**A Study of NoSQL Database**

Tóm tắt—Một số ứng dụng của dịch vụ web 2.0 cần xử lý dữ liệu lớn. Những vấn đề quang trọng phải đươc xem xét, đặc biệt là đối với các ứng dụng yêu cầu quy mô dữ liệu người dùng cao và tính đồng thời cao. các nhà thiết kế cân nhắc và xem xét những vấn đề quan trọng để đưa ra một nhóm cơ sở dữ liệu mới, thường được gọi là NoSQL. Nhu cầu ngày càng tăng về điện toán đám mây và sự phát triển của Internet thúc đẩy phong trào NoSQL. Tài liệu này đề cập đến các tính năng và mô hình dữ liệu của cơ sở dữ liệu NoSQL được sử dụng trong môi trường điện toán đám mây cùng với sức mạnh và giới hạn của từng mô hình. Ngoài ra, bài báo này nói về phân loại cơ sở dữ liệu NoSQL dựa trên định lý CAP.

Từ khóa—NoSQL; MŨ; Định hướng tài liệu; dạng cột; Dữ liệu lớn.

1. **Giới thiệu**

Lưu trữ hiệu quả và truy xuất dữ liệu với tính khả dụng và khả năng mở rộng là mục đích chính của cơ sở dữ liệu NoSQL. NoSQL không đại diện cho SQL; nó có nghĩa là "KHÔNG CHỈ SQL" [10]. Cơ sở dữ liệu NoSQL chỉ 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 ra đời của nhiều cơ sở dữ liệu phi quan hệ như MongoDB [11], Hbase [9], Neo4j [8] trong vài năm qua. Tùy thuộc vào yêu cầu và chiến lược kinh doanh, nhà cung cấp đám mây có thể đi kèm với bất kỳ loại cơ sở dữ liệu nào. Vẫn còn một số nhà thiết kế cơ sở dữ liệu tiền quan hệ cho rằng cơ sở dữ liệu NoSQL không đủ hiệu quả trong việc xử lý tính toàn vẹn của dữ liệu. Bài viết này được sắp xếp như sau: Phần 2 mô tả tầm quan trọng của cơ sở dữ liệu NoSQL. Phần 3 nêu bật trên các mô hình dữ liệu NoSQL. Phần 4 nêu bật về giao dịch trong cơ sở dữ liệu NoSQL. Phần 5 làm sáng tỏ sự so sánh cho cơ sở dữ liệu NoSQL. Cuối cùng, chúng tôi kết luận bài viết này trong Phần 6

1. **Vấn đề quan trọng của NOSQL**
2. **Tiểu sử**

Trong vài năm qua, SQL và NoSQL đã nổi lên như một cuộc tranh cãi nảy lửa trên Internet. Đối số "SQL vs NoSQL", thực sự nói về cơ sở dữ liệu quan hệ so với 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 giao dịch dựa trên lược đồ. Nó yêu cầu một lược đồ được xác định trước nghiêm ngặt trước khi lưu trữ dữ liệu vào đó. Xác định lại lược đồ trong trường hợp thay đổi trong tương lai, một lần sau khi dữ liệu được chèn vào cơ sở dữ liệu là gây gián đoạn. Trong khi đó, trong kỷ nguyên của Dữ liệu lớn, nhu cầu liên tục thêm các loại dữ liệu mới để làm phong phú thêm các ứng dụng. Một lần nữa, giải pháp lưu trữ của cơ sở dữ liệu quan hệ có thể tạo ra tác động lớn đến tốc độ và khả năng mở rộng. Các dịch vụ web như Amazon và Google có terabyte và petabyte dữ liệu được lưu trữ trong các trung tâm dữ liệu lớn của họ và phải đáp ứng các yêu cầu đọc-ghi lớn mà không có độ trễ đáng chú ý. Để mở rộng cơ sở dữ liệu quan hệ, dữ liệu cần được phân phối trên nhiều máy chủ. Trước khi cung cấp cho ứng dụng, thông tin mong muốn phải được thu thập từ nhiều bảng và kết hợp. Tương tự như vậy trong khi ghi dữ liệu cũng vậy; nó phải được thực hiện trên nhiều bàn một cách phối hợp. Đối với bất kỳ ứng dụng nào, nó có thể là một nút cổ chai để xử lý các bảng trên nhiều máy chủ. Trong cơ sở dữ liệu quan hệ "tham gia‟ hoạt động làm chậm hệ thống để thu thập dữ liệu, đặc biệt là khi hàng triệu người dùng đang thực hiện tra cứu đối với các bảng có hàng triệu hàng dữ liệu. Các dịch vụ web quy mô lớn như Google, Amazon, Yahoo, Facebook nhận thấy đây là những trường hợp để phát triển cơ sở dữ liệu phi quan hệ của riêng họ nhằm đáp ứng nhu cầu về khả năng mở rộng và hiệu suất.

