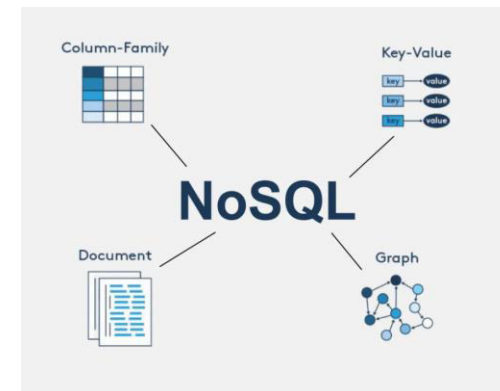




MÃ NGUỒN MỞ TRONG KHOA HỌC DỮ LIỆU


Bài 04. CƠ SỞ DỮ LIỆU NoSQL


ThS. Lê Nhật Tùng




NoSQL!

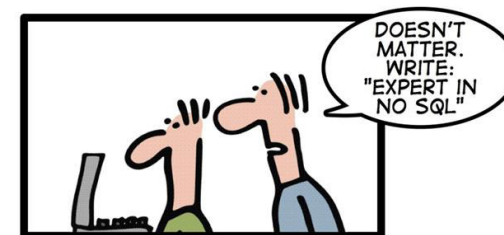
- Cơ sở dữ liệu NoSQL hiện đang là một chủ đề nóng trong một số lĩnh vực của ngành công nghệ thông tin, với hơn một trăm cơ sở dữ liệu NoSQL khác nhau.

**Backend Developer**
FamTech Technology., JSC
[Đăng nhập để xem mức lương](#) • Junior, Middle
Quận Thanh Xuân, Hà Nội (In Office)
[NodeJS](#) [MongoDB](#) [PostgreSql](#) Đăng 3 ngày trước

**Back-End Developer (Golang, MySQL)**
TrinTech
[Đăng nhập để xem mức lương](#) • Senior
Quận 1, Hồ Chí Minh (In Office)
[Linux](#) [MySQL](#) [MongoDB](#) [Golang](#) [Back-End](#) Đăng 1 tuần trước

**HCM/Hybrid - Senior Golang Developer_TS061802-Golang**
Talent Success
[Đăng nhập để xem mức lương](#) • Senior
Hồ Chí Minh (Hybrid)
[MongoDB](#) [Golang](#) [Postgre](#) Đăng 1 tuần trước

HOW TO WRITE A CV



Leverage the NoSQL boom

Đặc điểm của RDBMS

- Dữ liệu được lưu trữ trong các cột và bảng.
- Mỗi quan hệ được thể hiện thông qua dữ liệu.
- Ngôn ngữ thao tác dữ liệu (DML).
- Ngôn ngữ định nghĩa dữ liệu (DDL).
- Hỗ trợ giao dịch (Transactions). Tính trừu tượng khỏi lớp vật lý.
- Các ứng dụng chỉ xác định "cái gì", không phải "cách nào".
- Lớp vật lý có thể thay đổi mà không cần sửa đổi các ứng dụng.
- Tạo chỉ mục để hỗ trợ truy vấn.
- Cơ sở dữ liệu trong bộ nhớ (In-Memory databases).

Transactions – ACID Properties

- ***Tính nguyên tử (Atomic):*** Tất cả các bước trong một giao dịch phải hoàn thành (commit) hoặc không có bước nào được hoàn thành.

Ví dụ, khi chuyển tiền từ tài khoản này sang tài khoản khác, bạn phải thực hiện cả hai thao tác: rút tiền từ tài khoản đầu tiên và gửi tiền vào tài khoản thứ hai. Nếu thao tác gửi tiền không thành công, thì thao tác rút tiền cũng không nên diễn ra.

Transactions – ACID Properties

- ***Tính nhất quán (Consistent):*** Một giao dịch phải đảm bảo rằng cơ sở dữ liệu chuyển từ một trạng thái hợp lệ sang một trạng thái hợp lệ khác. Tính nhất quán được đảm bảo bởi các ràng buộc của cơ sở dữ liệu.

Ví dụ, trong một cơ sở dữ liệu theo dõi tài khoản séc, mỗi giao dịch chỉ được phép có một số séc duy nhất.

Transactions – ACID Properties

- ***Tính cách ly (Isolated)***: Kết quả của các thay đổi trong một giao dịch không được hiển thị cho đến khi giao dịch đó đã được hoàn tất.

Ví dụ, khi một giao dịch viên kiểm tra số dư tài khoản, họ phải được cách ly khỏi một giao dịch khác đang rút tiền từ cùng tài khoản đó. Chỉ khi giao dịch rút tiền hoàn thành và giao dịch viên kiểm tra số dư lại, số dư mới sẽ được cập nhật.

Transactions – ACID Properties

- ***Tính bền vững (Durable):*** Kết quả của một giao dịch đã được hoàn tất sẽ tồn tại ngay cả khi có sự cố xảy ra. Điều này có nghĩa là hệ thống không được phép mất kết quả của giao dịch hoặc dữ liệu trong cơ sở dữ liệu. Tính bền vững thường được đảm bảo thông qua các nhật ký giao dịch, cho phép "tái tạo" lại các giao dịch từ một thời điểm cụ thể (giống như sao lưu).

No SQL là gì?

NoSQL là viết tắt của:

- No Relational: Không quan hệ.
- No RDBMS: Không phải RDBMS (Hệ quản trị cơ sở dữ liệu quan hệ).
- Not Only SQL: Không chỉ là SQL.

NoSQL là một thuật ngữ chung cho tất cả các loại cơ sở dữ liệu và kho dữ liệu không tuân theo các nguyên tắc của RDBMS.

