: MongoDB дозволяє зберігати напівструктуровані дані, які генеруються IoT-пристроями.

Порівняння популярних NoSQL баз даних:
Cassandra vs Apache HBase

Основні характеристики Cassandra

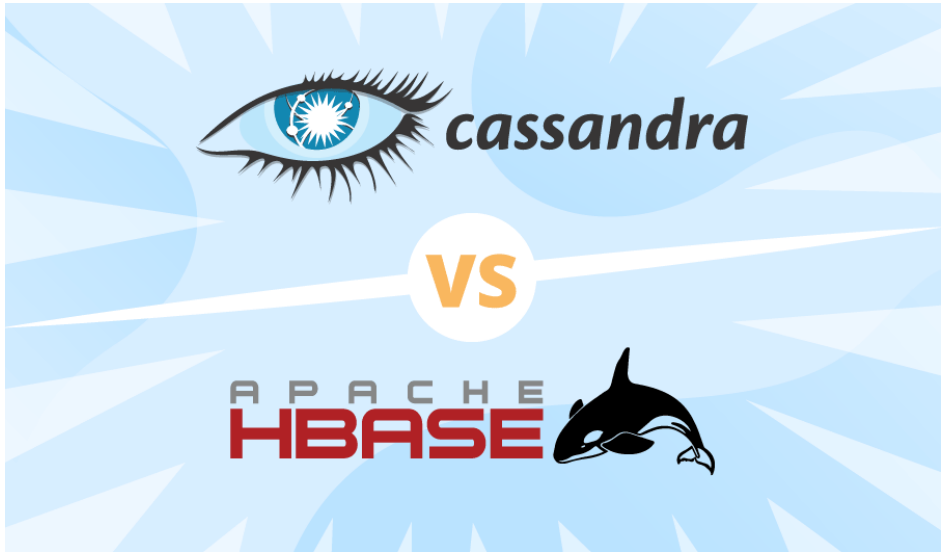
Cassandra — це розподілена база даних, створена для забезпечення високої доступності та стійкості до відмов. Вона використовує модель "кільцевого" розподілу даних, що дозволяє розміщувати дані на кількох вузлах кластеру. Ключові особливості Cassandra:

Модель даних:

- Використовує стовпцеву модель зберігання, де дані організовані в таблиці, але з можливістю зберігати значення у вигляді стовпців.
- Підтримує складні структури даних, такі як колекції (списки, множини, карти).

Масштабованість:

- Горизонтальна масштабованість є основною перевагою Cassandra. Нові вузли можуть бути додані без зупинки роботи системи.
- Дані автоматично реплікуються між вузлами для забезпечення високої доступності.



Порівняння популярних NoSQL баз даних:
Cassandra vs Apache HBase

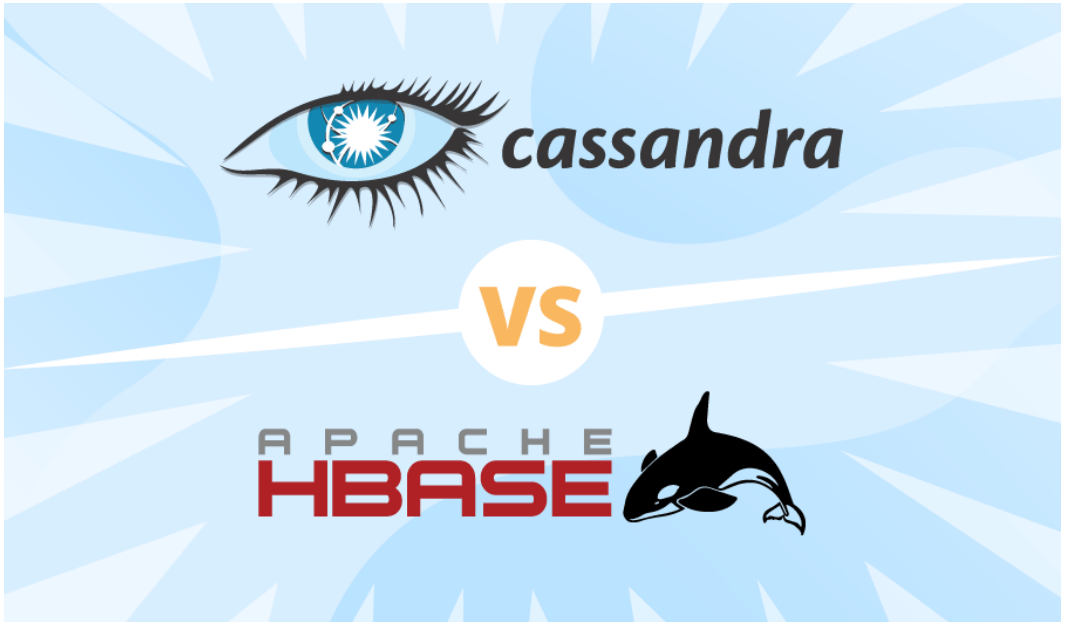
Основні характеристики Cassandra

Консистентність:

- Підтримує налаштовувану консистентність, що дозволяє обирати між високою доступністю (Availability) та строгістю даних (Consistency).

Архітектура:

- Побудована без центрального вузла, що забезпечує відсутність єдиної точки відмови.



Порівняння популярних NoSQL баз даних:
Cassandra vs Apache HBase

Основні характеристики Apache HBase

HBase — це база даних, побудована на основі HDFS (Hadoop Distributed File System). Вона спроектована для обробки великих обсягів даних і працює як розширення для Hadoop.