**B. Các tính năng của NoSQL**

Cơ sở dữ liệu NoSQL có thể không yêu cầu lược đồ bảng được xác định trước, thường chia tỷ lệ theo chiều ngang và thường tránh các hoạt động nối. Do bản chất lược đồ ít hơn và sự tham gia của phân tích tập hợp con nhỏ hơn của hệ thống NoSQL, cơ sở dữ liệu này có thể được mô tả tốt hơn dưới dạng kho dữ liệu có cấu trúc. Ba tính năng cơ bản quan trọng của cơ sở dữ liệu NoSQL là mở rộng quy mô, cấu trúc và sao chép dữ liệu linh hoạt, được giải thích như sau.

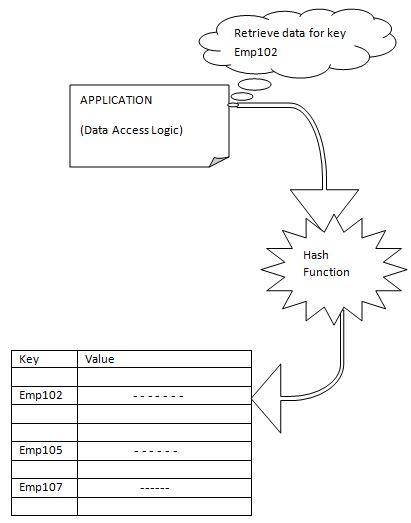
* Scale-out: Mở rộng quy mô đề cập đến việc đạt được hiệu suất cao trong môi trường phân tán bằng cách sử dụng nhiều máy móc đa năng. Cơ sở dữ liệu NoSQL cho phép phân phối dữ liệu trên một số lượng lớn máy có tải xử lý phân tán. Nhiều cơ sở dữ liệu NoSQL cho phép tự động phân phối dữ liệu đến các máy mới khi chúng được thêm vào cụm. Scale-out được đánh giá về khả năng mở rộng và độ đàn hồi
* Flexibility: Tính linh hoạt về cấu trúc dữ liệu nói rằng không cần phải 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 của các cấu trúc khác nhau trong cùng một bảng cơ sở dữ liệu. Tuy nhiên, hỗ trợ cho các ngôn ngữ truy vấn cấp cao như SQL không được hỗ trợ bởi hầu hết các cơ sở dữ liệu NoSQL.
* Data Replication: Một trong những tính năng của cơ sở dữ liệu NoSQL là sao chép dữ liệu. Trong quá trình này, một bản sao của dữ liệu được phân phối cho các hệ thống khác nhau để đạt được sự dư thừa và phân phối tải. Tuy nhiên, có khả năng mất tính nhất quán dữ liệu giữa các bản sao. Nhưng người ta tin rằng đôi khi sự nhất quán này cuối cùng có thể đạt được. Tính nhất quán và tính khả dụng là những yếu tố để đánh giá sao chép [3].