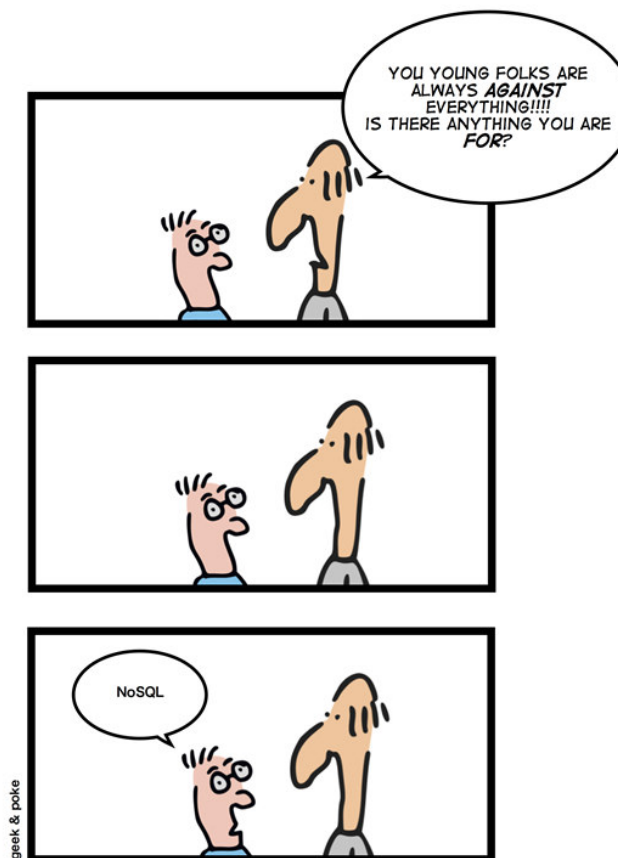
- Đây là một nhóm sản phẩm.
- Nó bao gồm một tập hợp các khái niệm liên quan về lưu trữ và thao tác dữ liệu.
- Thường liên quan đến các tập dữ liệu lớn.

Định nghĩa về NoSQL

From www.nosql-database.org:

Cơ sở dữ liệu thế hệ tiếp theo chủ yếu giải quyết một số điểm: không quan hệ, phân tán, mã nguồn mở và có thể mở rộng theo chiều ngang. Ý định ban đầu là cơ sở dữ liệu quy mô web hiện đại.

Phong trào này bắt đầu vào đầu năm 2009 và đang phát triển nhanh chóng. Thường có nhiều đặc điểm hơn được áp dụng như: không có lược đồ, hỗ trợ sao chép dễ dàng, API đơn giản, cuối cùng là nhất quán / BASE (không phải ACID), lượng dữ liệu khổng lồ và nhiều hơn nữa.



Định nghĩa về NoSQL

Một số đặc điểm thường thấy ở các cơ sở dữ liệu thể hệ tiếp theo bao gồm:

- **Không có schema (schema-free):** Không yêu cầu định nghĩa cấu trúc dữ liệu cố định trước.
- **Hỗ trợ sao chép dễ dàng:** Cho phép sao chép dữ liệu một cách nhanh chóng và hiệu quả.
- **API đơn giản:** Cung cấp giao diện lập trình ứng dụng dễ sử dụng.
- **Tính nhất quán cuối cùng (eventually consistent) / BASE:** Khác với các giao dịch ACID truyền thống, các hệ thống NoSQL thường sử dụng mô hình BASE, cho phép độ nhất quán dữ liệu đạt được sau một thời gian.
- **Khả năng xử lý khối lượng dữ liệu lớn:** Được thiết kế để quản lý và lưu trữ lượng dữ liệu khổng lồ.

NoSQL bắt nguồn từ đâu?

- Hệ quản trị cơ sở dữ liệu phi quan hệ (Non-relational DBMS) không phải là một khái niệm mới, nhưng NoSQL đại diện cho một hình thức mới của nó. Sự phát triển này diễn ra chủ yếu nhờ vào nhu cầu của các ứng dụng Internet quy mô lớn, dựa trên các mô hình tính toán phân tán và song song.

Lịch sử phát triển

- Bắt đầu với Google: Công ty này đã tiên phong trong việc phát triển các công nghệ hỗ trợ cho NoSQL.
- Bài báo nghiên cứu đầu tiên được công bố vào năm 2003, đánh dấu bước khởi đầu cho phong trào NoSQL.
- Tiếp nối phát triển nhờ vào các dự án như Lucene của Apache (Hadoop) và Amazon (Dynamo), tạo nền tảng cho nhiều ứng dụng NoSQL.
- Nhiều sản phẩm và sự quan tâm đã xuất hiện từ các công ty lớn như Facebook, Netflix, Yahoo, eBay, Hulu, IBM và nhiều công ty khác.

NoSQL và Big Data

- NoSQL xuất phát từ Internet, vì vậy nó thường liên quan đến khái niệm **"dữ liệu lớn"** (big data).
- Dữ liệu lớn là gì?
 - Khối lượng: Dữ liệu lớn thường được định nghĩa là các tập dữ liệu có kích thước từ vài terabyte trở lên, đủ lớn để cần phải sử dụng nhiều đơn vị lưu trữ khác nhau.

NoSQL và Big Data

- Những thách thức của dữ liệu lớn:
 - **Lưu trữ và truy xuất dữ liệu hiệu quả:** Khó khăn trong việc lưu trữ và truy xuất một lượng lớn dữ liệu, đồng thời đảm bảo tính chịu lỗi và khả năng sao lưu.
 - **Xử lý các tập dữ liệu lớn:** Cần thực hiện các quá trình song song rất lớn để thao tác với dữ liệu, yêu cầu tối ưu hóa và quản lý tài nguyên hiệu quả.
 - **Quản lý schema và metadata liên tục:** Khó khăn trong việc quản lý cấu trúc dữ liệu và siêu dữ liệu cho dữ liệu bán cấu trúc và phi cấu trúc, đặc biệt khi yêu cầu thay đổi cấu trúc thường xuyên.