Модель даних:

- В основі лежить модель "ключ-значення" з підтримкою стовпцевого зберігання.
- Дані організовані у вигляді таблиць із можливістю зберігання різноманітних форматів у стовпцях.

Інтеграція з Hadoop:

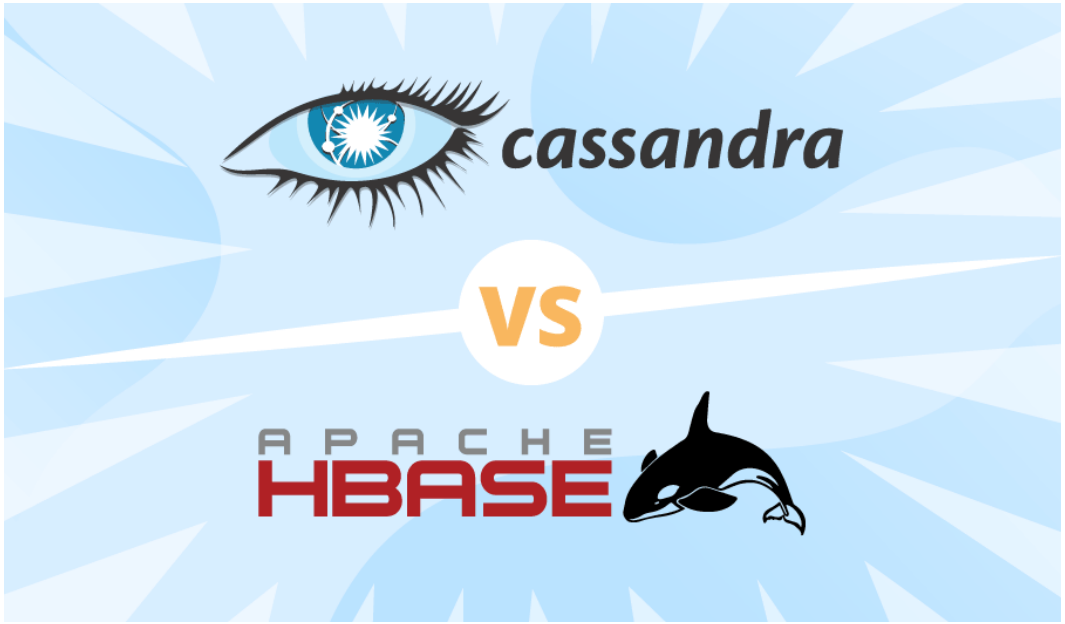
- Тісна інтеграція з екосистемою Hadoop дозволяє використовувати HBase для аналізу великих обсягів даних у поєднанні з MapReduce.

Консистентність:

- Підтримує Strong Consistency, що забезпечує строгість збереження даних.

Архітектура:

- Використовує майстер-слейв архітектуру з HMaster та регіон-серверами для управління даними.



Порівняння популярних NoSQL баз даних:
Cassandra vs Apache HBase

Порівняння Cassandra та Apache HBase

Масштабованість:

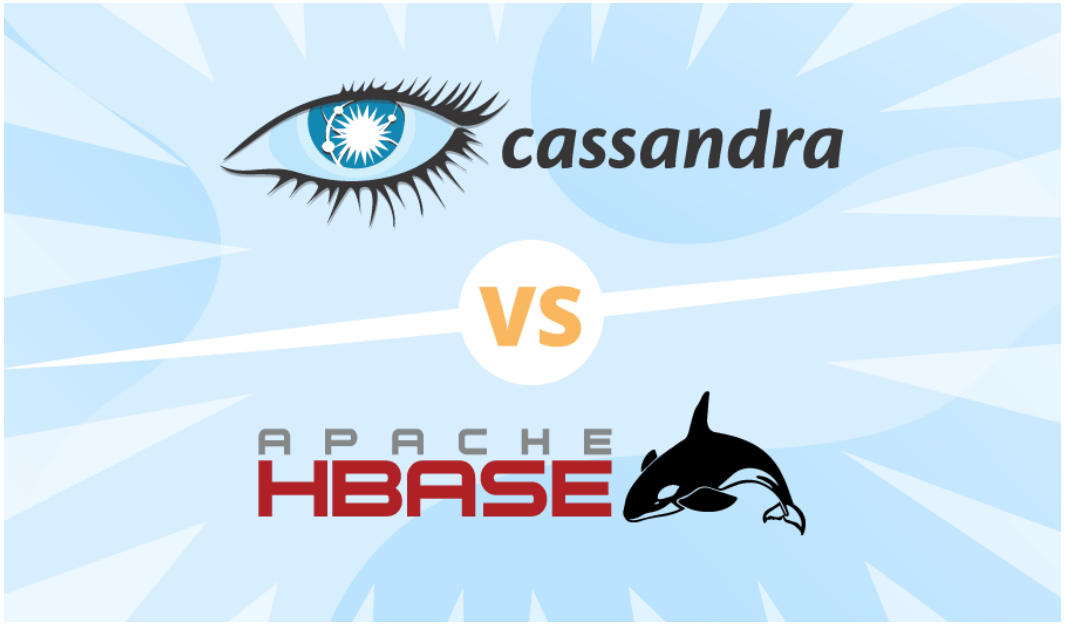
- Cassandra забезпечує повну горизонтальну масштабованість без простоїв, що робить її зручною для динамічного розширення кластерів.
- HBase масштабовується шляхом додавання нових регіон-серверів, але потребує централізованого управління через HMaster.