1. MÔ HÌNH DỮ LIỆU NOSQL

Đây là một số loại mô hình cơ sở dữ liệu NoSQL được thảo luận như sau [1][2][4].

1. **Kho dữ liệu khóa-giá trị**

Để xử lý quyền truy cập đồng thời cao vào cơ sở dữ liệu, danh mục NoSQL được thiết kế là các kho lưu trữ khóa-giá trị. Nó là kho dữ liệu đơn giản nhất, vẫn là mạnh mẽ nhất. Trong một kho lưu trữ khóa-giá trị, mỗi dữ liệu bao 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 gửi đến kho dữ liệu. Các giá trị dữ liệu được lưu trữ trong các kho lưu trữ keyvalue có thể có các bộ thuộc tính động được đính kèm với nó và không rõ ràng 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. Một ứng dụng cần cung cấp khóa cho kho dữ liệu để truy xuất dữ liệu. Nhiều kho dữ liệu khóa-giá trị sử dụng hàm băm. Ứng dụng băm khóa và tìm ra vị trí của dữ liệu trong cơ sở dữ liệu. Kho dữ liệu khóa-giá trị được tập trung vào hàng. Có nghĩa là nó cho phép ứng dụng truy xuất dữ liệu cho các thực thể hoàn chỉnh.

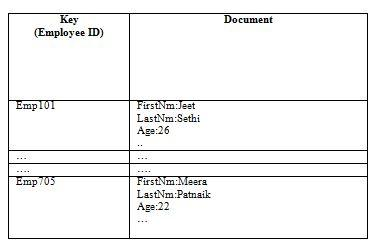
Hình 1 mô tả việc truy xuất dữ liệu từ cơ sở dữ liệu khóa-giá trị. Ứng dụng đã chỉ định một khóa 'Emp102' cho kho dữ liệu để truy xuất dữ liệu. Sử dụng hàm băm, ứng dụng băm khóa để theo dõi vị trí của dữ liệu trong kho dữ liệu. Thiết kế của khóa sẽ hỗ trợ các truy vấn thường xuyên nhất được kích hoạt trên kho 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 dữ liệu keyvalue. Các hoạt động được thực hiện trên các kho dữ liệu như vậy chủ yếu chỉ giới hạn ở các hoạt động đọc và ghi. Do sự đơn giản của kho dữ liệu khóa-giá trị, nó cung cấp cho người dùng phương tiện lưu trữ và tìm nạp dữ liệu nhanh nhất. Tất cả các danh mục khác của NoSQL được xây dựng dựa trên sự đơn giản, khả năng mở rộng và hiệu suất của kho dữ liệu khóa-giá trị. Các hệ thống cơ sở dữ liệu Redis, Voldemort và Membase là những ví dụ về các kho dữ liệu keyvalue nổi bật. 

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

1. **Kho dữ liệu định hướng tài liệu**

Ở cấp độ trừu tượng, cơ sở dữ liệu định hướng tài liệu tương tự như kho dữ liệu khóa-giá trị. Nó cũng giữ giá trị, mà một ứng dụng có thể đọc hoặc tìm nạp bằng cách sử dụng một khóa. Một số cơ sở dữ liệu tài liệu tự động tạo khóa duy nhất trong khi tạo một tài liệu mới. Tài liệu trong cơ sở dữ liệu tài liệu là một thực thể, là một tập hợp các trường được đặt tên. Tính năng phân biệt cơ sở dữ liệu hướng tài liệu với kho 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 tình huống trong đó 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, có thể chuyển đổi sang cơ sở dữ liệu tài liệu. Một tài liệu cần phải tự mô tả trong cơ sở dữ liệu hướng tài liệu. Thông tin được lưu trữ ở định dạng di động và được hiểu rõ như XML, BSON hoặc JSON.

