# NoSQL là gì ?

* Cơ sở dữ liệu NoSQL (còn gọi là "không chỉ SQL") là cơ sở dữ liệu không dạng bảng và lưu trữ dữ liệu khác với các bảng quan hệ. Cơ sở dữ liệu NoSQL có nhiều loại khác nhau dựa trên mô hình dữ liệu của chúng. Các loại chính là tài liệu, khóa-giá trị, cột rộng và biểu đồ. Chúng cung cấp các lược đồ linh hoạt và mở rộng quy mô một cách dễ dàng với lượng dữ liệu lớn và lượng người dùng cao.
* Cơ sở dữ liệu NoSQL cho phép các nhà phát triển lưu trữ lượng lớn dữ liệu phi cấu trúc, mang lại cho họ rất nhiều tính linh hoạt.

### **Tóm tắt lịch sử cơ sở dữ liệu NoSQL**

Cơ sở dữ liệu NoSQL xuất hiện vào cuối những năm 2000 khi chi phí lưu trữ giảm đáng kể. Đã qua rồi cái thời phải tạo một mô hình dữ liệu phức tạp, khó quản lý để tránh trùng lặp dữ liệu.

## Các tính năng :

Mỗi cơ sở dữ liệu NoSQL đều có những tính năng độc đáo riêng. Ở cấp độ cao, nhiều cơ sở dữ liệu NoSQL có các tính năng sau:

* [Lược đồ linh hoạt](https://docs.mongodb.com/manual/core/data-modeling-introduction/#flexible-schema)
* [Chia tỷ lệ theo chiều ngang](https://www.mongodb.com/basics/scaling)
* [Truy vấn nhanh nhờ mô hình dữ liệu](https://docs.mongodb.com/manual/core/data-modeling-introduction/#document-structure)
* [Dễ sử dụng cho các nhà phát triển](https://www.mongodb.com/why-use-mongodb)

## Các khuyết điểm:

* Không tham gia,group by, sắp xếp theo thao tác( ngoại trừ trong các phân vùng)
* Không có ràng buộc toàn vẹn tham chiếu đến các phân vùng
* Không có nhiều ngôn ngữ truy vấn khai báo (ví dụ: SQL) lập trình
* **Thiếu tính nhất quán:**NoSQL đánh đổi sự nhất quán để ưu tiên tốc độ, hiệu suất hiệu quả hơn.

### **Khi nào nên sử dụng NoSQL?**

Khi quyết định sử dụng cơ sở dữ liệu nào, những người ra quyết định thường tìm thấy một hoặc nhiều yếu tố sau đây khiến họ chọn cơ sở dữ liệu NoSQL:

* Phát triển Agile nhịp độ nhanh
* Lưu trữ dữ liệu có cấu trúc và bán cấu trúc
* Khối lượng dữ liệu khổng lồ
* Yêu cầu đối với kiến ​​trúc mở rộng quy mô
* Các mô hình ứng dụng hiện đại như microservice và phát trực tuyến theo thời gian thực

### **Những quan niệm sai lầm về cơ sở dữ liệu NoSQL**

Trong những năm qua, nhiều quan niệm sai lầm về cơ sở dữ liệu NoSQL đã lan rộng khắp cộng đồng nhà phát triển. Trong phần này, chúng ta sẽ thảo luận về hai quan niệm sai lầm phổ biến nhất:

* Dữ liệu mối quan hệ phù hợp nhất cho cơ sở dữ liệu quan hệ.
* Cơ sở dữ liệu NoSQL không hỗ trợ giao dịch ACID.

### **Các loại cơ sở dữ liệu NoSQL**

Theo thời gian, bốn [loại cơ sở dữ liệu NoSQL](https://www.mongodb.com/scale/types-of-nosql-databases) chính đã xuất hiện: document databases (CSDL tài liệu),  [key-value databases](https://www.mongodb.com/databases/key-value-database) ([cơ sở dữ liệu khóa-giá trị](https://www.mongodb.com/databases/key-value-database)), wide-column stores (kho lưu trữ cột rộng) và graph databases (cơ sở dữ liệu đồ thị).