Консистентність і доступність:

- Cassandra дозволяє налаштувати баланс між консистентністю та доступністю (CAP-теорема), що є перевагою для розподілених систем.
- HBase забезпечує строгий контроль консистентності, але це може знижувати швидкість операцій у розподіленому середовищі.

Продуктивність:

- Cassandra добре працює з великими обсягами записів та читає дані дуже швидко завдяки відсутності центрального вузла.
- HBase ефективний для аналітичних запитів та інтеграції з Hadoop, але може мати більші затримки для операцій запису.



Порівняння популярних NoSQL баз даних:
Cassandra vs Apache HBase

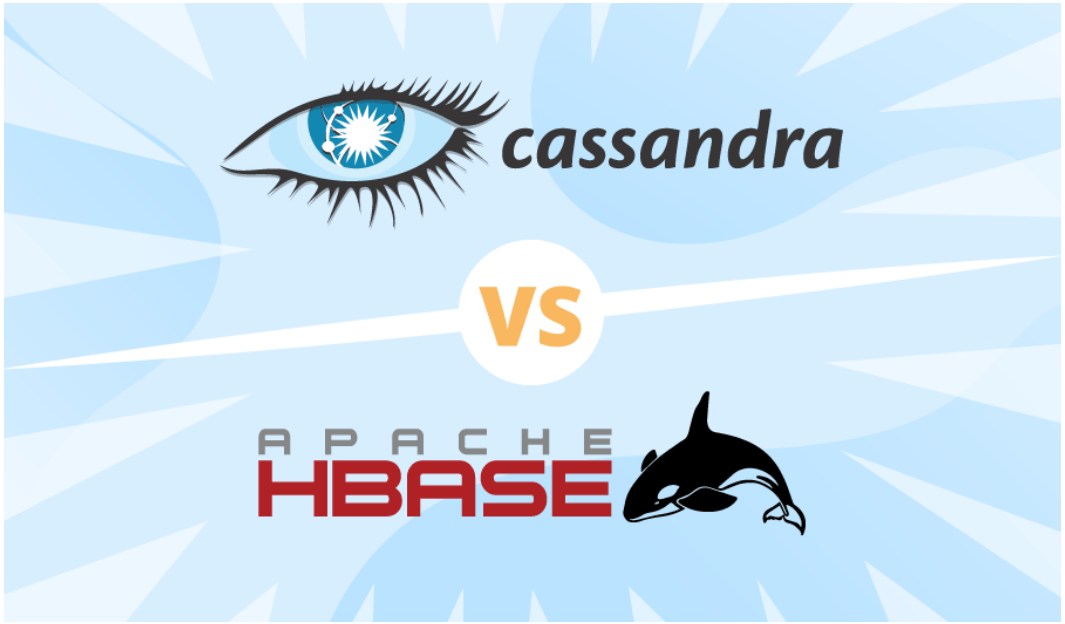
Порівняння Cassandra та Apache Hbase

Сфери використання:

- Cassandra часто використовується для зберігання даних у реальному часі, систем моніторингу, керування журналами та IoT-додатків.
- HBase ідеально підходить для аналітики, зберігання історичних даних та побудови великих архівів.

Простота налаштування та використання:

- Cassandra має простий механізм налаштування кластерів і не вимагає централізованого управління.
- HBase потребує більш складної інсталяції та конфігурації через інтеграцію з Hadoop.



Порівняння популярних NoSQL баз даних:
Cassandra vs Apache HBase

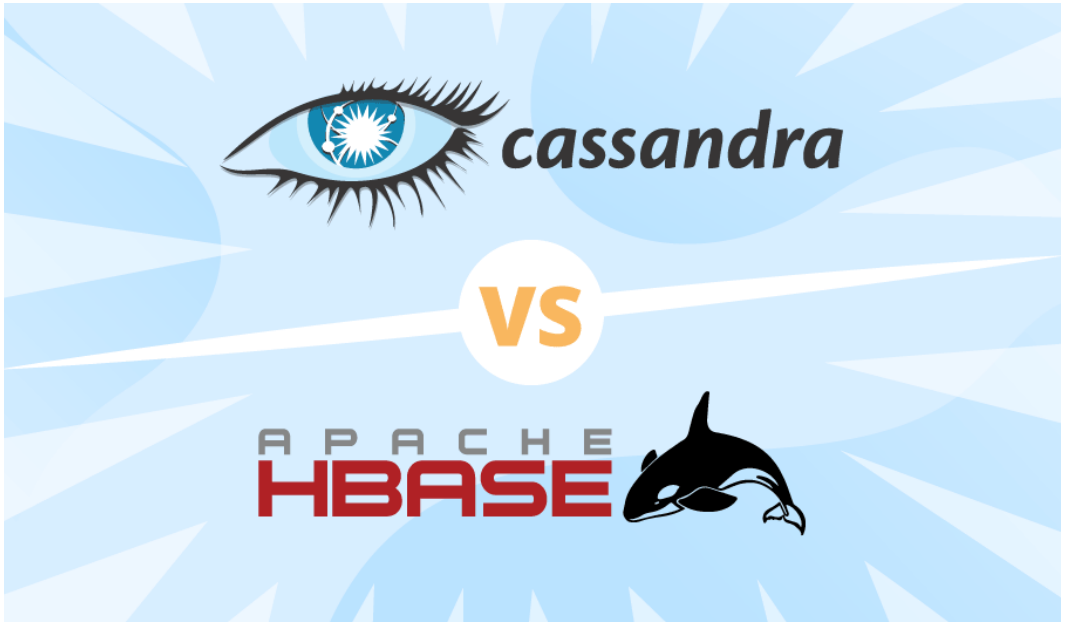
Області застосування

Cassandra:

- Інтернет-додатки з високим навантаженням.
- Збір та аналіз IoT-даних.
- Обробка транзакцій у реальному часі.

