Đề tài: A study of NOSQL
Nhóm 6:

Tóm tắt

Trong thời đại công nghệ số, ngày càng có nhiều dịch vụ web 2.0 cần xử lí dữ liệu lớn (big data). Điều này yêu cầu cơ sở dữ liệu quan hệ hiện có mở rộng quy mô theo chiều ngang(tăng khả năng lưu trữ, xử lý bằng cách thêm nhiều máy chủ hơn) để đạt được các yêu cầu cao về hiệu suất, đặc biệt là đối với các ứng dụng có yêu cầu lớn về quy mô dữ liệu người dùng và khả năng xử lý đồng thời cao. Tuy nhiên việc mở rộng quy mô của cơ sở dữ liệu quan hệ là khá khó khăn. Các vấn đề này là những cân nhắc quan trọng để các kĩ sư thiết kế cơ sở dữ liệu đưa ra một nhóm cơ sở dữ liệu mới, thường được gọi là NoSQL. Đồng thời, nhu cầu càng tăng đối với điện toán đám mây và sự phát triển của Internet đẩy mạnh sự phổ biến của NoSQL. Bài viết đề cập đến các tính năng và các mô hình dữ liệu của cơ sở dữ liệu NoSQL được sử dụng trong môi trường điện toán đám mây cùng với thế mạnh và hạn chế của từng mô hình. Ngoài ra, bài viết còn nói về việc phân loại Cơ sở dữ liệu NoSQL dựa trên định lý CAP.

Từ khóa: NoSQL, CAP, Document Oriente,Column Family, Big Data, Transaction

I. GIỚI THIỆU

Lưu trữ và truy xuất hiệu quả, nhanh chóng với tính khả dụng và khả năng mở rộng là các tính năng chính của cơ sở dữ liệu NoSQL. NoSQL không có nghĩa là "Not SQL (không có SQL)" mà nó có nghĩa là "Not Only SQL (không chỉ là SQL)" [10],. Cơ sở dữ liệu NoSQL là một giải pháp thay thế cho cơ sở dữ liệu quan hệ truyền thống. Ngành công nghiệp cơ sở dữ liệu đã chứng kiến sự xuất hiện của hàng loạt cơ sở dữ liệu NOSQL như: MongoDB [11], Hbase [9], Neo4j [8] trong vài năm gần đây. Tùy thuộc vào yêu cầu và chiến lược kinh doanh, nhà cung cấp có thể cung cấp dịch vụ đám mây kèm với bất kì loại cơ sở dữ liệu NOSQL nào. Tuy nhiên vẫn có một số nhà thiết kế cơ sở dữ liệu quan hệ cho rằng NOSQL không đủ hiệu quả trong việc đảm bảo tính toàn vẹn của dữ liệu.

Bài viết này bao gồm các phần như sau: Phần 2 mô tả các tầm quan trọng của cơ sở dữ liệu NoSQL. Phần 3 Nhấn mạnh về các Mô hình dữ liệu NoSQL. Phần 4: Những điểm nổi bật về transaction (một tiến trình xử lý có xác định điểm đầu và điểm cuối) trong cơ sở dữ liệu NoSQL. Phần 5: So sánh các Cơ sở dữ liệu NoSQL phổ biến. Cuối cùng, Phần 6: Là phần kết luận của bài viết.

II. TẦM QUAN TRỌNG CỦA NOSQL

A. Bối cảnh

Trong vài năm qua, SQL và NoSQL đã nổi lên những cuộc tranh cãi nảy lửa trên Internet. Thật ra các lập luận so sánh "SQL vs NoSQL" là đang nói về sự khác nhau của cở sở dữ liệu quan hệ và phi quan hệ. Do mô hình dữ liệu chuẩn hóa và thực thi các thuộc tính ACID nghiêm ngặt, cơ sở dữ liệu quan hệ truyền thống được coi là một cơ sở dữ liệu định hướng transaction dựa trên lược đồ cơ sở dữ liệu (thể hiện cấu trúc cơ sở dữ liệu dựa trên bảng, khóa chính,khóa ngoại). Lược đồ cơ sở dữ liệu được xác định nghiêm ngặt trước khi lưu trữ dữ liệu vào trong đó. Việc thay đổi lược đồ cơ sở dữ liệu sau khi đã thêm dữ liệu vào cơ sở dữ liệu sẽ gây lỗi cho cơ sở dữ liệu. Trong khi đó ở kỷ nguyên Big Data hiện nay, luôn có nhu cầu thêm các loại dữ liệu mới vào cơ sở dữ liệu để làm phong phú thêm ứng dụng. Và một lần nữa phải nhắc lại, giải pháp lưu trữ của cơ sở dữ liệu quan hệ làm ảnh hưởng lớn đến tốc độ xử lý và khả năng mở rộng. Các dịch vụ web như: Google, Amazon có tới hàng terabyte và petabyte dữ liệu được lưu trữ tại các trung tâm dữ liệu lớn của họ và phải đáp ứng lượng lớn nhu cầu đọc ghi với độ trễ là không đáng kể. Để mở rộng quy mô một cơ sở dữ liệu quan hệ, dữ liệu cần được phân phối trên nhiều máy chủ. Trong cơ sở dữ liệu quan hệ, trước khi cung cấp thông tin mà ứng dụng mong muốn, thông tin cần phải được thu thập và tổng hợp từ nhiều bảng. Tương tự, lúc ghi thông tin từ úng dụng vào cơ sở dữ liệu cần phải thực hiện phối hợp nhiều bảng. Vì vậy đối với bất kì ứng dụng nào việc đọc và ghi có thể tạo ra nút thắt cổ chai trong xử lý các bảng ở máy chủ. Trong cơ sở dữ liệu quan hệ, thực hiện lệnh "join" để thu thập dữ liệu sẽ làm chậm hệ thống đặc biệt là khi hàng triệu người dùng cùng thực hiện tra cứu với hàng ngàn bảng có hàng triệu dữ liệu. Các công ty dịch vụ web quy mô hàng đầu như: Google, Amazon, Yahoo nhận thấy đây là những nguyên nhân để họ phát triển cơ sở dữ liệu NOSQL để đáp ứng nhu cầu mở rộng và thực thi các nhu cầu với dữ liệu.