Như trong Hình 2, cơ sở dữ liệu tài liệu lưu trữ dữ liệu dưới dạng các cặp khóa-giá trị. Nhưng dữ liệu được lưu trữ trong cơ sở dữ liệu là minh bạch 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 'ID nhân viên' mà còn với các trường được xác định trong tài liệu, tức là FirstNm, LastNm, age etc, v.v. Kho dữ liệu tài liệu là cách tiếp cận hiệu quả để mô hình hóa dữ liệu dựa trên các sự cố phần mềm phổ biến. Nhưng nó phải trả giá bằng hiệu suất và khả năng mở rộng thấp hơn một chút so với các kho dữ liệu khóa-giá trị. Rất ít kho tài liệu nổi bật nhất là Riak, MongoDB [11], CouchDB.

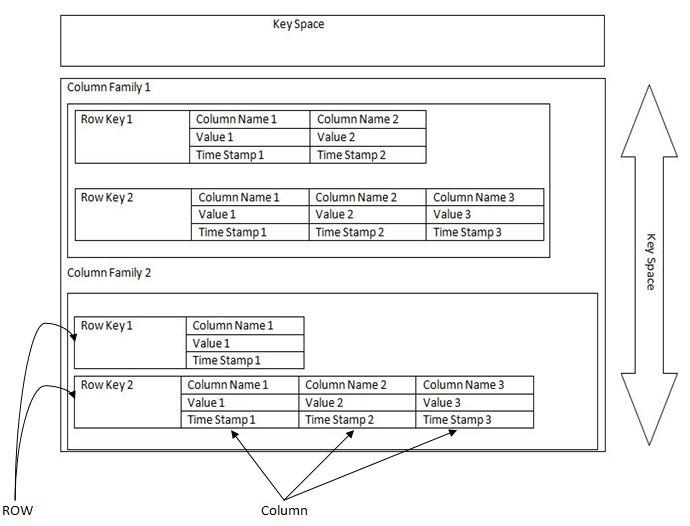


Hình 2. Một ví dụ về kho dữ liệu tài liệu.

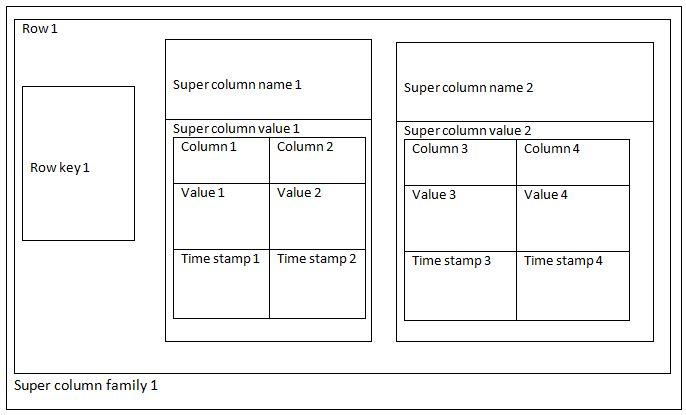
1. **Kho dữ liệu dạng cột**

Đôi khi một ứng dụng có thể muốn đọc hoặc tìm nạp một tập hợp con các trường, tương tự như hoạt động chiếu của SQL. Kho dữ liệu họ cột cho phép lưu trữ dữ liệu theo phương pháp tiếp cận lấy cột làm trung tâm. Kho dữ liệu họ cột phân vùng không gian chính. Trong NoSQL, một không gian khóa được coi là một đối tượng giữ tất cả các họ cột của một thiết kế lại với nhau. Đây là nhóm dữ liệu bên ngoài nhiều nhất trong kho dữ liệu. Mỗi phân vùng của không gian khóa 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 bản đồ này, các phím 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 phím 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, hãy 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. Luôn luôn người đọc đọc giá trị được viết hoặc cam kết lần cuối. 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 trình bày trong Hình 3 như sau.