Apache HBase:

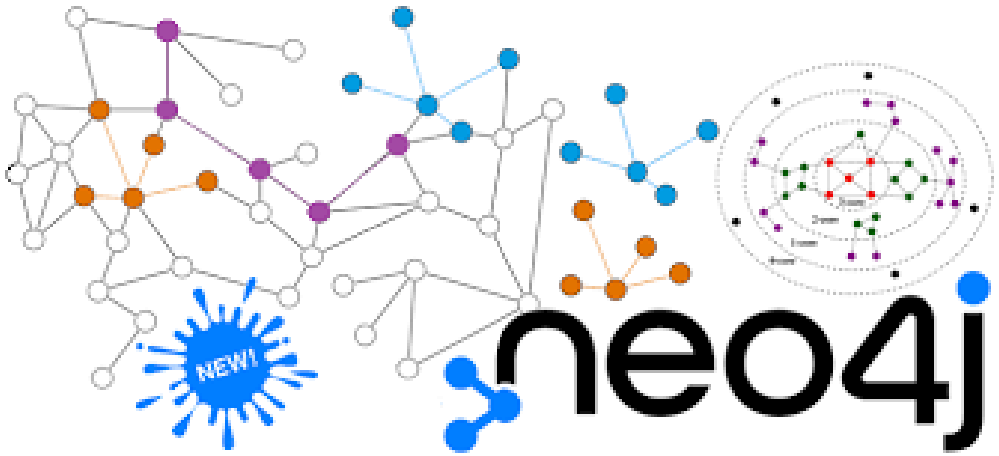
- Великі сховища даних із аналітикою.
- Проекти на основі Hadoop, які потребують глибокої інтеграції з MapReduce.
- Архівування великих обсягів історичних даних.



Порівняння популярних NoSQL баз даних:
Neo4j: переваги для роботи з графами

Основна ідея Neo4j

Neo4j — це графова база даних, яка зберігає дані у вигляді вузлів (nodes) та зв'язків (relationships). Вузли представляють об'єкти, а зв'язки — взаємозв'язки між ними. Графовий підхід забезпечує природне моделювання даних, що дозволяє ефективно працювати з мережами, ієрархіями, а також складними структурами.



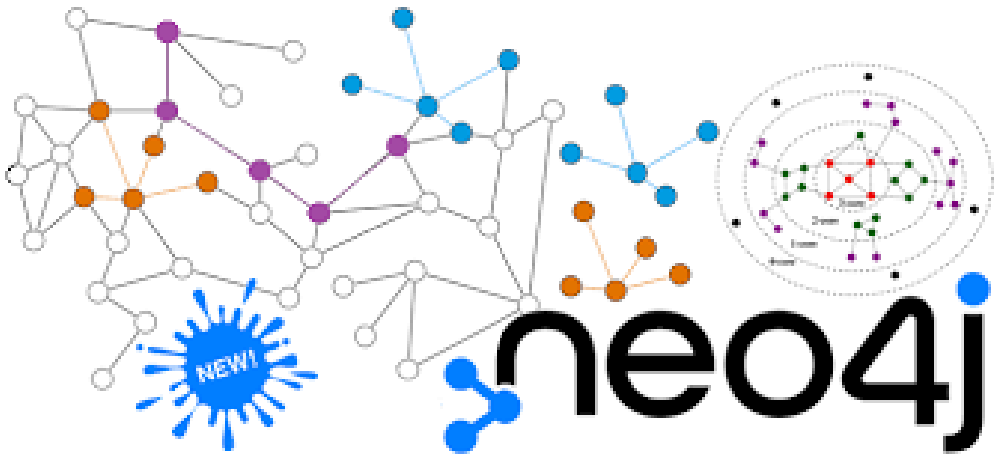
Порівняння популярних NoSQL баз даних:
Neo4j: переваги для роботи з графами

Переваги Neo4j для роботи з графами

Природне представлення даних. Графова структура даних у Neo4j дозволяє легко моделювати складні взаємозв'язки.

Висока продуктивність для графових запитів. Реляційні бази даних потребують виконання складних JOIN-операцій для отримання взаємозв'язків між таблицями. У Neo4j зв'язки є частиною самої моделі даних, тому запити виконуються значно швидше.

Гнучкість у моделюванні. У Neo4j немає жорстких схем, як у реляційних базах даних. Це дозволяє легко додавати нові типи вузлів і зв'язків без необхідності змінювати існуючу структуру.