Nguồn gốc của dữ liệu lớn hiện nay?

Bùng nổ các trang mạng xã hội: Sự phát triển mạnh mẽ của các trang mạng xã hội như Facebook và Twitter đã tạo ra nhu cầu lưu trữ và xử lý dữ liệu lớn.

Sự gia tăng của các giải pháp dựa trên đám mây: Các dịch vụ lưu trữ dựa trên đám mây như Amazon S3 (Simple Storage Service) đã cung cấp khả năng lưu trữ linh hoạt và mở rộng cho khối lượng dữ liệu khổng lồ.

Chuyển dịch sang dữ liệu có kiểu động: Tương tự như việc chuyển sang các ngôn ngữ lập trình kiểu động như Python, Ruby và Groovy, chúng ta cũng đang chứng kiến một sự chuyển dịch sang dữ liệu có kiểu động với sự thay đổi cấu trúc thường xuyên.

Cộng đồng mã nguồn mở: Sự phát triển của cộng đồng mã nguồn mở đã thúc đẩy việc phát triển và chia sẻ các công nghệ mới, hỗ trợ sự ra đời của các giải pháp NoSQL để quản lý dữ liệu lớn.

Tại sao RDBMS không phù hợp cho Dữ liệu lớn (Big Data)?

- Giả định về dữ liệu:
 - **Dữ liệu dày đặc (Dense Data):** RDBMS giả định rằng dữ liệu được lưu trữ một cách dày đặc, nghĩa là mỗi bảng có nhiều bản ghi và sử dụng không gian lưu trữ hiệu quả.
 - **Dữ liệu đồng nhất (Uniform Data):** RDBMS thường chỉ xử lý dữ liệu có cấu trúc (Structured Data), yêu cầu một cấu trúc cố định (schema) trước khi lưu trữ.

Tại sao RDBMS không phù hợp cho Dữ liệu lớn (Big Data)?

- Dữ liệu từ Internet:
 - **Khối lượng lớn và thưa thớt (Massive and Sparse Data):** Dữ liệu từ Internet thường rất lớn nhưng không dày đặc, với nhiều bản ghi thiếu dữ liệu hoặc không liên quan đến nhau.
 - **Dữ liệu bán cấu trúc hoặc phi cấu trúc (Semi-structured or Unstructured Data):** Dữ liệu này thường không tuân theo một cấu trúc cố định, ví dụ như dữ liệu JSON, XML, hay các định dạng văn bản, hình ảnh.

Tại sao RDBMS không phù hợp cho Dữ liệu lớn (Big Data)?

- Giới hạn của RDBMS:
 - Khi phải xử lý các tập dữ liệu lớn và thừa thớt, các phương pháp lưu trữ và truy xuất thông thường của RDBMS không còn hiệu quả, dẫn đến khó khăn trong việc quản lý và khai thác dữ liệu.

Đặc điểm nổi bật của NoSQL

- Khối lượng dữ liệu lớn:
 - NoSQL được thiết kế để xử lý các tập dữ liệu khổng lồ, như khái niệm “dữ liệu lớn” (big data) của Google.
- Khả năng mở rộng và phân phối:
 - Hệ thống NoSQL có thể mở rộng dễ dàng, có khả năng chạy trên hàng nghìn máy chủ.
 - Dữ liệu có thể được phân phối trên nhiều máy chủ nằm rải rác trên toàn cầu.
- Truy vấn nhanh:
 - NoSQL được tối ưu hóa để trả về kết quả truy vấn một cách nhanh chóng.
 - Chủ yếu tập trung vào các truy vấn (queries), trong khi số lượng cập nhật (updates) ít hơn.

Đặc điểm nổi bật của NoSQL

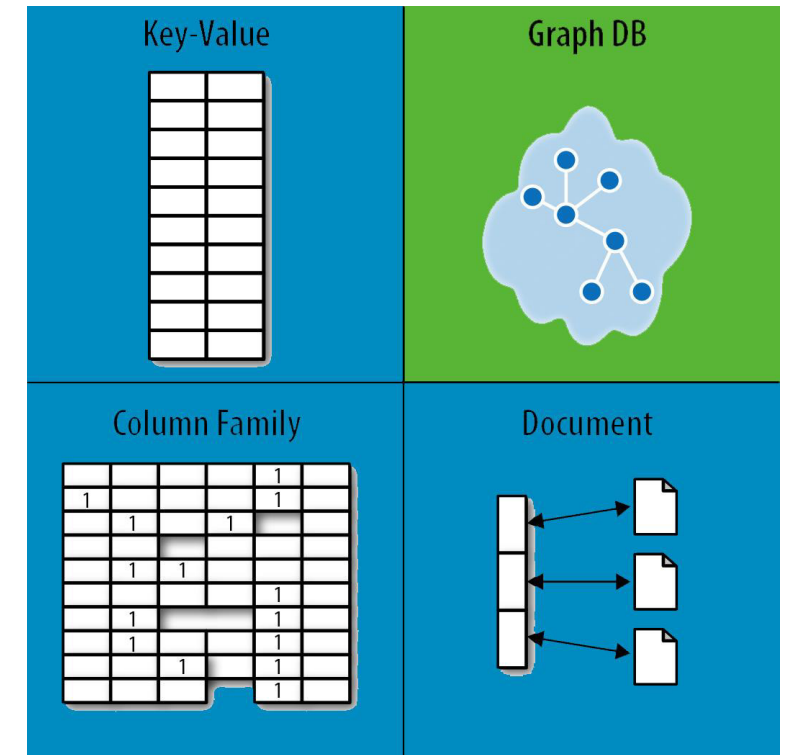
- Chèn và cập nhật không đồng bộ:
 - Hỗ trợ các hoạt động chèn và cập nhật dữ liệu không đồng bộ, cho phép cải thiện hiệu suất.
- Không cần cấu trúc (Schema-less):
 - NoSQL cho phép lưu trữ dữ liệu mà không yêu cầu một cấu trúc cố định, mang lại sự linh hoạt trong việc quản lý dữ liệu.
- Không cần thuộc tính giao dịch ACID – Chấp nhận BASE:
 - Thay vì đảm bảo các thuộc tính giao dịch ACID (Atomicity, Consistency, Isolation, Durability), NoSQL chấp nhận mô hình BASE (Basically Available, Soft state, Eventually consistent).

