**NoSQL**

**Giới thiệu về NoSQL**

Cơ sở dữ liệu NoSQL (phi quan hệ) cung cấp một cơ chế để lưu trữ và truy xuất dữ liệu được mô hình hóa khác với các quan hệ bảng được sử dụng trong các cơ sở dữ liệu kiểu quan hệ. Các cơ sở dữ liệu NoSQL đang ngày càng được sử dụng nhiều trong các ứng dụng dữ liệu lớn với khả năng mở rộng linh hoạt.

Cơ sở dữ liệu SQL thông thường (ví dụ cơ sở dữ liệu quan hệ) là một sản phẩm đã được áp dụng từ nhiều thập kỷ phát triển công nghệ, cho thấy khả năng ứng dụng và đáp ứng tốt trong nhu cầu thực tế. Chúng được thiết kế cho các giao dịch tin cậy, truy vấn ad hoc (truy vấn dành cho mục đích cụ thể) và các ứng dụng kinh doanh chính thống. Hệ thống các bảng, các mối quan hệ, cấu trúc dữ liệu thường được thiết kế cẩn thận trước khi triển khai. Tuy nhiên,điều này cũng có nghĩa rằng việc thay đổi cấu trúc dữ liệu sẽ trở nên khó khăn và hiếm khi xảy ra trong các phiên bản phần mềm. Ngày nay, với nhu cầu sử dụng dữ liệu lớn, các hệ thống cơ sở dữ liệu truyền thống được thiết kế để chạy trên một server mạnh thường gặp khó khăn trong viêc nâng cấp mở rộng với chi phí rất đắt đỏ.

NoSQL đã ra đời để khắc phục những hạn chế đó. Hệ thống NoSQL lưu trữ và quản trị dữ liệu sao cho có thể hỗ trợ được tốc độ vận hành ở công suất cao và cung cấp tính linh hoạt cho các nhà phát triển sử dụng. Có rất nhiều hệ thống đã được các công ty lớn như Google, Amazon, Yahoo, and Facebook phát triển, cung cấp các phương thức lưu trữ nội dung và xử lý dữ liệu cho các website lớn. Không giống với cơ sở dữ liệu SQL, rất nhiều cơ sở dữ liệu NoSQL có thể mở rộng theo chiều ngang trên hàng trăm hoặc hàng ngàn máy chủ.

Ví dụ về mô hình sơ đồ của cơ sở dữ liệu sách đơn giản:

* Trong cơ sở dữ liệu SQL, hồ sơ về một cuốn sách được phân tách và lưu trữ trong các bảng tách biệt nhau, còn mối quan hệ được quy định thành các ràng buộc khóa ngoại và khóa chính. Mô hình quan hệ được thiết kế để cho phép các cơ sở dữ liệu này thực thi tính toàn vẹn tham chiếu giữa nhiều bảng. Ví dụ,
  + Bảng **Sách** có các cột cho **ISBN** (International Standard Book Number – mã số tiêu chuẩn quốc tế cho sách), **Tên sách,Ngày xuất bản…**
  + Bảng **Tác giả** có các cột cho **ID tác giả** và **Tên tác giả**;
  + Và cuối cùng, bảng **Tác giả-ISBN** có các cột cho **ID tác giả** và **ISBN**.
* Trong cơ sở dữ liệu NoSQL, hồ sơ về một cuốn sách thường được lưu trữ dưới dạng văn bản JSON (JavaScript Object Notation). Với từng quyển sách, mục ISBN, Tên sách, Ngày xuất bản, Tên tác giả và ID tác giả được lưu trữ dưới dạng thuộc tính trong một văn bản duy nhất. Trong mô hình này, dữ liệu được tối ưu hóa cho việc phát triển trực quan và khả năng thay đổi quy mô theo chiều ngang.