Порівняння популярних NoSQL баз даних:
Neo4j: переваги для роботи з графами

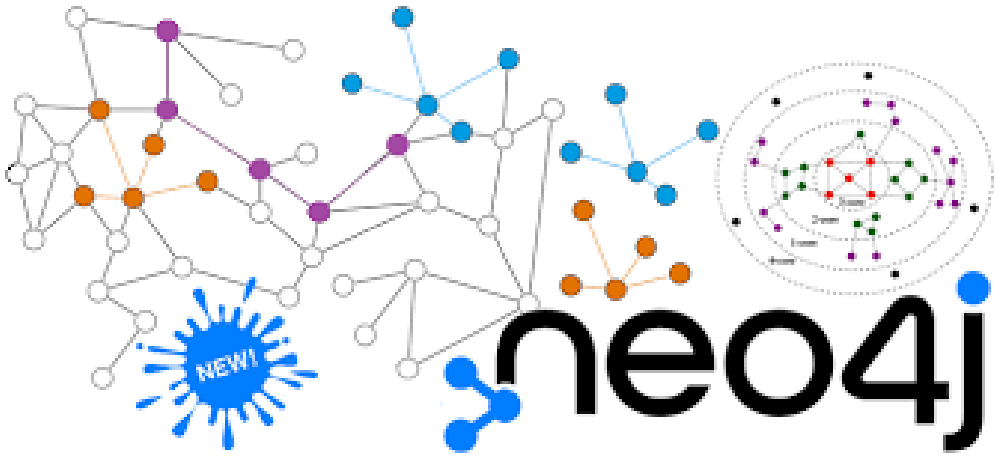
Переваги Neo4j для роботи з графами

Потужна мова запитів. Neo4j надає інтуїтивний синтаксис для роботи з графами. Вона дозволяє виконувати складні запити, зосереджуючись на логіці даних, а не на технічних аспектах.

Масштабованість. Neo4j підтримує горизонтальну та вертикальну масштабованість. Це означає, що база даних може обробляти великі обсяги даних і працювати в розподіленому середовищі.

Аналітика графів. Neo4j має вбудовані інструменти для аналізу графів, такі як пошук найкоротшого шляху, визначення центральності вузлів і пошук спільних груп.

Інтеграція з іншими системами. Neo4j легко інтегрується з популярними мовами програмування (Java, Python, JavaScript) та інструментами аналітики, такими як Apache Spark.



Порівняння популярних NoSQL баз даних:
Neo4j: переваги для роботи з графами

Приклади застосування Neo4j

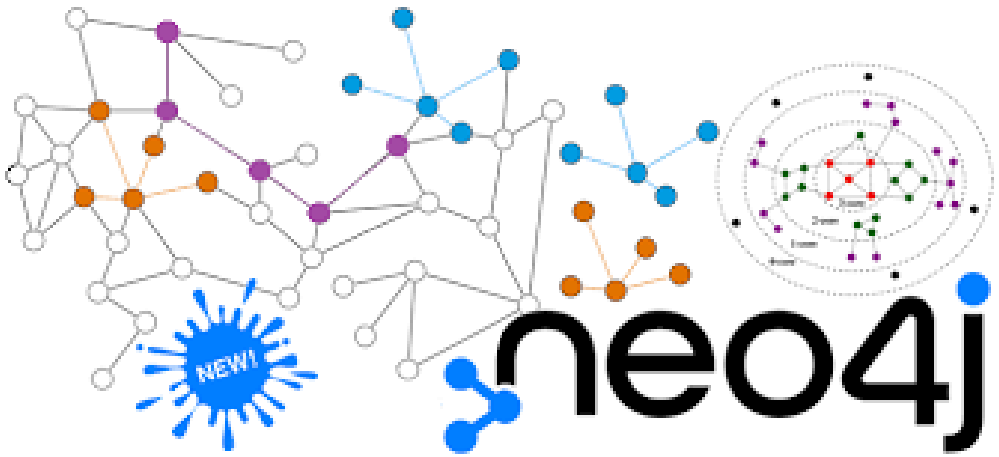
Соціальні мережі. Neo4j широко використовується для моделювання взаємозв'язків у соціальних мережах. Наприклад, Facebook чи LinkedIn можуть використовувати графи для рекомендацій друзів або аналізу взаємодій між користувачами.

Рекомендаційні системи. Завдяки здатності аналізувати складні зв'язки Neo4j використовується в e-commerce для створення персоналізованих рекомендацій.

Управління мережами. Neo4j застосовується для аналізу комп'ютерних або енергетичних мереж, відстеження збоїв і оптимізації потоків даних.

Фрод-моніторинг. У фінансових організаціях Neo4j використовується для виявлення шахрайських операцій шляхом аналізу транзакцій і пошуку аномальних шаблонів.

Біоінформатика. У дослідженнях геномів графова база даних допомагає аналізувати зв'язки між генами, білками та хворобами, що сприяє відкриттям у медицині.



Перспективи розвитку NoSQL баз даних

Коли обирати NoSQL базу даних замість реляційної?

Гнучкість схеми даних. Однією з головних причин для вибору NoSQL є відсутність жорсткої схеми (schema-less). Це дозволяє зберігати дані в довільних форматах, таких як JSON, BSON або XML

Масштабованість. Реляційні бази даних переважно підтримують вертикальну масштабованість, тобто збільшення потужності одного сервера. Це обмежує можливість обробляти великі обсяги даних без значних фінансових витрат. NoSQL бази, навпаки, розроблені для горизонтальної масштабованості.

Обробка великих обсягів даних. NoSQL бази створені для роботи з великими даними (Big Data). У багатьох системах, таких як соціальні мережі, електронна комерція або IoT-пристрої, обсяги даних можуть бути астрономічними, і реляційні бази можуть не впоратися з такими навантаженнями.

Перспективи розвитку NoSQL баз даних

Коли обирати NoSQL базу даних замість реляційної?

Робота з напівструктурованими та неструктурованими даними. У деяких сценаріях дані не підходять до табличного формату. Наприклад, графові бази даних, такі як Neo4j, забезпечують ефективне зберігання та обробку даних у вигляді вузлів і зв'язків.