Đặc điểm nổi bật của NoSQL

- Định lý CAP:
 - NoSQL thường hoạt động trong khuôn khổ của định lý CAP, cân bằng giữa tính nhất quán (Consistency), tính khả dụng (Availability), và tính chịu phân vùng (Partition tolerance).
- Phát triển mã nguồn mở:
 - Nhiều giải pháp NoSQL được phát triển dưới dạng mã nguồn mở, cho phép cộng đồng tham gia vào việc cải tiến và phát triển công nghệ.

Các loại cơ sở dữ liệu NoSQL

- **Cơ sở dữ liệu Key-Value:** Lưu trữ dữ liệu dưới dạng cặp khóa-giá trị, phù hợp cho các ứng dụng cần truy cập nhanh.
- **Cơ sở dữ liệu Document:** Lưu trữ dữ liệu dưới dạng tài liệu (documents), thường là JSON hoặc XML, cho phép linh hoạt trong cấu trúc dữ liệu.
- **Cơ sở dữ liệu Column-Family:** Lưu trữ dữ liệu theo cột, tối ưu hóa cho các truy vấn có tính toán trên các cột cụ thể.
- **Cơ sở dữ liệu Graph:** Tập trung vào việc lưu trữ và truy vấn dữ liệu liên quan đến mối quan hệ, lý tưởng cho các ứng dụng như mạng xã hội.



Cơ sở dữ liệu tài liệu (Document Databases)

- **Documents**

- Tài liệu bao gồm các tập hợp key/value được cấu trúc lỏng lẻo, chẳng hạn như XML, JSON, hoặc BSON.
 - Dữ liệu được bao bọc và mã hóa theo các định dạng hoặc mã hóa tiêu chuẩn.
 - Tài liệu được truy cập trong cơ sở dữ liệu thông qua một khóa duy nhất.
 - Tài liệu được coi là một khối thống nhất, không tách rời thành các cặp tên/giá trị riêng lẻ.
- Cho phép truy vấn tài liệu bằng cách sử dụng khóa hoặc nội dung của tài liệu.
 - Một số cơ sở dữ liệu dạng Documents:
 - **MongoDB** (FourSquare, Github, ... đang sử dụng)
 - **CouchDB** (Apple, BBC, Canonical, Cern, ... đang sử dụng)

Cơ sở dữ liệu tài liệu (Document Databases)

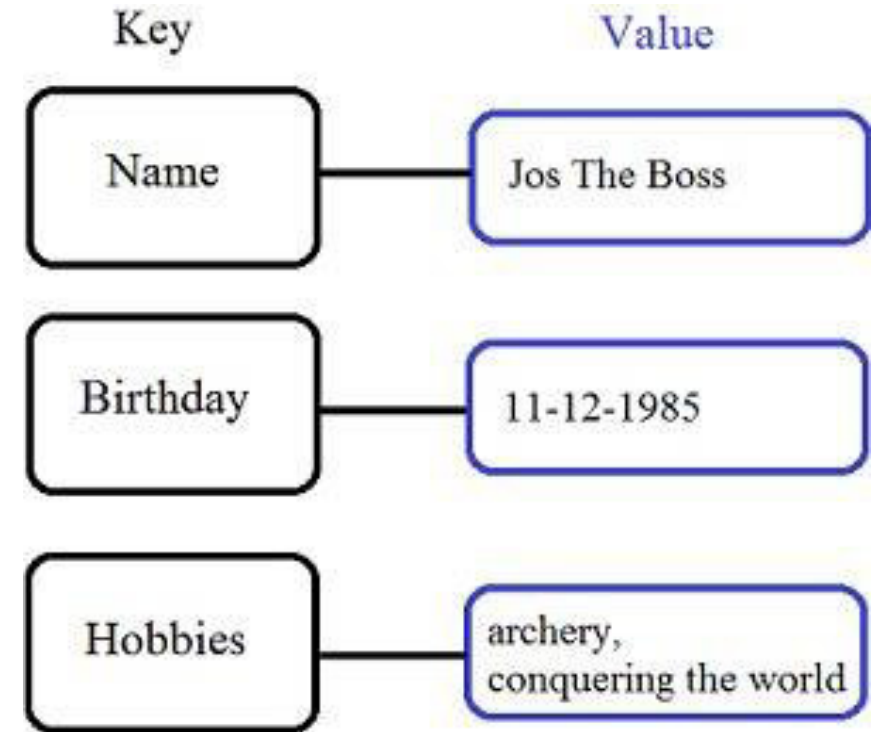
- Trong cơ sở dữ liệu NoSQL, tài liệu là đơn vị dữ liệu cơ bản, tương đương với một hàng trong cơ sở dữ liệu quan hệ.
- Mỗi tài liệu được cấu trúc dưới dạng các cặp khóa-giá trị, linh hoạt và không tuân theo một schema cố định như trong các cơ sở dữ liệu quan hệ.
- Tài liệu thường được lưu trữ ở các định dạng tiêu chuẩn như JSON hoặc BSON, cho phép biểu diễn dữ liệu có cấu trúc phức tạp một cách hiệu quả.