B. Các tính năng của NOSQL

Cơ sở dữ liệu NoSQL có thể không yêu cầu lược đồ cơ sở dữ liệu được xác định trước, thường chia tỷ lệ theo chiều ngang và tránh các thao tác nối. Do bản chất cấu trúc dữ liệu ít nghiêm ngặt hơn và khả năng tham gia phân tích các tập hợp con của NOSQL, cơ sở dữ liệu này được mô tả tốt hơn cơ sở dữ liệu quan hệ ở một số điểm như: khả năng mở rộng quy mô, tính linh hoạt và khả năng sao chép:

- Mở rộng quy mô: là việc đạt được hiệu suất cao trong môi trường dữ liệu phân tán bằng cách sử dụng nhiều máy chủ cùng sử lý dữ liệu. Cơ sở dữ liệu NoSQL cho phép phân phối dữ liệu trên một số lượng lớn máy với khả năng xử lý phân tán. Nhiều cơ sở dữ liệu NoSQL cho phép tự động phân phối dữ liệu tới các máy mới khi chúng được thêm vào cụm. Việc đánh giá khả năng mở rộng quy mô dựa trên khả năng mở rộng và tính linh hoạt của cơ sở dữ liệu

- Tính linh hoạt: Tính linh hoạt về mặt cấu trúc dữ liệu là không cần xác định lược đồ cho cơ sở dữ liệu. Cơ sở dữ liệu NoSQL không yêu cầu lược đồ được xác định trước. Điều này cho phép người dùng lưu trữ dữ liệu ở các cấu trúc khác nhau trong cùng một bảng cơ sở dữ liệu. Tuy nhiên, ngôn ngữ truy vấn cấp cao như SQL sẽ không được hỗ trợ trong hầu hết cơ sở dữ liệu NOSQL.

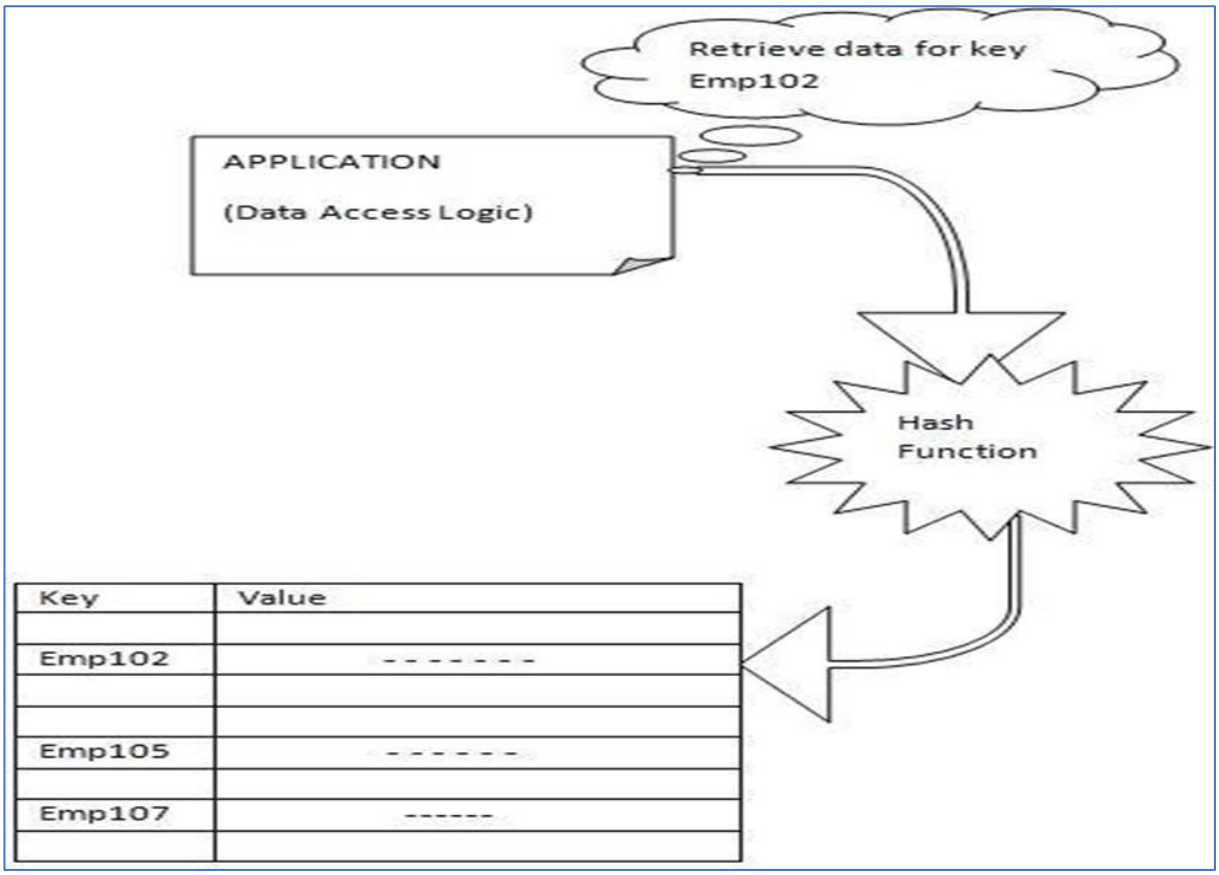
- Sao chép dữ liệu: là một trong những tính năng quan trọng của cơ sở dữ liệu NoSQL. Trong quá trình này, một bản sao của dữ liệu được phân phối cho các hệ thống khác nhau để dự phòng các sự cố và cân bằng mức sử lý dữ liệu ở các máy chủ. Tuy nhiên, trong quá trình này có khả năng làm mất tính nhất quán dữ liệu giữa các bản sao. Nhưng người ta tin rằng thông thường các bản sau cuối cùng rồi cũng sẽ nhất quán với nhau. Việc đánh giá khả năng sao chép dữ liệu dựa trên tính nhá quán và khả dụng.