Пріоритет продуктивності над узгодженістю. Для багатьох сучасних застосунків висока продуктивність є важливішою за повну узгодженість даних. NoSQL бази часто використовують модель eventual consistency (остаточна узгодженість), яка допускає тимчасову невідповідність даних у кластері, але гарантує, що вони стануть узгодженими згодом.

Використання спеціалізованих форматів зберігання. Реляційні бази даних призначені для роботи з табличними даними, тоді як NoSQL підтримують різні моделі: Ключ-значення (Redis, DynamoDB); Документно-орієнтовані бази (MongoDB, CouchDB); Стовпцеві бази (Cassandra, HBase); Графові бази (Neo4j)

Перспективи розвитку NoSQL баз даних

Коли обирати NoSQL базу даних замість реляційної?

7. *Складність і витрати на підтримку.* У проєктах, де важливо знизити витрати на адміністрування та підтримку бази даних, NoSQL може бути кращим вибором.

8. *Сценарії використання NoSQL баз даних.* NoSQL бази даних підходять для:

- **Веб-додатків** з великою кількістю користувачів, таких як соціальні мережі чи інтернет-магазини.
- **Аналітики великих даних**, де необхідна швидка обробка великих обсягів інформації.
- **IoT-додатків**, які генерують великі обсяги потокових даних.
- **Систем кешування** для швидкого доступу до часто використовуваних даних.
- **Реальних часів** систем, таких як обробка фінансових транзакцій чи ігрових платформ.

Перспективи розвитку NoSQL баз даних

Які виклики можуть виникнути при масштабуванні NoSQL баз даних?

1. *Складність розподіленого управління даними.* Одним із головних викликів при масштабуванні NoSQL баз даних є управління розподіленими даними. **Проблеми**:

- Консистентність даних: У розподілених системах важко забезпечити консистентність даних між вузлами.
- Реплікація: Реплікація даних між вузлами може бути складною, особливо якщо система працює в реальному часі.

Рішення:

- Використання моделей консистентності, таких як eventual consistency (консистентність у майбутньому) або strong consistency (строга консистентність), залежно від потреб застосування.
- Використання алгоритмів розподіленого консенсусу, таких як Paxos або Raft, для забезпечення узгодженості даних між вузлами.

Перспективи розвитку NoSQL баз даних

Які виклики можуть виникнути при масштабуванні NoSQL баз даних?

2. *Зростання складності архітектури.* Масштабування NoSQL баз даних часто вимагає додавання нових вузлів до кластера, що може значно ускладнити архітектуру системи. **Проблеми:**

- Мережеві затримки: Додавання нових вузлів може збільшити мережеві затримки, що впливає на продуктивність системи.
- Балансування навантаження: Неправильне балансування навантаження між вузлами може призвести до перевантаження окремих серверів, що знижує загальну продуктивність.

Рішення:

- Використання автоматизованих інструментів для балансування навантаження, таких як Kubernetes або Docker Swarm.
- Оптимізація мережевих налаштувань для зменшення затримок.

Перспективи розвитку NoSQL баз даних

Які виклики можуть виникнути при масштабуванні NoSQL баз даних?

3. *Обмеження продуктивності.* Масштабування NoSQL баз даних не завжди призводить до лінійного зростання продуктивності. Навпаки, додавання нових вузлів може викликати проблеми, пов'язані з обмеженнями продуктивності. **Проблеми:**

- Конкуренція за ресурси: Коли кількість вузлів зростає, може виникнути конкуренція за ресурси, такі як пам'ять, процесорний час та мережева пропускна здатність.
- Збільшення часу відгуку: У великих кластерах час відгуку може збільшуватися через необхідність синхронізації даних між вузлами.

Рішення:

- Використання кешування для зменшення навантаження на базу даних.
- Оптимізація запитів та індексації для підвищення продуктивності.

Перспективи розвитку NoSQL баз даних

Які виклики можуть виникнути при масштабуванні NoSQL баз даних?

4. *Забезпечення надійності та відмовостійкості.* Масштабування NoSQL баз даних вимагає забезпечення високої надійності та відмовостійкості. У розподілених системах відмова одного вузла може вплинути на всю систему. **Проблеми:**

- Відмови вузлів: Якщо один вузол виходить з ладу, це може призвести до втрати даних або зниження продуктивності.
- Реплікація та резервування: Неправильна настройка реплікації та резервування може призвести до втрати даних у разі відмови.

Рішення:

- Використання реплікації та резервування для забезпечення відмовостійкості.
- Регулярне тестування системи на відмовостійкість для виявлення потенційних проблем.

Перспективи розвитку NoSQL баз даних

Які виклики можуть виникнути при масштабуванні NoSQL баз даних?

5. *Складність управління схемами даних.* NoSQL бази даних відомі своєю гнучкістю у роботі зі схемами даних. Однак це також може стати викликом при масштабуванні, особливо якщо дані змінюються з часом. **Проблеми:**

- Еволюція схем даних: Зміни в структурі даних можуть вимагати складних міграцій, що впливає на продуктивність.
- Несумісність даних: Різні версії схем даних можуть співіснувати в системі, що ускладнює управління.

Рішення:

- Використання інструментів для управління міграціями схем даних.
- Регулярний аудит структури даних для забезпечення їх відповідності поточним потребам.

Перспективи розвитку NoSQL баз даних

Які виклики можуть виникнути при масштабуванні NoSQL баз даних?