Minh họa, hồ sơ cuốn sách lưu trữ dưới dạng JSON.

|  |
| --- |
| {  "books": [  {  "isbn": "9781593275846",  "title": "Eloquent JavaScript, Second Edition",  "author": "Marijn Haverbeke",  "authorid": 2375753,  "published": "2014-12-14T00:00:00.000Z",  "publisher": "No Starch Press",  "website": "http://eloquentjavascript.net/"  },  {  "isbn": "9781449331818",  "title": "Learning JavaScript Design Patterns",  "author": "Addy Osmani",  "authorid": 1159142,  "published": "2012-07-01T00:00:00.000Z",  "publisher": "O'Reilly Media",  "website": "http://www.addyosmani.com/resources/essentialjsdesignpatterns/book/"  }  ]  } |

**Ưu điểm của cơ sở dữ liệu NoSQL**

Cơ sở dữ liệu NoSQL là lựa chọn phù hợp cho nhiều ứng dụng hiện đại, ví dụ như di động, Web và trò chơi đòi hỏi phải sử dụng cơ sở dữ liệu linh hoạt, có khả năng thay đổi quy mô và hiệu năng cao để đem đến cho người dùng trải nghiệm tốt nhất.

* Linh hoạt: Cơ sở dữ liệu NoSQL thường cung cấp các sơ đồ linh hoạt giúp công đoạn phát triển nhanh hơn và có khả năng lặp lại cao hơn. Mô hình dữ liệu linh hoạt biến cơ sở dữ liệu NoSQL thành lựa chọn lý tưởng cho dữ liệu không được tổ chức thành cấu trúc hoặc có cấu trúc chưa hoàn chỉnh.
* Khả năng thay đổi quy mô: Cơ sở dữ liệu NoSQL thường được thiết kế để tăng quy mô bằng cách sử dụng sử dụng các cụm phần cứng được phân phối thay vì tăng quy mô bằng cách bổ sung máy chủ mạnh và tốn kém. Một số nhà cung cấp dịch vụ đám mây xử lý các hoạt động này một cách không công khai dưới dạng dịch vụ được quản lý đầy đủ như Google, Amazon.
* Hiệu năng cao: Cơ sở dữ liệu NoSQL được tối ưu hóa theo mô hình dữ liệu (ví dụ như key-value, document và graph) và các mẫu truy cập giúp tăng hiệu năng cao hơn so với việc cố gắng đạt được mức độ chức năng tương tự bằng cơ sở dữ liệu quan hệ.
* Tính thiết thực: Cơ sở dữ liệu NoSQL cung cấp các API và kiểu dữ liệu cực kỳ thiết thực được xây dựng riêng cho từng mô hình dữ liệu tương ứng.

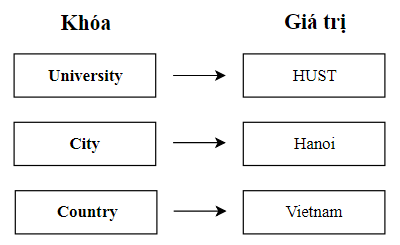
**Các loại cơ sở dữ liệu NoSQL phổ biến**

Các hệ thống cơ sở dữ liệu NoSQL sử dụng nhiều loại mô hình dữ liệu đa dạng để truy cập và quản lý dữ liệu. Người ta thường chia CSDL NoSQL thành nhiều loại căn cứ vào mô hình thiết kế của chúng, ví dụ như key-value (khóa-giá trị), document (văn bản), graph (đồ thị)…

**a/ Cơ sở dữ liệu dạng Key-value**

Lưu trữ kiểu key-value (khóa-giá trị) là kiểu lưu trữ dữ liệu NoSQL đơn giản nhất. Kiểu NoSQL này sử dụng bảng băm (hash table) để lưu các khóa-duy-nhất (unique-key) trỏ đến các giá trị tương ứng. Giá trị có thể là bất cứ dạng dữ liệu nào: giá trị dạng số, text, JSON... Client có thể sử dụng 3 phương thức cơ bản: GET, PUT, DELETE để ghi và đọc dữ liệu. Và vì NoSQL khóa-giá trị luôn truy cập thông qua khóa chính (Primary Key), nên rất dễ mở rộng vào có hiệu năng truy cập tốt.