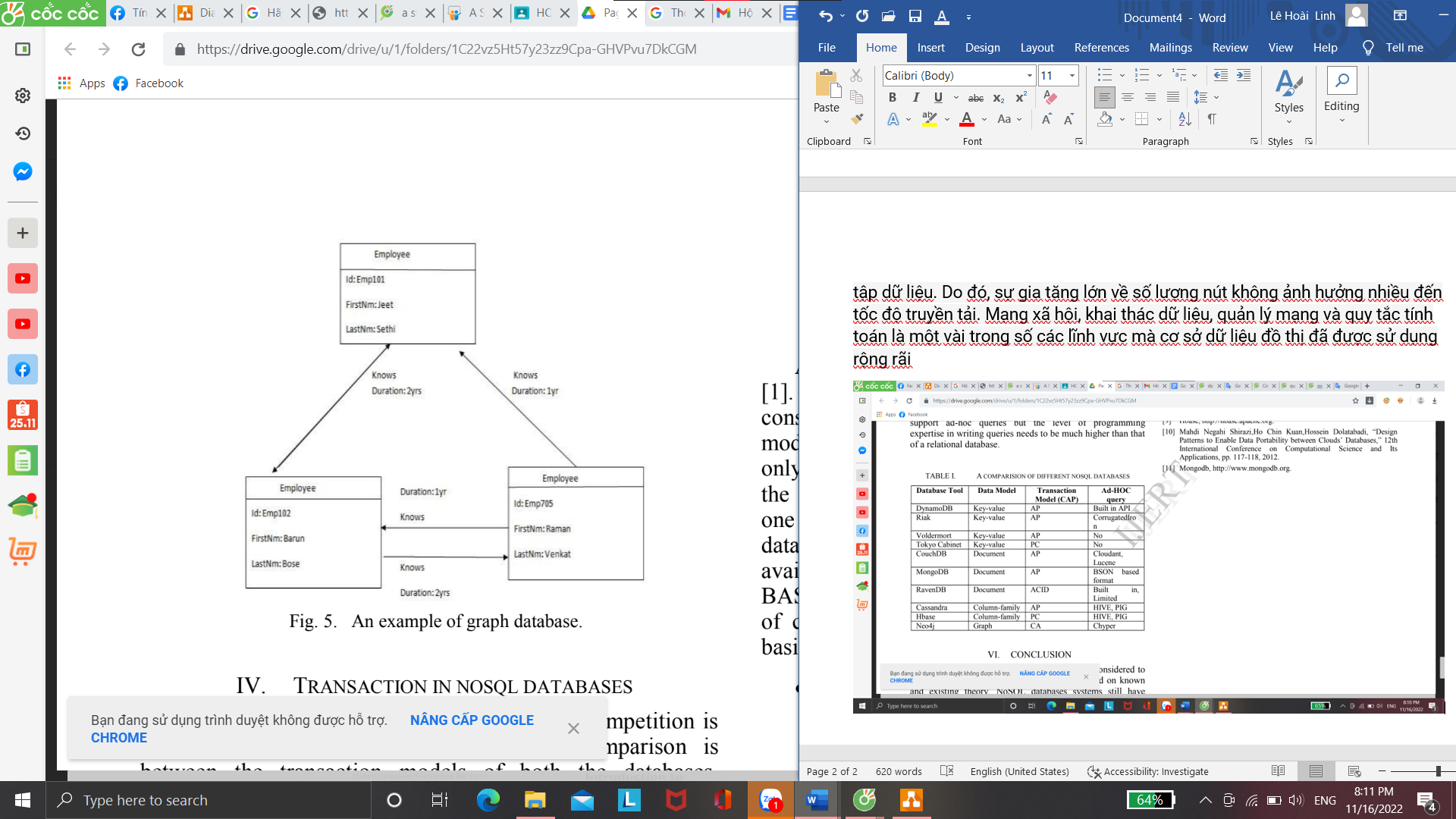
Hình 3. Lưu trữ dữ liệu dạng cột.