* Cơ sở dữ liệu tài liệu lưu trữ dữ liệu trong các tài liệu tương tự như các đối tượng JSON (Ký hiệu đối tượng JavaScript). Mỗi tài liệu chứa các cặp trường và giá trị. Các giá trị thường có thể có nhiều loại khác nhau, bao gồm những thứ như chuỗi, số, boolean, mảng hoặc đối tượng.
* Cơ sở dữ liệu khóa-giá trị là loại cơ sở dữ liệu đơn giản hơn trong đó mỗi mục chứa khóa và giá trị.
* Cửa hàng cột rộng lưu trữ dữ liệu trong bảng, hàng và cột động.
* Cơ sở dữ liệu đồ thị lưu trữ dữ liệu trong các nút và cạnh. Các nút thường lưu trữ thông tin về con người, địa điểm và đồ vật, trong khi các cạnh lưu trữ thông tin về mối quan hệ giữa các nút.

### Key-value

* Tập trung vào việc mở rộng quy mô tới lượng dữ liệu khổng lồ
* Được thiết kế để xử lý tải trọng lớn
* Dựa trên bài báo về máy phát điện của Amazon
* Mô hình dữ liệu: tập hợp (toàn cầu) các cặp Khóa-giá trị
* Phân vùng và sao chép vòng Dynamo
* Truy cập API cơ bản:
* là cấu trúc dữ liệu đặc biệt (ví dụ: Danh sách, Tập hợp, Bản đồ .... v.v.)

### Column-based

* Dựa trên bài viết BigTable của Google
* Giống như cơ sở dữ liệu quan hệ định hướng theo cột (lưu trữ dữ liệu theo thứ tự cột)
* Mô hình dữ liệu:

○ Bộ sưu tập các họ cột

○ Họ cột = (khóa, giá trị) trong đó giá trị = tập hợp các cột liên quan (tiêu chuẩn, siêu)

A diagram of a diagram of a product

Description automatically generated with medium confidence

Chúng mang lại hiệu suất cao cho các truy vấn tổng hợp như SUM, COUNT, AVG, MIN, v.v. vì dữ liệu có sẵn trong một cột.

Giới hạn của Column based NoSQL database: Cần nhiều thời gian để đọc dữ liệu

CSDL Column Based NoSQL tiêu biểu: Hbase, Cassandra

### **Document-based**

NoSQL Document Database lưu trữ và truy xuất dữ liệu dưới dạng một cặp giá trị khóa (key value pair )nhưng phần giá trị được lưu trữ dưới dạng tài liệu. Tài liệu được lưu trữ ở định dạng JSON hoặc XML (file Excel)

A screenshot of a computer

Description automatically generated

Loại document này chủ yếu được sử dụng cho các hệ thống CMS, nền tảng blog, phân tích thời gian thực và các ứng dụng thương mại điện tử. Document database không nên sử dụng cho các giao dịch phức tạp yêu cầu nhiều hoạt động hoặc truy vấn dựa trên các cấu trúc tổng hợp khác nhau.

Giới hạn của NoSQL document database:

* Thông tin cơ sở trùng lặp trên nhiều tài liệu
* Thiết kế phức tạp dẫn đến không nhất quán.

Các hệ thống DBMS Document database NoSQL tiêu biểu: Amazon SimpleDB, CouchDB, MongoDB

### **Graph-based**

* Tập trung vào mô hình hóa cấu trúc dữ liệu (kết nối liên thông)
* Đồ thị bao gồm hai phần tử: nút và mối quan hệ.
* Điều chỉnh theo mức độ phức tạp của dữ liệu
* Lấy cảm hứng từ Lý thuyết đồ thị toán học (G=(E,V))
* Mô hình dữ liệu:
* Giao diện và ngôn ngữ truy vấn khác nhau
* Biểu thức một bước, đường dẫn và đệ quy đầy đủ

**Ví dụ**

Giả sử chúng ta muốn tìm hiểu Alice, một nhân viên của một công ty, thuộc bộ phận nào.

A diagram of a group of people

Description automatically generated

Trong cơ sở dữ liệu quan hệ, thông thường cần thiết lập bảng thông tin nhân viên, bảng tương ứng nhân viên và phòng ban (giả sử rằng một nhân viên có thể thuộc nhiều phòng ban) và bảng thông tin phòng ban. Có 3 bước để tìm:

A, trước tiên hãy tìm mã số công việc tương ứng của Alice thông qua mẫu thông tin nhân viên;

B, sau đó sử dụng mã số công việc để tìm ID phòng ban tương ứng trong bảng mối quan hệ;