Cơ sở dữ liệu dạng key-value là giải pháp cho những ứng dụng yêu cầu tốc độ đọc ghi nhanh: Quản lý session, data caching. Mô hình dữ liệu khóa-giá trị được dùng cho trò chơi, công nghệ quảng cáo và đặc biệt là thích hợp cho IoT. Tuy nhiên, nó không phù hợp với các hệ thống đòi hỏi truy vấn phức tạp. Một vài cơ sở dữ liệu khóa-giá trị phổ biến là Redis, Riak (thường dùng phía Server); Amazon DynamoDB, HamsterDB (mã nguồn đóng).



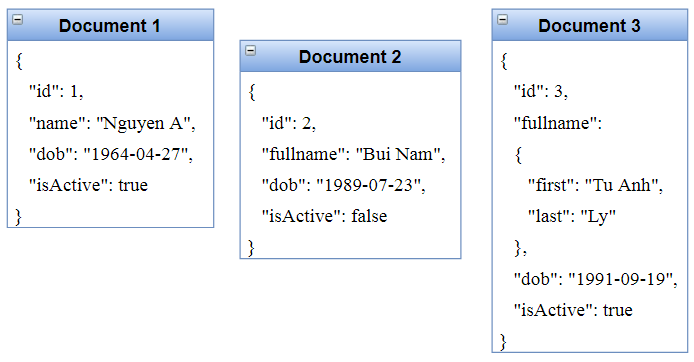
**b/ Cơ sở dữ liệu dạng Document**

Một số nhà phát triển không thích hình dung mô hình dữ liệu dưới dạng các hàng và cột phi chuẩn hóa. Thông thường, ở bậc ứng dụng, dữ liệu được thể hiện dưới dạng văn bản JSON bởi vì định dạng này trực quan hơn đối với các nhà phát triển để giúp họ xem mô hình dữ liệu của mình dưới dạng văn bản. Sự phổ biến của cơ sở dữ liệu document (văn bản) đã tăng lên do các nhà phát triển có thể tiếp tục sử dụng dữ liệu trong cơ sở dữ liệu bằng cách sử dụng cùng định dạng văn bản mà họ đã sử dụng trong mã ứng dụng của mình.

Lưu trữ document cũng gần giống lưu trữ key-value, với mô hình quản lý theo khóa trỏ đến giá trị tương ứng. Điểm khác biệt là document lưu trữ dữ liệu có cấu trúc dạng JSON, BSON hoặc XML. Mỗi document là một đối tượng có cấu trúc mà thành phần có thể là dạng string, date, binary, array... Điều này giúp document dễ dàng được đánh số và truy vấn thông qua các thuộc tính.

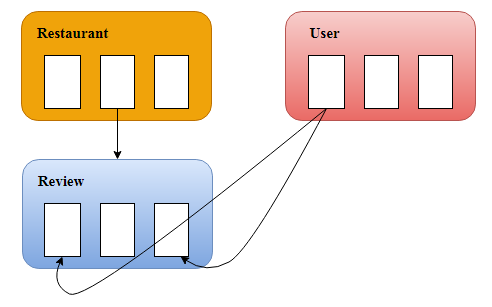
Document có cấu trúc mềm dẻo, dễ thay đổi theo thời gian, mỗi document có thể có cấu trúc khác nhau. Các documents có thể được nhóm lại trong các containers tùy thuộc vào yêu cầu bài toán. Mặt khác, các documents còn có thể được chứa ngay trong một document với cấu trúc không cần phải được xác định.

Với sự mềm dẻo trong cấu trúc dữ liệu, và khả năng làm việc trên Cluster với lượng dữ liệu lớn, dễ scale-out, Document Stores thường được sử dụng trong các bài toán: hệ thống Content Management, hệ thống E-commerce. DynamoDB và MongoDB là các cơ sở dữ liệu văn bản phổ biến cung cấp các API trực quan và mạnh mẽ để phục vụ công việc phát triển linh hoạt và nhanh chóng.