Document Databases, JSON

```
{
  "_id": ObjectId("51156a1e056d6f966f268f81"),
  "type": "Bài viết",
  "author": "Le Nhat Tung",
  "title": "Giới thiệu về cơ sở dữ liệu tài liệu với MongoDB",
  "date": ISODate("2024-04-24T16:26:31.911Z"),
  "body": "Bài viết này..."
},
{
  "_id": ObjectId("51156a1e056d6f966f268f82"),
  "type": "Sách",
  "author": "Le Nhat Tung",
  "title": "Hướng dẫn của php|architect về lập trình ngày và giờ với PHP",
  "isbn": "999-0-9738621-5-9"
}
```

CSDL dạng Key/Value

- Cơ sở dữ liệu NoSQL Key-Value là một loại hệ thống lưu trữ dữ liệu mà trong đó dữ liệu được lưu trữ dưới dạng các cặp khóa-giá trị. Mỗi khóa (key) là duy nhất và được sử dụng để truy cập giá trị (value) tương ứng.



CSDL dạng Key/Value

- Dữ liệu được lưu trữ theo cách rất đơn giản, cho phép truy cập nhanh và hiệu quả.
- Giá trị có thể là bất kỳ kiểu dữ liệu nào, từ chuỗi (string), số nguyên (integer), đến các cấu trúc dữ liệu phức tạp hơn như danh sách hoặc bản đồ.
- Cơ sở dữ liệu Key-Value được tối ưu hóa cho tốc độ, thường cho phép truy cập nhanh hơn so với các loại cơ sở dữ liệu khác.
- Nhiều hệ thống Key-Value có khả năng phân phối, cho phép dữ liệu được lưu trữ trên nhiều máy chủ và dễ dàng mở rộng.

CSDL dạng Key/Value

- Các hệ thống này thường thiết kế để đảm bảo khả năng phục hồi và khả dụng cao, ngay cả khi có sự cố với một hoặc nhiều máy chủ.
- Các cơ sở dữ liệu Key-Value thường được sử dụng trong các ứng dụng cần lưu trữ và truy cập dữ liệu nhanh chóng, như cache, lưu trữ phiên người dùng, và quản lý cấu hình.

Ví dụ: Một số hệ thống cơ sở dữ liệu Key-Value phổ biến bao gồm Redis, Amazon DynamoDB, và Riak.

Cơ sở dữ liệu định hướng cột có sắp xếp

•**Lưu trữ dữ liệu theo cột:** Dữ liệu được tổ chức và lưu trữ theo cột thay vì theo hàng như trong các cơ sở dữ liệu quan hệ truyền thống.

•**Tối ưu hóa không gian:**

•**Tránh lãng phí không gian cho các giá trị null:** Chỉ lưu trữ các giá trị không null, giúp tiết kiệm dung lượng.

•**Nén dữ liệu hiệu quả:** Do các giá trị trong cùng một cột thường có cùng kiểu dữ liệu, nên việc nén dữ liệu trở nên hiệu quả hơn.

•**Tổ chức dữ liệu theo nhóm cột (column families):** Các cột có liên quan với nhau được nhóm lại thành các "gia đình cột" để quản lý và truy vấn dễ dàng hơn.

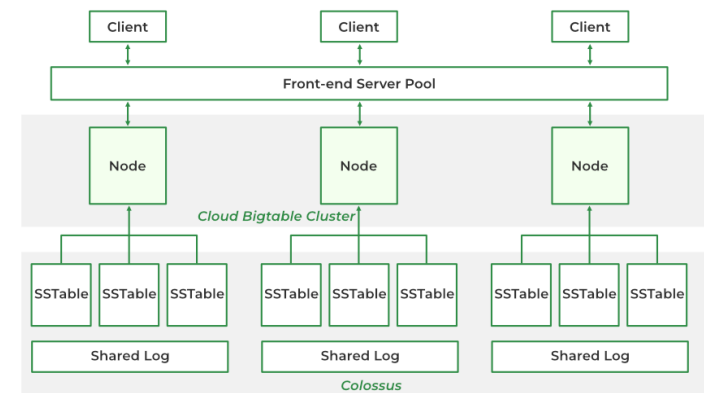


Cơ sở dữ liệu định hướng cột có sắp xếp

Ví dụ:

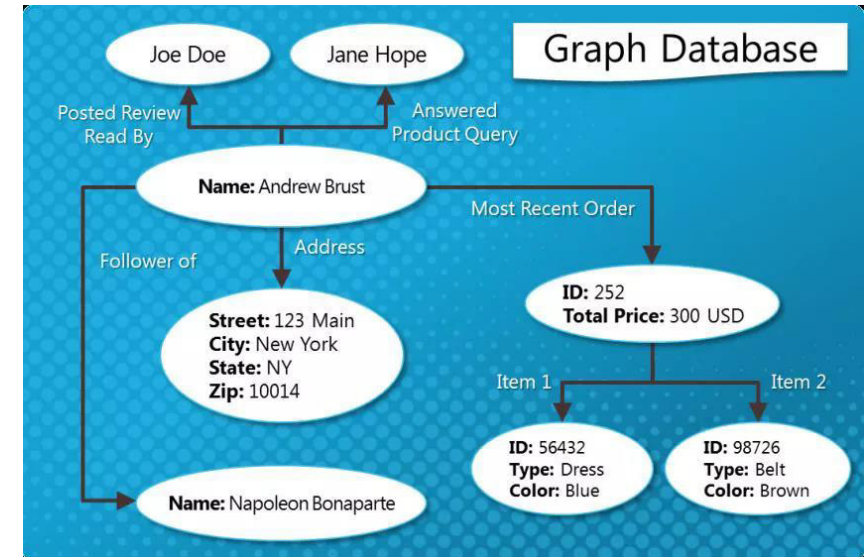
Google's Bigtable: Được sử dụng rộng rãi trong các dịch vụ của Google để lưu trữ và xử lý lượng dữ liệu khổng lồ.