6. *Висока вартість підтримки.* Масштабування NoSQL баз даних часто вимагає значних інвестицій у інфраструктуру та підтримку. Це може включати витрати на обладнання, програмне забезпечення та персонал. **Проблеми:**

- Витрати на обладнання: Додавання нових вузлів вимагає додаткових інвестицій у сервери та мережеве обладнання.
- Підтримка та адміністрування: Великі кластери потребують кваліфікованого персоналу для управління та підтримки.

Рішення:

- Використання хмарних рішень для зменшення витрат на обладнання.
- Автоматизація процесів адміністрування для зменшення навантаження на персонал.

Перспективи розвитку NoSQL баз даних

Як забезпечити узгодженість даних у NoSQL базах?

1. *Моделі консистентності в NoSQL базах даних*

Консистентність даних — це властивість системи, яка гарантує, що всі вузли бази даних мають однакове уявлення про дані в будь-який момент часу. У NoSQL базах даних існує кілька моделей консистентності, кожна з яких має свої переваги та недоліки. Вибір моделі залежить від вимог до продуктивності, доступності та надійності системи.

Перспективи розвитку NoSQL баз даних

Як забезпечити узгодженість даних у NoSQL базах?

1. *Моделі консистентності в NoSQL базах даних*

Консистентність даних — це властивість системи, яка гарантує, що всі вузли бази даних мають однакове уявлення про дані в будь-який момент часу. У NoSQL базах даних існує кілька моделей консистентності, кожна з яких має свої переваги та недоліки. Вибір моделі залежить від вимог до продуктивності, доступності та надійності системи.

Перспективи розвитку NoSQL баз даних

Як забезпечити узгодженість даних у NoSQL базах?

1. Моделі консистентності в NoSQL базах даних

а) *Строга консистентність* (Strong Consistency). Строга консистентність гарантує, що після оновлення даних усі наступні операції читання повертають останнє оновлене значення. Це означає, що система поводиться так, ніби вона працює на одному вузлі, навіть якщо дані розподілені між багатьма серверами.

Переваги:

- Гарантує, що дані завжди актуальні.
- Підходить для систем, де точність даних є критичною (наприклад, фінансові системи).

Недоліки:

- Може знижувати продуктивність через необхідність синхронізації між вузлами.
- Може призводити до збільшення часу відгуку.

Приклади реалізації:

- Google Spanner: Використовує строгу консистентність, забезпечуючи глобальну синхронізацію даних.
- MongoDB: Підтримує строгу консистентність за допомогою механізмів реплікації та транзакцій.

Перспективи розвитку NoSQL баз даних

Як забезпечити узгодженість даних у NoSQL базах?

1. Моделі консистентності в NoSQL базах даних

b) *Консистентність у майбутньому* (Eventual Consistency)

Консистентність у майбутньому дозволяє системі повертати дані, які можуть бути неактуальними на момент читання, але з часом стають консистентними. Це означає, що після оновлення даних може знадобитися час, щоб зміни поширилися на всі вузли.

Переваги:

- Висока продуктивність та доступність.
- Підходить для систем, де невелика затримка в оновленні даних є допустимою (наприклад, соціальні мережі).

Недоліки:

- Може призводити до несхожості даних на короткий час.
- Не підходить для систем, де точність даних є критичною.

Приклади реалізації:

- Cassandra: Використовує консистентність у майбутньому, забезпечуючи високу доступність та продуктивність.
- DynamoDB: Підтримує консистентність у майбутньому, дозволяючи налаштувати рівень консистентності для кожного запиту.

Перспективи розвитку NoSQL баз даних

Як забезпечити узгодженість даних у NoSQL базах?

1. Моделі консистентності в NoSQL базах даних

c. *Квазіконсистентність* (Quorum Consistency)

Квазіконсистентність є компромісом між строгою консистентністю та консистентністю у майбутньому. Вона гарантує, що дані будуть консистентними після отримання підтвердження від більшості вузлів.

Переваги:

- Забезпечує баланс між продуктивністю та консистентністю.
- Підходить для систем, де важливо зменшити ризик несхожості даних.

Недоліки:

- Вимагає більшої кількості операцій для забезпечення консистентності.
- Може збільшувати час відгуку.

Приклади реалізації:

- Riak: Використовує квазіконсистентність для забезпечення балансу між доступністю та консистентністю.

Перспективи розвитку NoSQL баз даних

Як забезпечити узгодженість даних у NoSQL базах?

2. *Механізми забезпечення узгодженості*

Для забезпечення узгодженості даних у NoSQL базах використовуються різні механізми, які дозволяють синхронізувати дані між вузлами та вирішувати конфлікти.

а) *Реплікація даних*

Реплікація — це процес копіювання даних між вузлами для забезпечення їх доступності та надійності. У NoSQL базах даних реплікація може бути синхронною або асинхронною.

- Синхронна реплікація: Дані оновлюються на всіх вузлах одночасно. Це забезпечує строгу консистентність, але може знижувати продуктивність.

- Асинхронна реплікація: Дані оновлюються на одному вузлі, а потім поширюються на інші. Це забезпечує високу продуктивність, але може призводити до несхожості даних.

Перспективи розвитку NoSQL баз даних

Як забезпечити узгодженість даних у NoSQL базах?

2. Механізми забезпечення узгодженості

b) *Конфлікти та їх вирішення*

У розподілених системах можуть виникати конфлікти, коли дані оновлюються на різних вузлах одночасно. Для вирішення конфліктів використовуються різні стратегії:

- Останнє оновлення перемагає (Last Write Wins): Використовується часова мітка для визначення останнього оновлення.

- Версіонування даних: Кожне оновлення даних зберігається з версією, що дозволяє вирішувати конфлікти шляхом злиття змін.