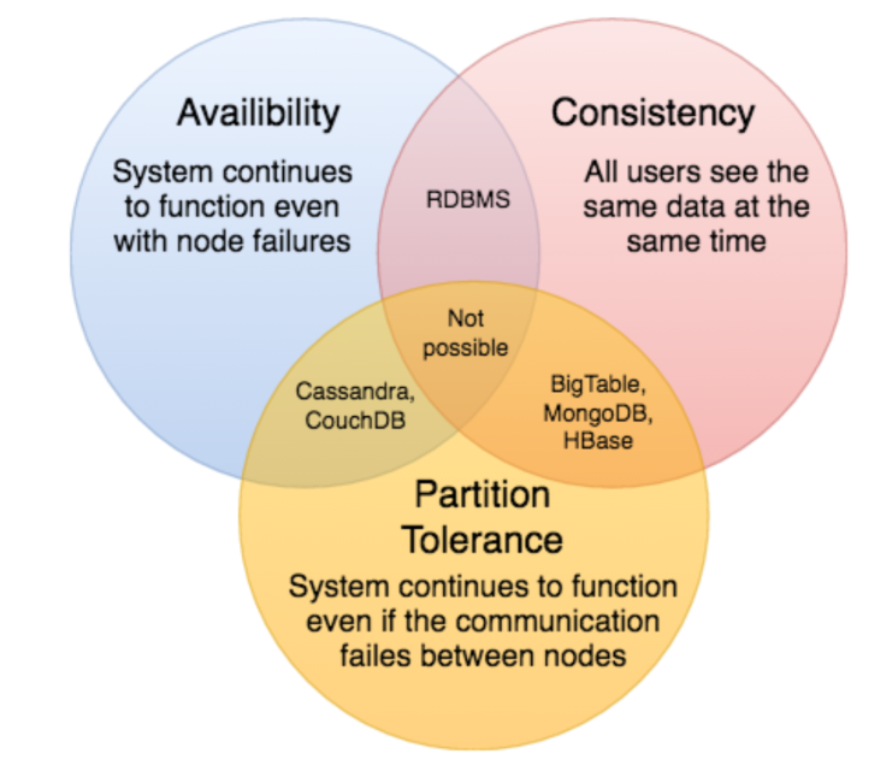
C. Cuối cùng, sử dụng ID phòng ban để tìm tên phòng ban và các thông tin khác trong bảng thông tin phòng ban.

A cần quá trình tra cứu chỉ mục, B cũng cần tra cứu chỉ mục và C có thể cần 3 lần tra cứu chỉ mục. Giả sử đó là một công ty lớn với hàng chục nghìn, thậm chí hàng trăm nghìn nhân viên. Trong trường hợp đó, bản ghi của các bảng nhân viên và phòng ban tương ứng sẽ rất phong phú và hiệu quả tìm kiếm của B sẽ thấp hơn nhiều so với A và C.

# Định lý CAP

Định lý CAP nói rằng một hệ thống phân tán (*distributed system*) không thể thỏa mãn cả ba yếu tố CAP đó là:

* **Consistency**: trong 1 hệ thống phân tán,mọi request tới hệ thống phải có phản hồi kết quả là lần ghi cuối cùng hoặc thông báo lỗi.
* **Availability**: tính sẵn sàng hoạt động của các node. Hệ thống có thể vẫn hoạt động được khi một số node bị chết hoặc không sẵn sàng.
  + **Partition Fault Tolerance** (phân mảnh): hệ thống vẫn hoạt động được(thực hiện chức năng hoặc đáp trả query) ngay cả khi mạng bị lỗi (1 bộ phận bị lỗi hay cô lập) hoặc có độ trễ khi gửi message (hoặc mất mesage) giữa các node.



Chúng ta không thể thiết kế một hệ thống bao gồm cả ba tính CAP, bởi vì đảm bảo tính C (consistency) tất cả các cập nhật dữ liệu phải được thực hiện trên các node cùng một thời điểm. Nhưng nếu đường kết nối (mạng) giữa các node không được đảm bảo dẫn đến việc các node sẽ không được update dữ liệu cùng một thời điểm, điều này dẫn tới việc một vài node dữ liệu sẽ bị out-of-date do chưa được cập nhật dữ liệu từ đó vi phạm tính C (consistency). Và để đảm bảo đối phó với điều này ta sẽ ngừng phục vụ nhữg node bị out-of-date đó cho tới khi nó được update dữ liệu đầy đủ, nhưng việc này lại vi phạm tính A (availability) của hệ thống.