HBase: Được phát triển bởi Apache Software Foundation, được sử dụng bởi nhiều công ty lớn như Facebook, StumbleUpon, Hulu, Yahoo! để xây dựng các hệ thống phân tán lớn.



Graph Databases

- Hướng đồ thị
- Mọi thứ được lưu trữ dưới dạng một cạnh (edge), một nút (node) hoặc một thuộc tính (attribute).
- Mỗi nút và cạnh có thể có bất kỳ số lượng thuộc tính nào.
- Cả nút và cạnh đều có thể được gán nhãn (label).
- Nhãn có thể được sử dụng để thu hẹp các tìm kiếm.



Ứng dụng thực tế của cơ sở dữ liệu dạng đồ thị

- Mạng xã hội: Cơ sở dữ liệu đồ thị được sử dụng để lưu trữ và quản lý các mối quan hệ giữa người dùng, bài viết, và các tương tác trong các nền tảng mạng xã hội như Facebook và LinkedIn.
- Hệ thống khuyến nghị: Nhiều hệ thống khuyến nghị, như Netflix và Amazon, sử dụng cơ sở dữ liệu đồ thị để phân tích mối quan hệ giữa người dùng, sản phẩm và các yếu tố khác nhằm cung cấp đề xuất chính xác hơn.
- Phân tích dữ liệu lớn: Cơ sở dữ liệu đồ thị được áp dụng trong phân tích dữ liệu lớn để tìm ra các mẫu và mối quan hệ ẩn giữa các yếu tố khác nhau trong một tập dữ liệu lớn.

Ứng dụng thực tế của cơ sở dữ liệu dạng đồ thị

- Quản lý chuỗi cung ứng: Cơ sở dữ liệu đồ thị có thể được sử dụng để mô hình hóa và tối ưu hóa các mối quan hệ trong chuỗi cung ứng, giúp theo dõi hàng hóa, nhà cung cấp, và khách hàng.
- Phát hiện gian lận: Các tổ chức tài chính sử dụng cơ sở dữ liệu đồ thị để phát hiện các mẫu giao dịch bất thường và mối quan hệ gian lận giữa các tài khoản.
- Quản lý hạ tầng mạng: Cơ sở dữ liệu đồ thị có thể được sử dụng để theo dõi và quản lý các mối quan hệ giữa các thiết bị trong một mạng lưới, giúp tối ưu hóa hiệu suất và bảo mật mạng.

Xử lý Dữ liệu Lớn và Khả năng Mở rộng

- Hệ quản trị cơ sở dữ liệu quan hệ (RDBMS) không được thiết kế để phân tán.
- Các hệ quản trị cơ sở dữ liệu truyền thống được thiết kế tối ưu để chạy trên một máy duy nhất.
- Khi khối lượng dữ liệu hoặc số lượng thao tác lớn hơn, cần nâng cấp máy chủ với CPU nhanh hơn hoặc thêm bộ nhớ, được gọi là "mở rộng quy mô dọc" (vertical scaling).
- Giải pháp NoSQL được thiết kế để chạy trên các cụm máy chủ (cluster) hoặc các hệ cơ sở dữ liệu nhiều node.
- Khi khối lượng dữ liệu hoặc số lượng thao tác lớn hơn, cần thêm nhiều máy vào cụm, được gọi là "mở rộng quy mô ngang" (horizontal scaling).

Xử lý Dữ liệu Lớn và Khả năng Mở rộng

- Các cách tiếp cận khác nhau bao gồm:
 - Mô hình chủ - tớ (Master-slave)
 - Phân mảnh (Sharding)

Scaling RDBMS

- Master-Slave
 - Trong mô hình Master-Slave, tất cả các thao tác ghi (writes) được thực hiện trên máy chủ chính (master).
 - Sau đó, dữ liệu sẽ được nhân bản và sao chép tới các máy chủ phụ (slaves).
 - Các thao tác đọc (reads) được thực hiện trên các máy chủ phụ, giúp giảm tải cho máy chủ chính và tăng khả năng đáp ứng cho hệ thống.
- Sharding
 - Khi một cơ sở dữ liệu được phân tán trên nhiều máy (nodes), điều quan trọng là phải biết dữ liệu nào được lưu trữ trên máy nào. Hệ thống cần xác định vị trí chính xác của một đoạn dữ liệu để truy vấn hoặc cập nhật, nếu không, quá trình truy xuất sẽ bị chậm hoặc không hiệu quả.
 - Để giải quyết vấn đề này, một hệ thống sharding (phân mảnh dữ liệu) được sử dụng, nơi dữ liệu được chia thành các mảnh nhỏ hơn (shards) và lưu trữ trên các máy khác nhau.

NoSQL, No ACID

- Nhiều cơ sở dữ liệu NoSQL không tuân theo đầy đủ các thuộc tính ACID, thay vào đó, họ áp dụng nguyên tắc **BASE** để ưu tiên khả năng mở rộng và hiệu suất: **Basically Available (Khả dụng cơ bản)**: Hệ thống sẽ luôn phản hồi (dù có thể trả về kết quả không đầy đủ).
- **Soft state (Trạng thái mềm)**: Trạng thái của hệ thống có thể thay đổi theo thời gian, ngay cả khi không có đầu vào thêm.
- **Eventual consistency (Nhất quán dần dần)**: Dữ liệu cuối cùng sẽ nhất quán sau một khoảng thời gian nào đó, nhưng không phải ngay lập tức.