III. CÁC MÔ HÌNH DỮ LIỆU NOSQL

A. Mô hình khóa-giá trị (Key-Value Data Stores)

Để xử lý đồng thời lượng truy cập cao vào cơ sở dữ liệu, mô hình cơ sở dữ liệu NoSQL được thiết kế như các kho lưu trữ dữ liệu dưới dạng khóa-giá trị. Đây là mô hình lưu trữ dữ liệu đơn giản nhất nhưng cũng mạnh mẽ nhất. Trong mô hình khóa-giá trị dữ liệu được lưu gồm một cặp khóa và giá trị duy nhất. Để lưu dữ liệu, một khóa được tạo bởi ứng dụng và giá trị được liên kết với khóa. Và cặp khóa-giá trị này được lưu vào kho lưu trữ dữ liệu. Các giá trị dữ liệu được lưu trữ trong các kho lưu trữ khóa-giá trị có thể có các tập thuộc tính và nó không rõ ràng đối với hệ thống quản lý cơ sở dữ liệu. Do đó, khóa là phương tiện duy nhất để truy cập các giá trị dữ liệu. Loại ràng buộc từ khóa đến giá trị phụ thuộc vào ngôn ngữ lập trình được sử dụng trong ứng dụng. Ứng dụng cần cung cấp khóa cho kho lưu trữ dữ liệu để truy xuất dữ liệu. Nhiều kho lưu trữ dữ liệu khóa-giá trị sử dụng hàm băm. Ứng dụng băm khóa để tìm ra vị trí của dữ liệu trong cơ sở dữ liệu nhanh chóng và hiệu quả hơn. Các khóa-giá trị được tập trung vào hàng. Điều đó có nghĩa là nó cho phép ứng dụng truy xuất dữ liệu cho các thực thể hoàn chỉnh.

Hình.1 mô tả việc truy xuất dữ liệu từ cơ sở dữ liệu khóa-giá trị. Ứng dụng đã chỉ định một khóa 'Emp102' cho kho lưu trữ dữ liệu để truy xuất dữ liệu. Ứng dụng sẽ sử dụng hàm băm thực hiện băm khóa để tìm vị trí của dữ liệu trong kho lưu trữ dữ liệu. Thiết kế của khóa phải hỗ trợ các truy vấn được thực hiện thường xuyên nhất trên kho lưu trữ dữ liệu. Hiệu quả của hàm băm, thiết kế khóa và kích thước của các giá trị được lưu trữ là những yếu tố ảnh hưởng đến hiệu suất của kho lưu trữ dữ liệu khóa-giá trị. Các hoạt động được thực hiện trên các kho lưu trữ dữ liệu như vậy hầu hết chỉ giới hạn ở các hoạt động đọc và ghi. Do tính đơn giản của kho lưu trữ dữ liệu khóa-giá trị, nó cung cấp cho người dùng phương tiện lưu trữ và tìm ghi dữ liệu nhanh nhất. Tất cả các loại NoSQL khác được xây dựng dựa trên tính đơn giản, khả năng mở rộng và hiệu suất của các kho lưu trữ dữ liệu khóa-giá trị. Các hệ thống cơ sở dữ liệu lưu trữ dữ liệu khóa-giá trị nổi bật như: Redis, Voldemort và Membase.



Hình 1. Một ví dụ về kho dữ liệu khóa-giá trị.

B. Mô hình cơ sở dữ liệu tài liệu (Document Oriente)

Ở mức trừu tượng, cơ sở dữ liệu hướng tài liệu tương tự như kho lưu trữ dữ liệu khóa-giá trị. Nó cũng giữ giá trị mà ứng dụng có thể đọc hoặc tìm nạp bằng cách sử dụng khóa. Một số cơ sở dữ liệu tài liệu tự động tạo khóa duy nhất trong khi tạo tài liệu mới. Một tài liệu trong cơ sở dữ liệu tài liệu là một đối tượng, là một tập hợp các trường được đặt tên. Tính năng phân biệt cơ sở dữ liệu hướng tài liệu với mô hình dữ liệu khóa-giá trị là tính minh bạch của dữ liệu do cơ sở dữ liệu nắm giữ. Do đó, khả năng truy vấn không bị hạn chế chỉ với khóa. Để hỗ trợ các trường hợp ứng dụng yêu cầu truy vấn cơ sở dữ liệu không chỉ dựa trên khóa của nó mà còn với các giá trị thuộc tính. Thông tin được lưu trữ ở định dạng di động và dễ hiểu như XML, BSON hoặc JSON.

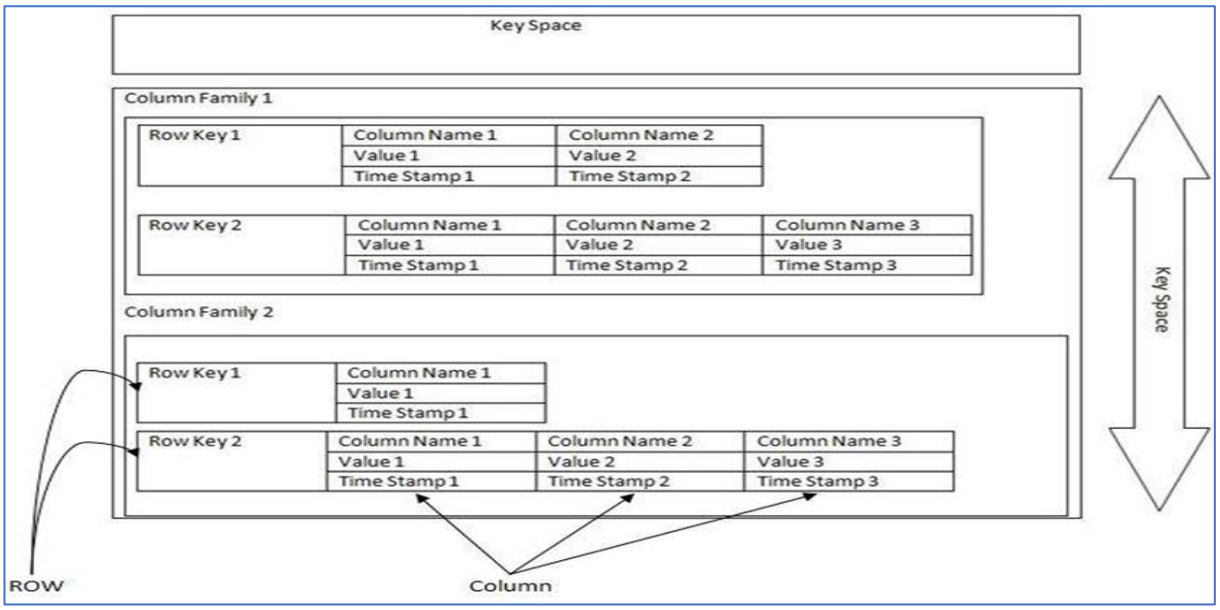
Trong Hình 2, cơ sở dữ liệu hướng tài liệu lưu trữ dữ liệu dưới dạng các cặp khóa-giá trị. Nhưng thuộc tính dữ liệu được lưu trữ trong cơ sở dữ liệu là rõ ràng đối với hệ thống không giống như cơ sở dữ liệu khóa-giá trị. Ứng dụng có thể truy vấn cơ sở dữ liệu không chỉ bằng khóa tức là ‘Employee ID’ mà còn với các trường được xác định trong tài liệu tức là FirstNm, LastNm, ‘Age’, v.v. Lưu trữ dữ liệu tài liệu là mô hình dữ liệu hiệu quả đối với các phần mềm phổ biến. Nhưng cũng có nhược điểm là hiệu suất và khả năng mở rộng thấp hơn một chút so với các kho lưu trữ dữ liệu khóa-giá trị. Một số kho lưu trữ tài liệu nổi bật nhất là Riak, MongoDB [11], CouchDB.