Thông thường, người ta thường đánh đổi yếu tố C để lấy hai yếu tố A và P. Khi đó họ sẽ thay thế Consistency thành eventually consistency (tính nhất quán có độ trễ), làm như thế hệ thống sẽ có hiệu năng tốt hơn

## Tính nhất quán

● Có 2 loại tính nhất quán:

○ Tính nhất quán mạnh mẽ – **ACID** (Tính nguyên tử, Tính nhất quán, Sự cô lập, Độ bền)

○ Tính nhất quán yếu – **BASE** (Tính nhất quán cuối cùng ở trạng thái mềm có sẵn về cơ bản)

### **ACID**

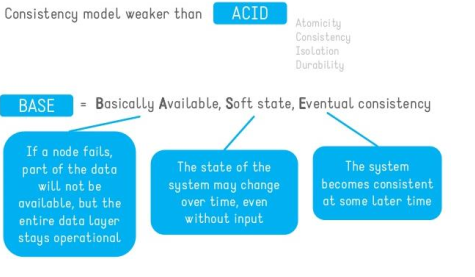
A diagram of properties

Description automatically generated with medium confidence

Khi tính toàn vẹn và độ tin cậy của dữ liệu được cân nhắc hàng đầu trong [hệ thống xử lý giao dịch](https://www.techtarget.com/searchdatacenter/definition/OLTP) , hệ thống thường sẽ áp dụng bốn thuộc tính cho các giao dịch đó để tuân thủ ACID:

* **Tính nguyên tử.**Một giao dịch được coi là một đơn vị nguyên tử duy nhất. Tất cả các bước tạo nên giao dịch phải thành công nếu không toàn bộ giao dịch sẽ bị hủy. Nếu tất cả đều thành công, những thay đổi do giao dịch thực hiện sẽ được cam kết vĩnh viễn với hệ thống quản lý. Hãy xem xét ví dụ về giao dịch chuyển nhượng. Để giao dịch được cam kết với [cơ sở dữ liệu](https://www.techtarget.com/searchdatamanagement/definition/database) , 200 đô la phải được khấu trừ thành công từ tài khoản tiết kiệm và thêm vào tài khoản séc, đồng thời số tiền trong cả hai tài khoản phải được xác minh để đảm bảo tính chính xác của chúng. Nếu bất kỳ tác vụ nào trong số này không thành công, tất cả các thay đổi sẽ được khôi phục và không có thay đổi nào được cam kết.
* **Tính nhất quán.**Một giao dịch phải duy trì tính nhất quán của dữ liệu cơ bản. Giao dịch không được thực hiện các thay đổi vi phạm các quy tắc hoặc ràng buộc được đặt trên dữ liệu. Ví dụ: cơ sở dữ liệu hỗ trợ các giao dịch ngân hàng có thể bao gồm quy tắc cho biết số dư tài khoản của khách hàng không bao giờ có thể là số âm. Nếu một giao dịch cố gắng rút nhiều tiền từ tài khoản hơn số tiền có sẵn thì giao dịch đó sẽ không thành công và mọi thay đổi được thực hiện đối với dữ liệu sẽ bị hủy bỏ.
* **Sự cách ly.**Một giao dịch được tách biệt khỏi tất cả các giao dịch khác. Các giao dịch chỉ có thể chạy đồng thời nếu chúng không ảnh hưởng lẫn nhau. Quay lại ví dụ về giao dịch chuyển khoản, nếu một giao dịch khác cố gắng rút tiền từ cùng một tài khoản tiết kiệm, thì việc cô lập sẽ ngăn giao dịch thứ hai thực hiện. Nếu không bị cô lập, giao dịch thứ hai có thể rút được nhiều tiền hơn số tiền có trong tài khoản sau khi giao dịch đầu tiên hoàn tất.
* **Độ bền.**Một giao dịch được cam kết sẽ được đảm bảo duy trì cam kết -- nghĩa là tất cả các thay đổi được thực hiện vĩnh viễn và sẽ không bị mất nếu xảy ra sự kiện như mất điện. Điều này thường có nghĩa là duy trì các thay đổi đối với [bộ nhớ](https://www.techtarget.com/searchstorage/definition/storage) không biến đổi . Nếu độ bền không được đảm bảo, một số hoặc tất cả các thay đổi có thể bị mất, ảnh hưởng đến độ tin cậy của dữ liệu.

### **BASE**



BASE =Về cơ bản có sẵn, Trạng thái mềm, Tính nhất quán cuối cùng.

BASE là một triết lý thiết kế hệ thống dữ liệu coi trọng tính sẵn sàng hơn tính nhất quán của hoạt động.