BASE Transactions

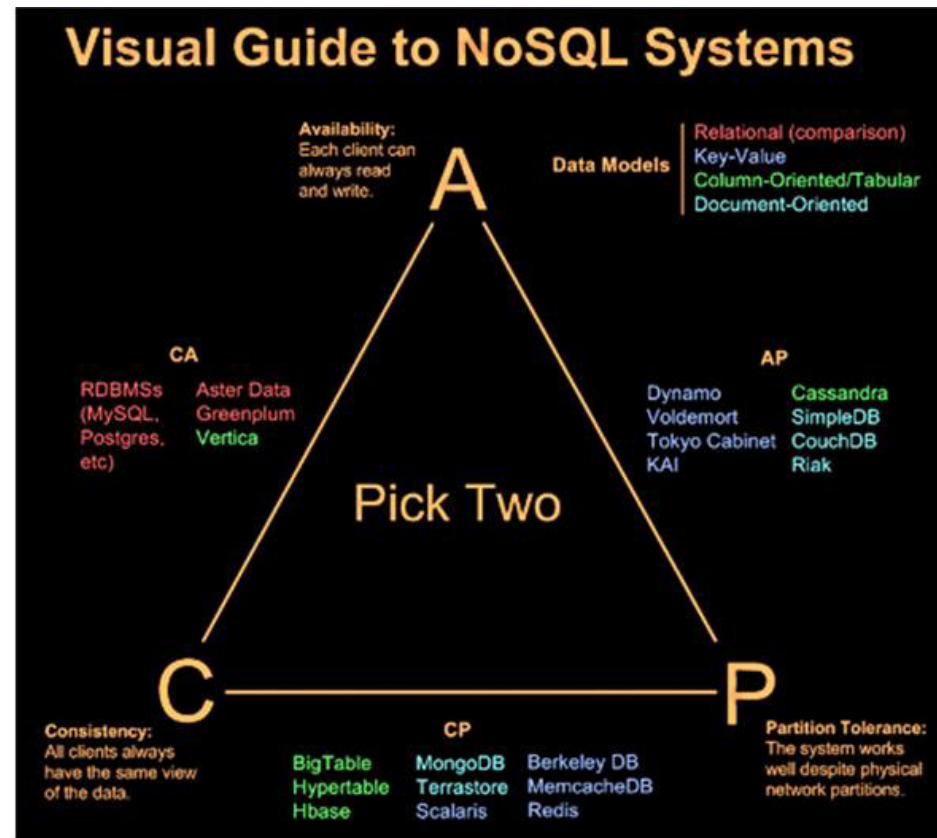
- Acronym contrived to be the opposite of ACID
 - **B**asically **A**vailable,
 - **S**oft state,
 - **E**ventually **C**onsistent
- Characteristics
 - Weak consistency – stale data OK
 - Availability first
 - Best effort
 - Approximate answers OK
 - Aggressive (optimistic)
 - Simpler and faster

CAP Theorem

A congruent and logical way for assessing the problems involved in assuring ACID-like guarantees in distributed systems is provided by the CAP theorem

At most two of the following three can be maximized at one time

- Consistency
 - Each client has the same view of the data
- Availability
 - Each client can always read and write
- Partition tolerance
 - System works well across distributed physical networks



Định lý CAP: Chỉ Có Thể Chọn Hai trong Ba

- **Định lý CAP** (Consistency, Availability, Partition Tolerance) phát biểu rằng trong một hệ thống phân tán, chỉ có thể đảm bảo **hai** trong ba yếu tố sau đây đồng thời:
 1. Tính khả dụng bị hy sinh, nhưng tính nhất quán và khả năng chịu phân hoạch được ưu tiên hơn.
 2. Hệ thống có ít hoặc không có khả năng chịu phân hoạch. Tính nhất quán và tính khả dụng được ưu tiên.
 3. Tính nhất quán bị hy sinh, nhưng hệ thống luôn khả dụng và có thể hoạt động khi một phần của nó bị phân hoạch.

Consistency hay Availability?

- **Consistency (Tính nhất quán):**

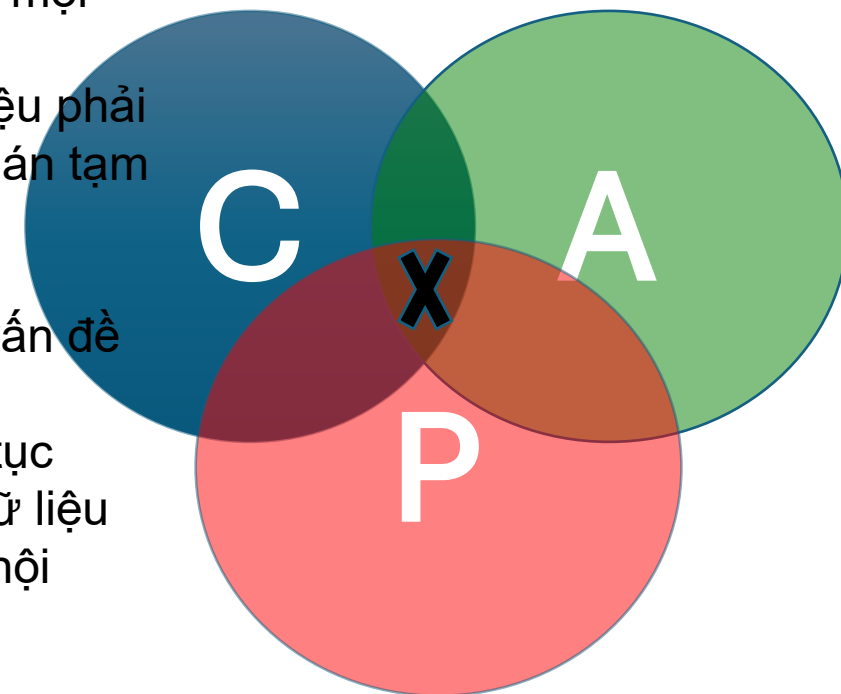
- Mọi thao tác đọc sẽ trả về dữ liệu chính xác và cập nhật mới nhất từ mọi nút trong hệ thống.