Key (Employee ID)	Document
Emp101	FirstNm:Jeet LastNm:Sethi Age:26 ..
...	...
....	....
Emp705	FirstNm:Meera LastNm:Patnaik Age:22 ...

Hình 2. Một ví dụ về kho lưu trữ dữ liệu tài liệu.

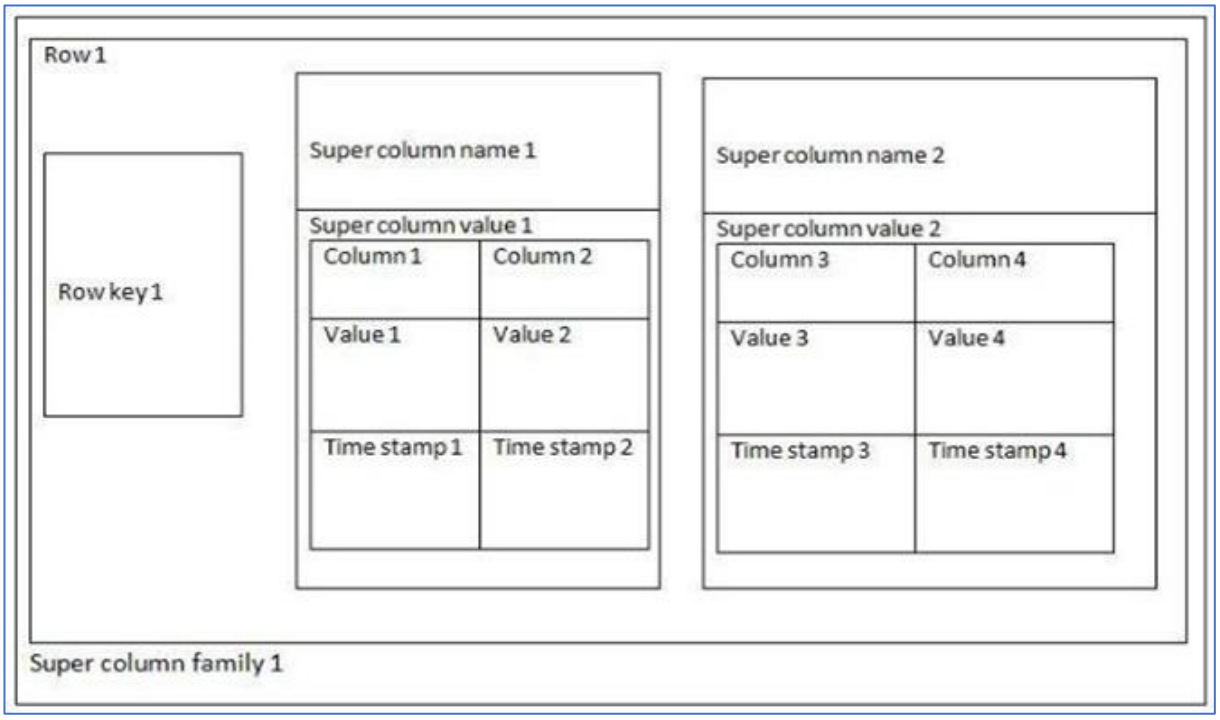
C. Mô hình cơ sở dữ liệu họ cột (Column Family)

Đôi khi một ứng dụng có thể muốn đọc hoặc tìm nạp một tập hợp con của các trường, tương tự như hoạt động chiếu của SQL. Kho dữ liệu dạng cột cho phép lưu trữ dữ liệu theo phương pháp tiếp cận lấy cột làm trung tâm. Mô hình dữ liệu hộ cột có các phân vùng khóa chính. Trong NoSQL, một phân vùng khóa chính được coi là một đối tượng giữ tất cả các họ cột của một nhóm lại với nhau. Đây là nhóm dữ liệu bên ngoài nhiều nhất trong kho dữ liệu. Mỗi phân vùng khóa chính được biết đến là một Bảng. Các họ cột được khai báo bởi các bảng này. Mỗi họ cột bao gồm số cột. Một hàng trong họ cột được cấu trúc dưới dạng tập hợp số cột tùy ý. Mỗi cột là một ánh xạ của một cặp khóa-giá trị. Trong sơ đồ này, các khóa là tên của các cột và chính các cột là các giá trị. Mỗi ánh xạ này được gọi là một ô. Mỗi hàng trong cơ sở dữ liệu họ cột được xác định bằng một khóa hàng duy nhất, được xác định bởi ứng dụng. Việc sử dụng các khóa hàng này giúp truy xuất dữ liệu nhanh hơn. Để tránh ghi đè các giá trị ô, một số cơ sở dữ liệu nhóm-cột phổ biến, đã tự động thêm thông tin dấu thời gian vào các cột riêng lẻ. Mỗi khi có bản cập nhật, nó sẽ tạo ra một phiên bản mới của các ô đã bị ảnh hưởng bởi thao tác cập nhật. Người đọc luôn luôn đọc được giá trị viết hoặc cập nhật cuối cùng. Khóa hàng, họ cột, cột và dấu thời gian tạo thành một khóa. Do đó, ánh xạ chính xác có thể được biểu diễn dưới dạng (khóa hàng, họ cột, cột, dấu thời gian) -- > giá trị. Một cấu trúc tổng quát của cơ sở dữ liệu họ cột đã được hiển thị trong Hình.3 như sau.