Mô hình dữ liệu này đã được chấp nhận rộng rãi là "bản đồ được sắp xếp đa chiều thưa thớt, phân tán, nhất quán". Một lợi thế của việc sử dụng kho dữ liệu họ cột so với cơ sở dữ liệu truyền thống là xử lý các giá trị NULL. Trong cơ sở dữ liệu quan hệ, khi một 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 gia đình 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 tế bào trên hàng nghìn máy. Các ô được sắp xếp trên cơ sở các phím 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 quan hệ, nó không cần bất kỳ lược đồ được xác định trước nào. Trong thời gian chạy, các hàng và cột có thể được thêm linh hoạt nhưng đôi khi các họ cột phải được xác định trước, điều này khiến kho dữ liệu kém linh hoạt hơn so với kho dữ liệu key-value hoặc tài liệu. Các nhà phát triển nên 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 gia đình 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 hơn và nhanh hơn. Nhưng hiệu suất đi kèm với cái giá của cơ sở dữ liệu ít được khái quát hóa hơn so với cơ sở dữ liệu quan hệ vì nó được thiết kế để hỗ trợ cho một nhóm truy vấn cụ thể. Hệ thống cơ sở dữ liệu Hbase và Hypertable dựa trên mô hình dữ liệu được mô tả ở trên. Trong khi đó, một hệ thống cơ sở dữ liệu khác Cassandra khác với mô hình dữ liệu, vì nó đang có một thứ nguyên mới được thêm vào được gọi là siêu cột. Như trong Hình 4, một cột siêu bao gồm nhiều cột. Một bộ sưu tập các siêu cột cùng với một phím hàng tạo thành một hàng của họ siêu cột. Như trong các cột, tên siêu cột và tên cột phụ được sắp xếp. Cột siêu cũng là một thực thể tên-giá trị nhưng không có dấu thời gian. 

Hình 4. Kho dữ liệu dạng cột(Cassandra).

1. **Cơ sở dữ liệu đồ thị**

Graph databases 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. Graph databases về cơ bản có ba bản tóm tắt cốt lõi. Đây là các nút, các cạnh kết nối hai nút khác nhau và các thuộc tính. Mỗi nút chứa thông tin về một thực thể. Các cạnh thể hiện 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. Các thuộc tính của Key-value không chỉ được liên kết 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 đi ngang từ nút này sang nút khác trong Graph Database