* Về cơ bản có sẵn: Nếu một nút bị lỗi, một phần dữ liệu sẽ không có sẵn nhưng toàn bộ Lớp dữ liệu vẫn hoạt động
* Trạng thái mềm : Trạng thái của hệ thống có thể thay đổi theo thời gian, ngay cả khi không có đầu vào
* Tinh nhất quán cuối cùng : Hệ thống trở nên nhất quán vào một thời điểm sau đó

# **SỰ KHÁC NHAU GIỮA SQL VÀ NoSQL**

# Bảng dưới đây tóm tắt sự khác nhau giữa SQL và NoSQL database:

|  |  |  |
| --- | --- | --- |
|  | **SQL Databases** | **NoSQL Databases** |
| Mô hình dữ liệu | Mô hình quan hệ chuẩn hóa dữ liệu vào bảng được hình thành từ hàng và cột. Sơ đồ quy định rõ ràng bảng, hàng, cột, chỉ mục, mối quan hệ giữa các bảng và các thành tố cơ sở dữ liệu khác. Cơ sở dữ liệu sẽ thực thi tính toàn vẹn tham chiếu trong mối quan hệ giữa các bảng. | * Kiểu Document: JSON documents * Kiểu Key-value: key-value pairs * Kiểu column: bảng với hàng và cột thay đổi (dynamic) * Graph: nodes và cạnh |
| Lịch sử phát triển | Phát triển từ những năm 1970, tập trung vào việc giảm thiểu sự trùng lắp dữ liệu. | Phát triển sau năm 2000, mục đích chính phục vụ việc mở rộng nhanh (scale) và phát triển các ứng dụng theo phương pháp Agile và DevOps. |
| Các CSDL tiêu biểu | Oracle, MySQL, Microsoft SQL Server  PostgreSQL | * NoSQL Document: MongoDB, CouchDB * Key-value: Redis, DynamoDB * Column: Cassandra, HBase * Graph: Neo4j, Amazon Neptune |
| Mục đích sử dụng | Nhiều mục đích | * NoSQL Document DB: nhiều mục địch * Key-value: dữ liệu lớn với các truy xuất đơn giản * Column: dữ liệu lớn với các mẫu truy xuất dữ liệu có thể dự đoán được (predictable query patterns) * Graph: phân tích và duyệt các mối quan hệ giữa các dữ liệu được kết nối |
| Schemas | Cố định (Rigid) | Uyển chuyển |
| Mở rộng | Theo chiều dọc. Cơ sở dữ liệu quan hệ thường tăng quy mô bằng cách tăng năng lực điện toán của phần cứng hoặc tăng quy mô bằng cách thêm bản sao của khối lượng công việc chỉ đọc. | Cho phép thay đổi quy mô theo chiều ngang (scale-out bằng cách phân tán trên nhiều server) |
| Thuộc tính ACID | Hỗ trợ. Cơ sở dữ liệu quan hệ có các thuộc tính mang tính nguyên tố, nhất quán, tách biệt và bền vững (ACID) | Cơ sở dữ liệu NoSQL tuân theo định lý Brewers CAP (Consistency, Availability, Partition tolerance). (Xem thêm bên dưới về CAP) |
| Joins | Thường được yêu cầu | Thường không yêu cầu |
| Data to Object Mapping | Yêu cầu ORM (object-relational mapping) | Nhiều NoSQL database không yêu cầu ORMs. |
| Hiệu năng | Hiệu năng thường phụ thuộc vào hệ thống con của ổ đĩa. Thông thường, việc tối ưu hóa các truy vấn, chỉ mục và cấu trúc bảng bắt buộc phải được thực hiện để đạt mức hiệu năng tối đa. | Hiệu năng thường được xem là chức năng của kích cỡ cụm phần cứng ngầm, độ trễ mạng và ứng dụng đưa ra lệnh gọi. |
| API | Yêu cầu lưu trữ và truy xuất dữ liệu được truyền đạt bằng cách sử dụng các truy vấn nhất quán với ngôn ngữ truy vấn có cấu trúc (SQL). Các truy vấn này được phân tích và thực thi bởi cơ sở dữ liệu quan hệ. | API trên cơ sở đối tượng cho phép các nhà phát triển ứng dụng dễ dàng lưu trữ và truy xuất cấu trúc dữ liệu trong bộ nhớ. Khóa phân mảnh tìm kiếm các cặp khóa-giá trị, tập hợp cột hoặc văn bản có cấu trúc chưa hoàn chỉnh có chứa đối tượng và thuộc tính của ứng dụng được xếp theo chuỗi. |

# **Phần kết luận**

Cơ sở dữ liệu noSQL được tạo ra để giải quyết các hạn chế của CSDL quan hệ SQL. Tuy nhiên đây không phải là sự thay thế mà tùy vào mục đích mà các nhà phát triển quyết định lựa chọn CSDL SQL hay NoSQL cho phù hợp.