Hình 3. Lưu trữ dữ liệu họ cột.

Mô hình dữ liệu này đã được chấp nhận một cách phổ biến là "bản đồ được sắp xếp đa chiều thưa thớt, phân tán, nhất quán" [4]. Ưu điểm của việc sử dụng kho dữ liệu họ cột trên cơ sở dữ liệu truyền thống là xử lý các giá trị NULL. Trong cơ sở dữ liệu quan hệ, khi giá trị cho một thuộc tính không áp dụng cho một hàng cụ thể, null sẽ được lưu trữ. Trong khi trong cơ sở dữ liệu họ cột, cột có thể được xóa đơn giản cho hàng tương ứng trong trường hợp dữ liệu không có sẵn. Đó là lý do tại sao Google gọi nó là một cơ sở dữ liệu thưa thớt. Một trong những tính năng chính của cơ sở dữ liệu này là nó có thể được phân phối trong hàng tỷ ô nhớ trên hàng nghìn máy. Các ô được sắp xếp trên cơ sở các khóa hàng. Sắp xếp các khóa cho phép tìm kiếm dữ liệu cho một loạt các khóa. Vì dữ liệu trong loại mô hình như vậy được tổ chức thành một tập hợp các hàng và cột, nên cơ sở dữ liệu này tương tự như cơ sở dữ liệu quan hệ nhất. Nhưng giống như một cơ sở dữ liệu NOSQL, nó không cần bất kỳ lược đồ được xác định trước nào. Trong thời gian chạy, các hàng và cột có thể được thêm vào một cách linh hoạt nhưng đôi khi các họ cột phải được xác định trước, điều này dẫn đến mô hình dữ liệu kém linh hoạt hơn so với kho dữ liệu khóa-giá trị hoặc tài liệu. Các nhà phát triển phải hiểu dữ liệu được thu thập bởi ứng dụng và các khả năng truy vấn trước khi quyết định các họ cột. Cơ sở dữ liệu họ cột được thiết kế tốt cho phép ứng dụng đáp ứng phần lớn các truy vấn của nó bằng cách truy cập ít họ cột nhất có thể. So với cơ sở dữ liệu quan hệ nắm giữ lượng dữ liệu tương đương, kho dữ liệu họ cột có khả năng mở rộng và nhanh hơn. Nhưng hiệu suất và khả năng truy vấn của cơ sở dữ liệu họ cột ít khái quát hơn cơ sở dữ liệu quan hệ vì nó được thiết kế chỉ để hỗ trợ cho một tập hợp các truy vấn cụ thể. Hệ thống cơ sở dữ liệu Hbase [9] và Hypertable dựa trên mô hình dữ liệu được mô tả ở trên. Trong khi đó, một hệ thống cơ sở dữ liệu khác là Cassandra khác với mô hình dữ liệu họ cột, vì nó đang có một chiều mới được thêm vào được gọi là siêu cột [1]. Như thể hiện trong Hình 4, một siêu cột bao gồm nhiều cột. Một bộ sưu tập các siêu cột cùng với một khóa hàng tạo thành một hàng của họ siêu cột. Giống như trong các cột, tên siêu cột và tên cột phụ được sắp xếp. Siêu cột cũng là thực thể giá trị tên nhưng không có dấu thời gian.