- **Ưu tiên tính nhất quán** thường cần thiết cho các hệ thống mà dữ liệu phải hoàn toàn chính xác và không thể chấp nhận được sự không nhất quán tạm thời. Ví dụ: các hệ thống ngân hàng hoặc giao dịch tài chính.

- **Availability (Tính khả dụng):**

- Hệ thống luôn có thể phản hồi yêu cầu, bất kể có lỗi mạng hay các vấn đề phân hoạch.

- **Ưu tiên tính khả dụng** phù hợp cho các hệ thống yêu cầu tính liên tục trong dịch vụ và khả năng phục vụ không bị gián đoạn, ngay cả khi dữ liệu có thể không hoàn toàn nhất quán ngay lập tức. Ví dụ: các mạng xã hội hoặc dịch vụ web trực tuyến.



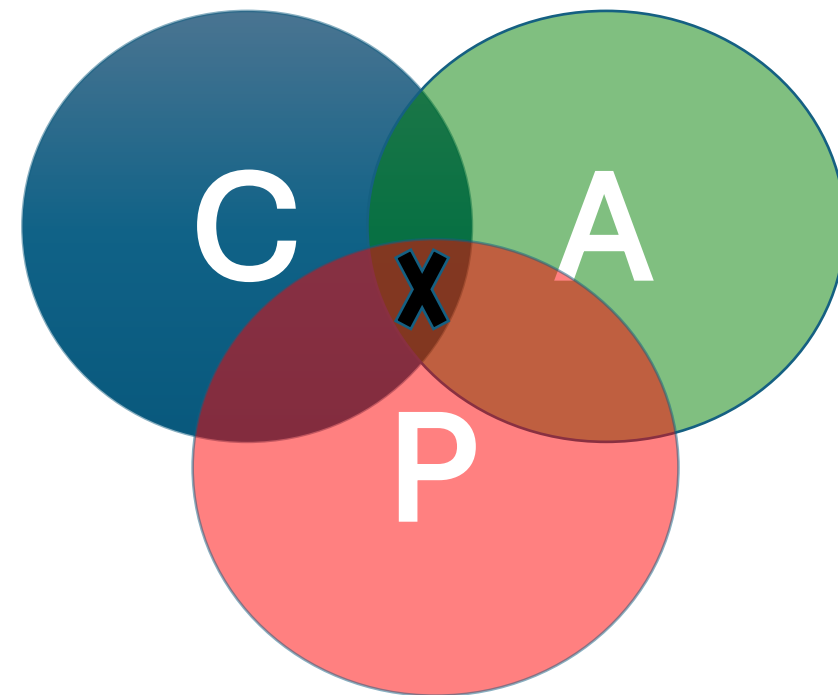
Consistency hay Availability?

Chọn tính nhất quán (Consistency):

- Khi yêu cầu dữ liệu phải chính xác tại mọi thời điểm.
- Thích hợp với các hệ thống mà sai lệch dữ liệu có thể gây ra hậu quả nghiêm trọng.

Chọn tính khả dụng (Availability):

- Khi yêu cầu hệ thống luôn sẵn sàng và không bị ngắt quãng, ngay cả khi dữ liệu không ngay lập tức đồng bộ.
- Phù hợp với các ứng dụng yêu cầu tính liên tục và có thể chấp nhận sự không nhất quán tạm thời.



Hiệu năng

Không có cơ sở dữ liệu NoSQL nào hoàn hảo.

Mỗi cơ sở dữ liệu đều có ưu và nhược điểm riêng.

Tùy thuộc vào loại nhiệm vụ (và sở thích) cần thực hiện.

NoSQL là một tập hợp các khái niệm, ý tưởng, công nghệ, và phần mềm giải quyết:

- Dữ liệu lớn (Big Data)
- Dữ liệu thưa thớt, không có cấu trúc hoặc bán cấu trúc
- Khả năng mở rộng ngang cao
- Xử lý song song khổng lồ

Các ứng dụng, mục tiêu, đối tượng, và phương pháp khác nhau đòi hỏi các giải pháp NoSQL khác nhau.

Khi nào sử dụng cơ sở dữ liệu NoSQL?

- Bạn có một tập hợp lớn dữ liệu không được kiểm soát, không có cấu trúc mà bạn đang cố gắng đưa vào RDBMS không?
- Phân tích nhật ký
- Nguồn cấp dữ liệu mạng xã hội (nhiều công ty kết nối thông qua Facebook hoặc Twitter)
- Nguồn cấp dữ liệu bên ngoài từ các đối tác
- Dữ liệu không dễ phân tích trong RDBMS như dữ liệu theo thời gian
- Nguồn cấp dữ liệu lớn cần được xử lý trước khi đưa vào RDBMS

Đừng quên vai trò của DBA (Quản trị viên cơ sở dữ liệu)

- Không quan trọng liệu dữ liệu được triển khai trên nền tảng NoSQL hay RDBMS.

Vẫn cần phải giải quyết các vấn đề sau:

- Sao lưu và phục hồi
- Lập kế hoạch dung lượng
- Giám sát hiệu suất
- Tích hợp dữ liệu
- Tinh chỉnh và tối ưu hóa