**Những hạn chế của NoSQL**

Điểm yếu lớn nhất của NoSQL là lưu trữ dữ liệu trùng lặp, dẫn đến việc thay đổi dữ liệu phải cập nhật ở nhiều nơi, nếu có thiếu sót sẽ dẫn đến việc dữ liệu không nhất quán. Ví dụ một cơ sở dữ liệu NoSQL quản lý hệ thống nhà hàng trong đó mỗi nhà hàng được người dùng đánh giá chất lượng sau khi sử dụng dịch vụ. Như vậy mỗi nhà hàng sẽ có một tập các review mà mỗi review sẽ liên kết với thông tin của một người dùng. Rõ ràng nếu muốn thay đổi thông tin của đối tượng người dùng, thì chúng ta sẽ phải tìm và sửa đổi thông tin liên quan đến đối tượng đó được lưu trữ trong tất cả các review của họ.



Tuy nhiên, xét ở một góc nhìn khác thì đây không hẳn là một hạn chế lớn nếu so sánh số lần thay đổi thông tin cá nhân của người dùng (ví dụ cập nhật profile tài khoản foody) là rất ít so với số lần mà hàng ngàn người dùng khác muốn xem các đánh giá về nhà hàng đó – rất dễ dàng đối với NoSQL khi dữ liệu đã tập trung một nơi, không phải mất công tìm kiếm ở nhiều bảng khác nhau.

NoSQL có một số trở ngại phải vượt qua để có thể trở thành một cơ sở dữ liệu có sức hút lớn hơn với các nhà phát triển. Sau đây là một số điểm hạn chế của NoSQL:

* Khả năng tin cậy: Với đa số doanh nghiệp và người sử dụng đã quen thuộc với khái niệm SQL – hệ cơ sở dữ liệu truyền thống đã có thời gian phát triển lâu, nhận được sự hỗ trợ từ nhà cung cấp khi có sự cố. Họ yên tâm về sự ổn định cũng như sự đa dạng về chức năng của nó đủ đáp ứng các yêu cầu. Trong khi các lựa chọn NoSQL vẫn đang được các nhà phát triển nghiên cứu, mới bước đầu thực hiện một số chức năng quan trọng, việc tìm được các chuyên gia về NoSQL là khó khăn hơn rất nhiều.
* So với hệ cơ sở dữ liệu SQL, NoSQL không có tính nhất quán bằng. Trên thực tế, cơ sở dữ liệu SQL thường sẽ ưu tiên tuân thủ các thuộc tính ACID (nguyên tố, nhất quán, tách biệt, và bền vững) đảm bảo độ tin cậy cho các giao dịch, sau đó mới xét đến hiệu suất và khả năng mở rộng, trong khi cơ sở dữ liệu NoSQL gần như bỏ qua các đảm bảo ACID để ưu tiên tốc độ và khả năng mở rộng.

Tóm lại, NoSQL và SQL có các ưu nhược điểm khác nhau và chúng có vai trò tương hỗ, bổ sung cho nhau khi đặt trong một bức tranh tổng thể. Mỗi hệ thống sẽ phù hợp với các trường hợp sử dụng khác nhau. Việc quyết định lựa chọn công cụ nào cần phụ thuộc vào yêu cầu của hệ thống và tính chất công việc thực tế.

Tham khảo:

<https://tech.vccloud.vn/nosql-la-gi-20181013113252686.htm>

<https://aws.amazon.com/vi/nosql/>

<https://www.youtube.com/watch?v=v_hR4K4auoQ>

Thống kê tỉ lệ sử dụng NoSQL

<https://jaxenter.com/top-databases-2017-132912.html>

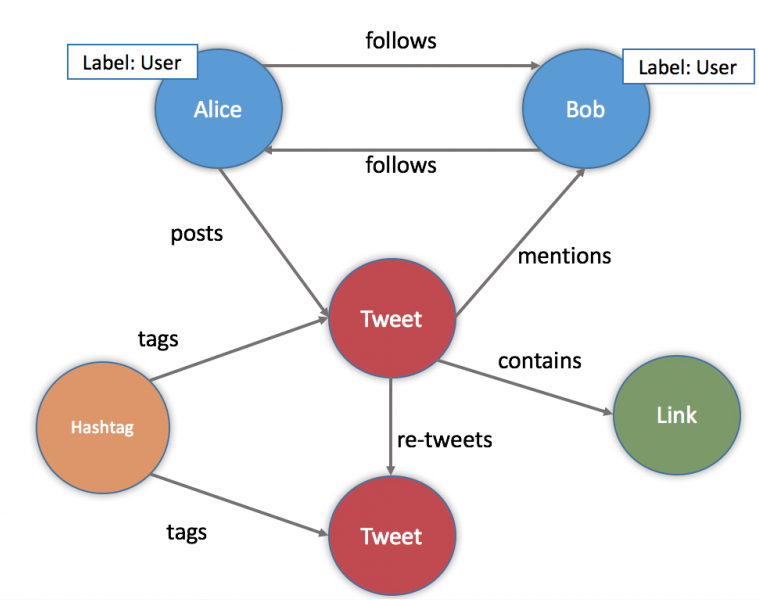
c/ Đồ thị:

Mục đích của cơ sở dữ liệu đồ thị là giúp việc dựng và chạy ứng dụng hoạt động với các bộ dữ liệu có khả năng kết nối cao trở nên dễ dàng. Cơ sở dữ liệu đồ thị thường được sử dụng cho các đồ thị tri thức, mạng xã hội, công cụ đề xuất và phát hiện lừa đảo. Amazon Neptune là dịch vụ cơ sở dữ liệu đồ thị được quản lý đầy đủ. Neptune hỗ trợ cả mô hình Đồ thị thuộc tính lẫn Framework mô tả tài nguyên (RDF), cung cấp cho người dùng lựa chọn hai API đồ thị: TinkerPop và RDF/SPARQL. Các cơ sở dữ liệu đồ thị phổ biến gồm có Neo4j và Giraph.

Graph Database chú trọng vào mối quan hệ giữa các đối tượng trong hệ thống. Ví dụ điển hình là trong mạng xã hội Facebook, bạn có bài viết nào, được tag vào ở bài viết nào, có mối quan hệ với ai... điều đó được thể hiện trong các Nodes và Relationships của Graph Database.

Graph Database bao gồm các Nodes, và Relationships giữa các Nodes. cả Nodes và Relationships đều có các thuộc tính gọi là properties. Nodes là các thực thể trong graph và có thể được gắn các Lable cung cấp context và metadata. Trong khi đó Relationships cung cấp sự kết nối trực tiếp, đơn phương hay song phương giữa hai Notes. Có thể có nhiều hơn 1 mối quan hệ (relationships) giữa hai Nodes và mỗi mối quan hệ có hướng, kiểu, điểm đầu (Start Node) và điểm cuối (End Node).

Ví dụ cho loại cơ sở dữ liệu này là Neo4j.