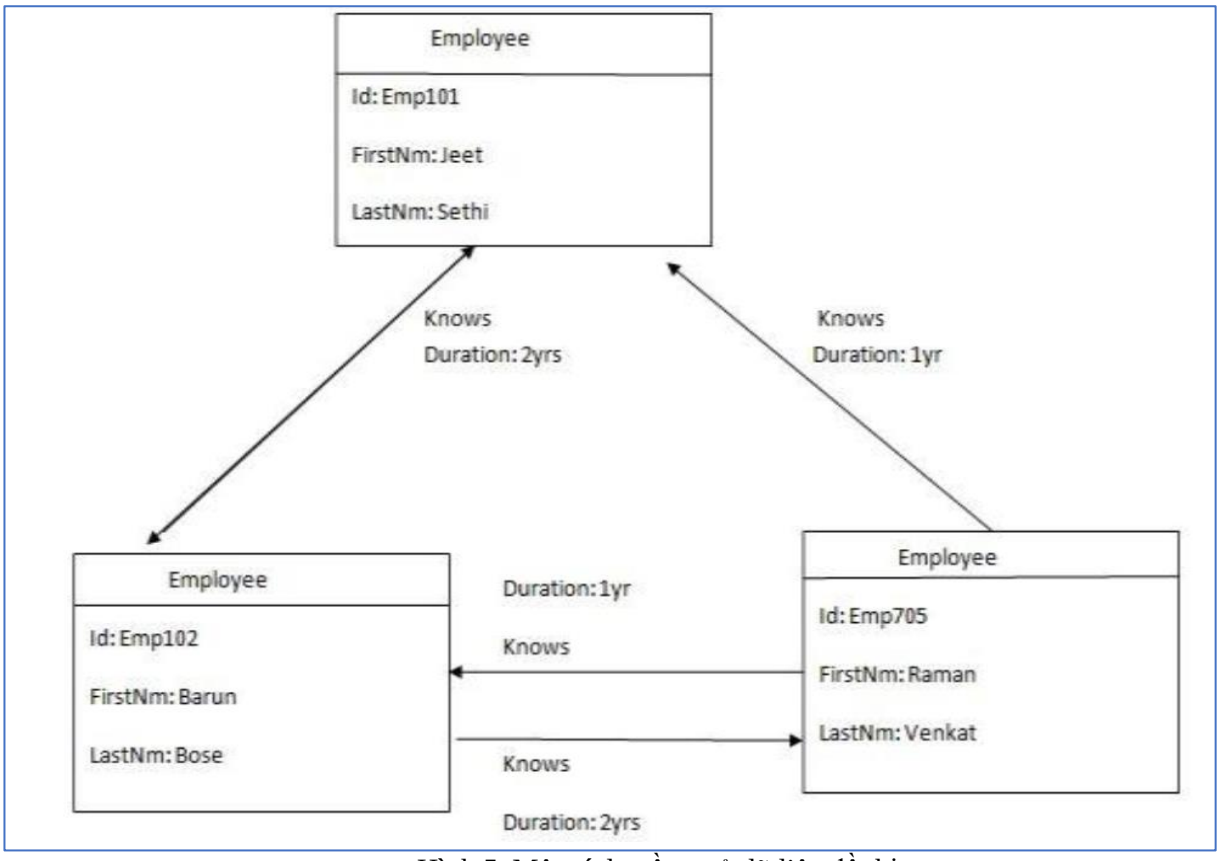


Hình 4. Lưu trữ dữ liệu họ cột (Cassandra).

D. Mô hình cơ sở dữ liệu đồ thị (Graph Database)

Cơ sở dữ liệu đồ thị được coi là chuyên gia của dữ liệu được liên kết cao. Do đó, nó xử lý dữ liệu liên quan đến một số lượng lớn các mối quan hệ [8]. Về cơ bản có ba trừu tượng cốt lõi của cơ sở dữ liệu đồ thị. Đó là các nút, các cạnh kết nối hai nút khác nhau và các thuộc tính. Mỗi nút chứa thông tin về một đối tượng. Các cạnh đại diện cho sự tồn tại của mối quan hệ giữa các thực thể. Mỗi mối quan hệ có một loại mối quan hệ và có hướng với điểm bắt đầu (nút) và điểm kết thúc. Điểm kết thúc có thể là một số nút khác với nút bắt đầu hoặc có thể là cùng một nút. Thuộc tính khóa-giá trị được liên kết không chỉ với các nút mà còn với các mối quan hệ. Các thuộc tính của các mối quan hệ cung cấp thông tin bổ sung về các mối quan hệ. Hướng của mối quan hệ xác định đường truyền từ nút này đến nút kia trong cơ sở dữ liệu đồ thị.

Hình 5 đại diện cho một phần của cơ sở dữ liệu ‘Employee’ được cấu trúc dưới dạng cơ sở dữ liệu đồ thị. Mỗi nút trong cơ sở dữ liệu đồ thị này đại diện cho một thực thể nhân viên. Các thực thể này có liên quan với nhau thông qua mối quan hệ của loại mối quan hệ "knows". Thuộc tính liên quan đến mối quan hệ là "Duration". Sự khác biệt chính giữa mô hình cơ sở dữ liệu đồ thị và cơ sở dữ liệu quan hệ là truy vấn dữ liệu. Thay vì sử dụng lệnh “join” có thể làm chậm hệ thống xử lý như trong cơ sở dữ liệu quan hệ, cơ sở dữ liệu đồ thị sử dụng phương pháp truyền qua phương thức. Trong khi truy vấn thông qua cơ sở dữ liệu đồ thị, một nút bắt đầu phải được chỉ định bởi ứng dụng. Truyền tải bắt đầu từ nút bắt đầu và tiến triển thông qua các mối quan hệ đến các nút được kết nối với nút bắt đầu, dựa trên một số quy tắc được xác định bởi logic ứng dụng. Phương thức truyền tải chỉ liên quan đến các nút có liên quan đến chỉ định của ứng dụng chứ không phải toàn bộ tập dữ liệu. Do đó, sự gia tăng lớn về số lượng các nút không ảnh hưởng nhiều đến tốc độ truyền tải. Mạng xã hội, khai phá dữ liệu, quản lý mạng và tính toán tìm đường đi trên bản đồ là một số lĩnh vực mà cơ sở dữ liệu đồ thị đã được sử dụng rộng rãi. Neo4j [8], GraphDB là cơ sở dữ liệu đồ thị phổ biến được sử dụng ngày nay.



Hình 5. Một ví dụ về cơ sở dữ liệu đồ thị.

IV. TRANSACTION (tiến trình xử lý cơ sở dữ liệu được tạo bởi một chương trình đang thực thi)

Khi chúng ta nói về SQL và NoSQL, sự cạnh tranh thực sự không phải giữa các cơ sở dữ liệu mà là sự so sánh giữa các mô hình transation của cả hai cơ sở dữ liệu. Transation được định nghĩa là đơn vị logic của một quá trình xử lý cơ sở dữ liệu được tạo bởi một chương trình đang thực thi. Transation của cơ sở dữ liệu SQL dựa trên các thuộc tính ACID nghiêm ngặt. Trong đó ACID là tên viết tắt của Atomicity (Tính nguyên tử: Một giao dịch có nhiều thao tác khác biệt thì hoặc là toàn bộ các thao tác hoặc là không một thao tác nào được hoàn thành), Consistence (Tính nhất quán), Isolation (Tính độc lập) và Durability (Tính bền vững). Nhưng các nhà thiết kế cơ sở dữ liệu NoSQL đã đưa ra một quyết định, rằng thuộc tính ACID là quá nghiêm ngặt đối với nhu cầu của dữ liệu lớn. Do đó, giáo sư Eric Brewer vào năm 2000 đã đưa ra một định lý mới được gọi là định lý CAP [2]. CAP là tên viết tắt của Consistency (Tính nhất quán), Availability (Tính khả dụng) và Partition tolerance (Dung sai phân vùng hay khả năng chịu đựng của phân vùng). Định lý nói rằng các nhà thiết kế có thể đạt được bất kỳ hai trong số các tính chất này cùng một lúc trong môi trường phân tán. Các nhà thiết kế có thể đảm bảo Tính nhất quán và Tính khả dụng thì phải đánh đổi dung sai Phân vùng, tức là cơ sở dữ liệu dựa trên CA. Nếu nhà thiết kế yêu cầu tính khả dụng và dung sai phân vùng thì phải đánh đổi Tính nhất quán, thì đó là cơ sở dữ liệu dựa trên AP. Và nếu đảm bảo tính nhất quán và dung sai phân vùng thì phải đánh đổi Tính khả dụng thì cơ sở dữ liệu dựa trên CP. Transation của NoSQL có thể được phân loại như sau.