Перспективи розвитку NoSQL баз даних

Як забезпечити узгодженість даних у NoSQL базах?

2. Механізми забезпечення узгодженості

c) *Транзакції*

Деякі NoSQL бази даних, такі як MongoDB, підтримують транзакції для забезпечення атомарності операцій. Транзакції дозволяють групувати кілька операцій в одну логічну одиницю, що гарантує їх виконання або повне скасування.

Перспективи розвитку NoSQL баз даних

Як забезпечити узгодженість даних у NoSQL базах?

3. *Приклади забезпечення узгодженості в популярних NoSQL базах*

MongoDB підтримує строгу консистентність за допомогою реплікації та транзакцій. Реплікаційний набір MongoDB забезпечує синхронну реплікацію даних між вузлами, що гарантує консистентність.

Cassandra використовує консистентність у майбутньому та дозволяє налаштувати рівень консистентності для кожного запиту. Наприклад, можна вказати, скільки вузлів повинно підтвердити операцію, щоб вона вважалася успішною. Це дозволяє знаходити баланс між продуктивністю та консистентністю.

Riak використовує квазіконсистентність та механізми вирішення конфліктів, такі як версіонування даних. Це дозволяє забезпечити баланс між доступністю та консистентністю, що робить Riak ідеальним вибором для розподілених систем.

Перспективи розвитку NoSQL баз даних

Важливість вибору бази даних залежно від завдання

1. Різноманітність баз даних та їх призначення

Реляційні бази даних (SQL): Наприклад, MySQL, PostgreSQL, Oracle. Вони використовують таблиці для зберігання даних та підтримують мову SQL для запитів.

NoSQL бази даних: Наприклад, MongoDB, Cassandra, Redis. Вони пропонують гнучкість у зберіганні даних та підходять для роботи з неструктурованими або напівструктурованими даними.

Графові бази даних: Наприклад, Neo4j, ArangoDB. Вони орієнтовані на роботу зі складними взаємозв'язками між даними.

Стовпцеві бази даних: Наприклад, Apache HBase, Amazon Redshift. Вони оптимізовані для аналітичних запитів та обробки великих обсягів даних.

Ключ-значення бази даних: Наприклад, Redis, DynamoDB. Вони призначені для швидкого доступу до даних за ключем.

Перспективи розвитку NoSQL баз даних

Важливість вибору бази даних залежно від завдання

2. Фактори, які слід враховувати при виборі бази даних

a. *Тип даних та їх структура*
b. *Продуктивність та масштабованість*
c. *Консистентність та надійність*
d. *Екосистема та підтримка*

Перспективи розвитку NoSQL баз даних
Важливість вибору бази даних залежно від завдання

3. *Приклади вибору бази даних залежно від завдання*

a. *Соціальні мережі.* Соціальні мережі, такі як Facebook або Twitter, потребують обробки великих обсягів даних та складних взаємозв'язків між користувачами. Для таких завдань підходять графові бази даних, такі як Neo4j, які дозволяють ефективно моделювати взаємозв'язки та виконувати складні запити.

b. *Електронна комерція.* Електронні комерційні платформи, такі як Amazon або eBay, потребують швидкого доступу до даних про товари, користувачів та транзакції. Для таких завдань підходять реляційні бази даних, такі як MySQL або PostgreSQL, які забезпечують строгу консистентність та підтримку транзакцій.

c. *Аналітика великих даних.* Для аналітики великих даних, де важлива обробка великих обсягів інформації та виконання складних аналітичних запитів, підходять стовпцеві бази даних, такі як Apache HBase або Amazon Redshift. Вони оптимізовані для швидкого читання великих наборів даних.

d. *Інтернет речей* (IoT)/ У системах IoT, де дані генеруються великою кількістю пристроїв, важлива швидкість та масштабованість. Для таких завдань підходять NoSQL бази даних, такі як Cassandra або MongoDB, які дозволяють ефективно зберігати та обробляти великі обсяги даних.

Перспективи розвитку NoSQL баз даних
Важливість вибору бази даних залежно від завдання

4. *Помилки при виборі бази даних*

Неправильний вибір бази даних може призвести до серйозних проблем, таких як низька продуктивність, складність у підтримці або втрата даних.

Наприклад:

- Використання реляційної бази даних для зберігання неструктурованих даних може призвести до складних схем та низької продуктивності.
- Використання NoSQL бази даних для систем, де важлива строга консистентність, може призвести до несхожості даних.

Підсумки

- NoSQL — це нереляційна DMS, яка не потребує фіксованої схеми, уникає об’єднань і легко масштабується
- Концепція баз даних NoSQL стала популярною серед таких гігантів Інтернету, як Google, Facebook, Amazontощо, які мають справу з величезними обсягами даних
- У 1998 році Карло Строцці використав термін NoSQL для своєї легкої реляційної бази даних з відкритим кодом.
- Бази даних NoSQL ніколи не дотримуються реляційної моделі, вони або не містять схем, або мають розслаблені схеми
- Чотири типи баз даних NoSQL: 1). На основі пари ключ-значення 2). Стовпчастий графік 3). Графіки на основі 4). Документоорієнтований
- NOSQL може однаково ефективно обробляти структуровані, напівструктуровані та неструктуровані дані
- Теорема CAP складається з трьох слів Consistency, Availability і Partition Tolerance
- Термін «можлива узгодженість» означає наявність копій даних на кількох машинах для досягнення високої доступності та масштабованості
- NOSQL пропонують обмежені можливості запитів