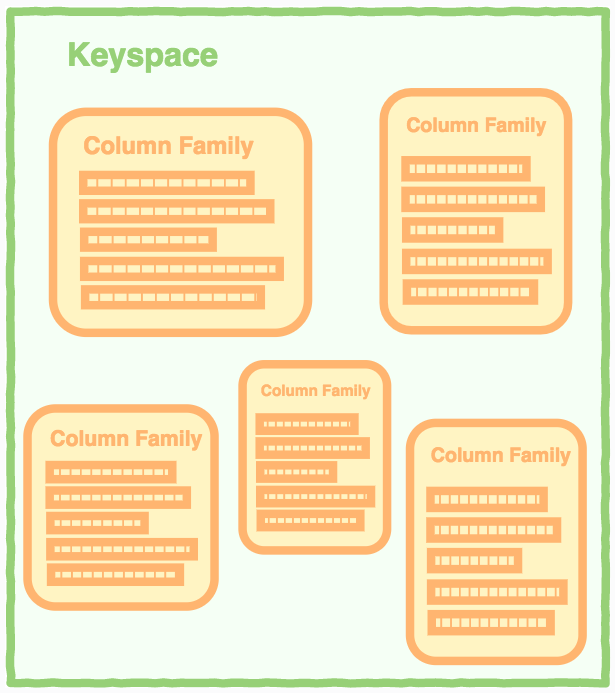


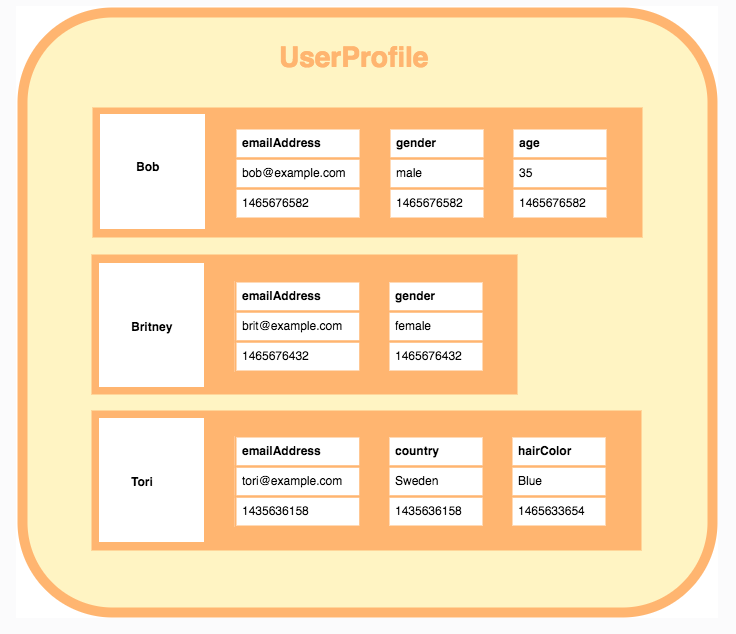
d/

Column Family database

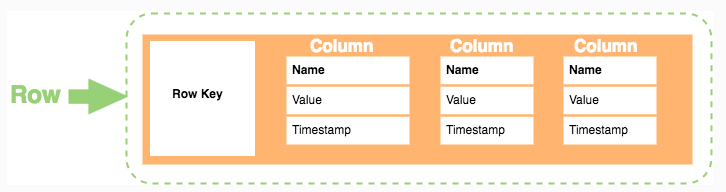
Column Family database sử dụng concept gọi là keyspace. Keyspace tương đương với Schema trong Rational Database. Keyspace chứa tất cả các Column Family.

Vậy Column Family là gì? Hãy xem một ví dụ về Column Family lưu thông tin User Profile như sau:





Như vậy, mỗi Column Family chứa các Rows, mỗi Row chứa các Columns, Mỗi Column có tên chính là từng attribute của Object và Value, cùng timestamp.



Như vậy nếu so sánh với Rational Database ta thấy rõ sự khác biệt: Mỗi Table có nhiều rows, mỗi rows là một object với các column là các attributes, điều này cũng có nghĩa tất cả các object thuộc một table sẽ được lưu vào cùng một nơi.

Với cách tiếp cận của Column Family, mỗi object được lưu vào 1 row trong Column Family, và dễ dàng phân tán ở nhiều nơi.

Column Family mang lại các lợi ích như sau:

Hiệu quả khi nén (Compression) hoặc phân vùng dữ liệu

Dễ mở rộng

Truy vấn và load dữ liệu nhanh

Các hệ cơ sở dữ liệu loại Column Family: BigTable, Cassandra, HBase...

* Hỗ trợ chuyên môn: Đa số các hệ cơ sở dữ liệu truyền thống đã có thời gian phát triển lâu, nhận được sự hỗ trợ từ nhà sản xuất khi có sự cố. Trong khi đó, NoSQL vẫn đang được các nhà phát triển nghiên cứu, việc tìm được các chuyên gia về NoSQL là khó khăn hơn rất nhiều.