- Quan tâm đến tính nhất quán và tính khả dụng (CA): Loại hệ thống cơ sở dữ liệu này đảm bảo ưu tiên của nó nhiều hơn đối với tính khả dụng và tính nhất quán của dữ liệu bằng cách sử dụng cách tiếp cận sao chép [2]. Một phần của cơ sở dữ liệu không bận tâm về khả năng chịu đựng của phân vùng. Trong trường hợp xảy ra phân vùng giữa các nút, dữ liệu sẽ không đồng bộ. Hệ thống cơ sở dữ liệu quan hệ, Vertica và Greenplum thuộc loại cơ sở dữ liệu như vậy.

- Quan tâm đến tính nhất quán và khả năng chịu đựng phân vùng (CP): Ưu tiên của hệ thống cơ sở dữ liệu là đảm bảo tính nhất quán của dữ liệu. Nhưng nó không hỗ trợ cho tính khả dụng

tốt. Dữ liệu được lưu trữ trong các nút phân tán [2].Khi một nút gặp sự cố, dữ liệu sẽ không khả dụng để duy trì tính nhất quán giữa các nút. Nó duy trì khả năng chịu đựng của phân vùng bằng cách ngăn chặn việc đồng bộ hóa lại dữ liệu. Hypertable, BigTable, HBase là vài hệ thống cơ sở dữ liệu quan tâm đến CP.

- Quan tâm đến tính khả dụng và khả năng chịu đựng của phân vùng (AP): Ưu tiên của hệ thống cơ sở dữ liệu đó là chủ yếu đảm bảo tính sẵn có của dữ liệu và khả năng chịu đựng của phân vùng. Ngay cả khi có lỗi giao tiếp giữa các nút, các nút vẫn ở trạng thái hoạt động. Sau khi phân vùng được giải quyết, quá trình đồng bộ hóa lại dữ liệu sẽ diễn ra nhưng không đảm bảo tính nhất quán. Riak, CouchDB, KAI là một số cơ sở dữ liệu tuân theo nguyên tắc này.

Sau đó, định lý CAP được mở rộng thành PACELC [1]. PACELC là tên viết tắt của Partition (phân vùng), Availability (tính khả dụng), consistency (tính nhất quán), else (khác), Latency (độ trễ), consistency (tính nhất quán). Theo mô hình này, sự cân bằng giữa tính khả dụng và tính nhất quán không chỉ dựa trên dung sai phân vùng mà còn phụ thuộc vào sự tồn tại của phân vùng mạng. Nó cho thấy độ trễ là một trong những yếu tố quan trọng, vì hầu hết các hệ thống cơ sở dữ liệu phân tán sử dụng công nghệ sao chép để đảm bảo tính khả dụng.Sau đó, eBay đã giới thiệu một định lý mới được gọi là định lý BASE [3]. [1]. c BASE nhằm đạt được tính khả dụng thay vì tính nhất quán của cơ sở dữ liệu. BASE là chữ viết tắt của Basically Available, Soft State và Eventually Consistent.

- Basically Available (Tính khả dụng cơ bản): Tính khả dụng cơ bản là ngay cả khi một phần của cơ sở dữ liệu trở nên không khả dụng, các phần khác của cơ sở dữ liệu vẫn tiếp tục hoạt động như mong đợi. Trong trường hợp nút bị lỗi, hoạt động tiếp tục trên bản sao dữ liệu được lưu trữ trong một số nút khác.

- Soft State (Trạng thái hữu ích): Trạng thái mềm là dựa trên cơ sở tương tác của người dùng, dữ liệu có thể phụ thuộc vào thời gian. Những dữ liệu này cũng có thể hết hạn sau một khoảng thời gian nhất định. Do đó, để giữ cho dữ liệu có liên quan trong một hệ thống, nó phải được cập nhật hoặc truy cập.

- Eventually Consistent (Tính nhất quán cuối cùng): Tính nhất quán cuối cùng cho biết sau bất kỳ cập nhật dữ liệu nào, dữ liệu có thể trở nên nhất không quán trên toàn bộ hệ thống nhưng cuối cùng nó sẽ trở nên nhất quán theo thời gian. Do đó, dữ liệu được cho là nhất quán trong tương lai.

V. SO SÁNH CƠ SỞ DỮ LIỆU NOSQL

Không có bất kỳ quy tắc nào để quyết định cơ sở dữ liệu NoSQL nào là tốt nhất cho doanh nghiệp. Mô hình kinh doanh, chiến lược, chi phí và nhu cầu về mô hình giao dịch là một số yếu tố quan trọng mà doanh nghiệp nên cân nhắc khi lựa chọn cơ sở dữ liệu. Sau đây là một số thông tin có thể giúp ích cho việc lựa chọn cơ sở dữ liệu cho một doanh nghiệp.

- Nếu các ứng dụng chỉ đơn giản là lưu trữ và truy xuất các mục dữ liệu không rõ ràng vào hệ thống quản lý cơ sở dữ liệu và các đối tượng nhị phân bằng cách sử dụng khóa làm mã định danh, thì kho lưu trữ khóa-giá trị là lựa chọn tốt nhất. Nhưng nếu ứng dụng muốn truy vấn cơ sở dữ liệu với một số giá trị thuộc tính khác với khóa, nó không thể thực hiện được. Ngoài ra, cũng không thể cập nhật hoặc đọc một trường riêng lẻ trong kho lưu trữ khóa bản ghi-giá trị.
- Khi các ứng dụng chọn lọc hơn và cần lọc các bản ghi dựa trên các trường không phải khóa hoặc truy xuất hoặc cập nhật các trường riêng lẻ trong một bản ghi như nó, thì cơ sở dữ liệu tài liệu là một giải pháp hiệu quả. Kho dữ liệu tài liệu cung cấp khả năng truy vấn tốt hơn so với kho dữ liệu khóa-giá trị.
- Khi các ứng dụng cần lưu trữ các bản ghi với hàng trăm hoặc hàng nghìn trường, nhưng truy xuất một tập hợp con của các trường đó trong hầu hết các truy vấn mà nó thực hiện, trong trường hợp đó, kho dữ liệu nhóm-cột là một lựa chọn hiệu quả. Kho dữ liệu như vậy phù hợp với các bộ dữ liệu lớn có quy mô cao.

• Nếu các ứng dụng cần lưu trữ và xử lý thông tin về dữ liệu được liên kết nhiều với mối quan hệ rất phức tạp giữa các thực thể, cơ sở dữ liệu đồ thị là lựa chọn tốt nhất.Trong cơ sở dữ liệu đồ thị, các thực thể và mối quan hệ giữa các thực thể được xử lý với tầm quan trọng như nhau.

Bảng 1 trình bày danh sách các cơ sở dữ liệu với các mô hình dữ liệu tương ứng của chúng, cùng với mô hình Transation và ngôn ngữ truy vấn được sử dụng bởi các cơ sở dữ liệu này. Cassandra được cung cấp bởi Facebook, HBase [9] được cung cấp bởi Google, DynamoDB được cung cấp bởi Amazon là một vài cơ sở dữ liệu khác được phát triển bởi các công ty khác nhau nhằm đáp ứng nhu cầu lưu trữ dữ liệu cao của họ. Mặt khác, các hệ thống cơ sở dữ liệu như Neo4j, Riak và MongoDB được phát triển để phục vụ các tổ chức khác nhau. Về mô hình Transation, hầu hết các cơ sở dữ liệu như DynamoDB, Riak, Cassandra và Voldermort ưu tiên tính khả dụng hơn tính nhất quán. Trong khi Tokyo Cabinet, Hbase thích tính nhất quán hơn tính khả dụng. Cơ sở dữ liệu NoSQL được thiết kế để xử lý xử lý dữ liệu khối lượng lớn, ngoại trừ một số hệ thống hỗ trợ của RDBMS như truy vấn đặc biệt. Mặc dù nhiều cơ sở dữ liệu NoSQL được đề cập trong Bảng. 1 hỗ trợ truy vấn ad-hoc nhưng trình độ chuyên môn lập trình viết truy vấn cần cao hơn nhiều so với cơ sở dữ liệu quan hệ.

BẢNG I. SO SÁNH CÁC CƠ SỞ DỮ LIỆU NOSQL KHÁC NHAU

Cơ sở dữ liệu	**Mô hình dữ liệu**	**Transaction Model (CAP)**	**Truy vấn Ad-HOC**
DynamoDB	Key-value (khóa-giá tri)	AP	Built in API
Riak	Key-value (khóa-giá tri)	AP	CorrugatedIro n
Voldermort	Key-value	AP	Không hỗ trợ
Tokyo Cabinet	Key-value (khóa-giá tri)	PC	Không hỗ trợ
CouchDB	Document (tài liệu)	AP	Cloudant, Lucene
MongoDB	Document (tài liệu)	AP	BSON based format
RavenDB	Document (tài liệu)	ACID	Built in, Limited
Cassandra	Column-family (họ cột)	AP	HIVE, PIG
Hbase	Column-family (họ cột)	PC	HIVE, PIG
Neo4j	Graph (đồ thị)	CA	Chyper

VI. PHẦN KẾT LUẬN

Trong lĩnh vực cơ sở dữ liệu, cơ sở dữ liệu NoSQL được coi là khá mới. Tuy nhiên NOSQL đang được phát triển trên lý thuyết đã biết và hiện có. .Hệ thống cơ sở dữ liệu NoSQL vẫn còn những hạn chế khác nhau. Không có tiêu chuẩn chung cũng như bất kỳ ngôn ngữ truy vấn phổ biến và quen thuộc nào để truy vấn cơ sở dữ liệu NoSQL. Mỗi cơ sở dữ liệu hoạt động theo một cách khác nhau và thực hiện mọi truy vấn khác nhau. Các cơ sở dữ liệu này chưa hoàn toàn hoàn thiện và tối ưu nên chúng sẽ không ngừng phát triển. Cơ sở dữ liệu NoSQL không hỗ trợ các thuộc tính ACID nghiêm ngặt, do đó không có gì đảm bảo rằng tất cả dữ liệu sẽ được ghi thành công vào kho dữ liệu. Bài báo này mô tả những hạn chế của cơ sở dữ liệu quan hệ khi so với từng mô hình dữ liệu NoSQL khác nhau. Vì không có sẵn đánh giá để tìm công cụ phù hợp, bài báo này chỉ so sánh ưu điểm và khuyết điểm của từng mô hình dữ liệu. Hạn chế của cơ sở dữ liệu NoSQL và việc sử dụng nó trong môi trường điện toán đám mây là những lĩnh vực cần nghiên cứu chi tiết trong tương lai.

NGUỒN THAM KHẢO

[1] Maria Indrawan, “Database Research: Are We At A Crossroad?,” 15th International Conference on Network-Based Information Systems, pp. 45-48, 2012.

[2] Jing Han, Haihong E, Guan Le,Jian Du, “Survey on NoSQL Database,” IEEE, pp. 363-366, 2011.

[3] Shalini R., Savita G., Subramanian A., “Comparison of Cloud Database: Amazon‟s SimpleDB and Google‟s Bigtable,” International Conference on Recent Trends in Information Systems, IEEE, pp. 165- 168, 2011.

[4] R Hecht, S Jablonski, “NoSQL Evaluation,” International Conference on Cloud and Service Computing, IEEE, pp. 336-338, 2011.

[5] Alexandru Boicea, Florin Radulescu, Laura Ioana Agapin, “MongoDB vs Oracle - database comparison,” Third International Conference on Emerging Intelligent Data and Web Technologies, 2012.

[6] Jing Han, Meina Song and Junde Song, “A Novel Solution of Distributed Memory NoSQL Database for Cloud Computing,” 10th IEEE/ACIS International Conference on Computer and Information Science, 2011.

[7] Guoxi Wang,Jianfeng Tang, “The NoSQL Principles and Basic Application of Cassandra Model,” IEEE, pp.1332-1333, 2012.

[8] Neo4j, http://neo4j.org.

[9] Hbase, http://hbase.apache.org.

[10] Mahdi Negahi Shirazi,Ho Chin Kuan,Hossein Dolatabadi, “Design Patterns to Enable Data Portability between Clouds‟ Databases,” 12th International Conference on Computational Science and Its Applications, pp. 117-118, 2012.

[11] Mongodb, http://www.mongodb.org.