**Hình 5. Ví dụ về cơ sở dữ liệu đồ thị**

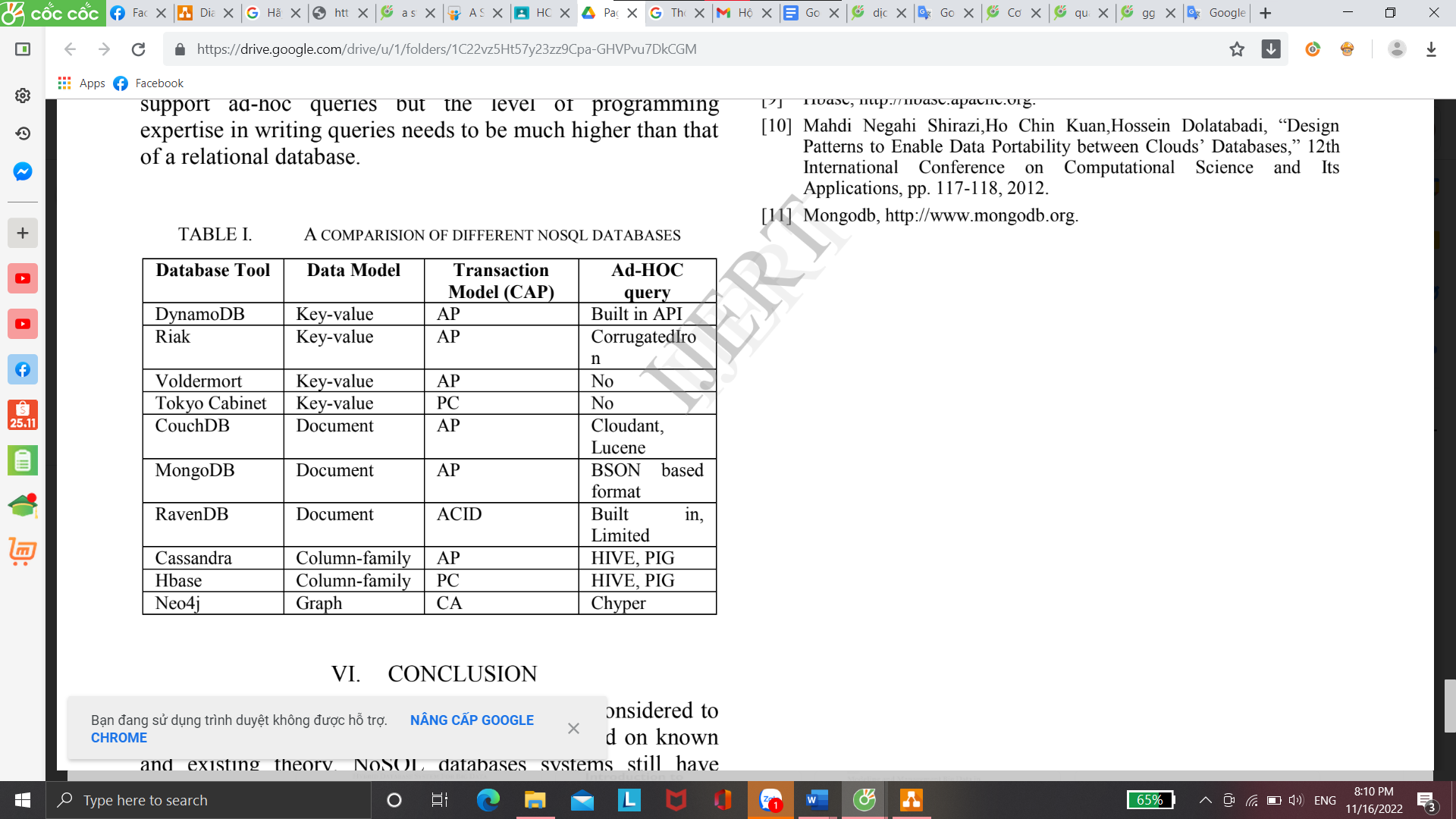
Hình 5 thể hiện một phần của cơ sở dữ liệu 'Nhân viên' được cấu trúc dưới dạng Graph Database. Mỗi nút trong Graph Database đại diện cho một thực thể nhân viên. Các thực thể này có quan hệ với nhau thông qua mối quan hệ kiểu quan hệ Knows. Thuộc tính được liên kết với mối quan hệ là Duration. Sự khác biệt chính giữa biểu đồ và cơ sở dữ liệu quan hệ là data querying. Thay vì sử dụng quy trình tốn nhiều chi phí như phép nối đệ quy như trong relational database, Graph Database sử dụng phương pháp truyền tải. Trong khi truy vấn thông qua Graph Database, 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 với các nút được kết nối với nút bắt đầu, dựa trên một số quy tắc được xác định bởi logic ứng dụng. Phương pháp truyền tải chỉ liên quan đến các nút có liên quan đến ứng dụng chứ không phải toàn bộ tập dữ liệu. Do đó, sự gia tăng lớn về số lượng nút không ảnh hưởng nhiều đến tốc độ truyền tải. Mạng xã hội, khai thác dữ liệu, quản lý mạng và tính toán các tuyến đường là một vài trong số các lĩnh vực mà Graph Database đã được sử dụng rộng rãi. Neo4j , GraphDB là những Graph Database phổ biến được sử dụng ngày nay. Khi chúng ta nói về SQL so với NoSQL, sự cạnh tranh thực sự không phải giữa các Database. So sánh là giữa các mô hình giao dịch của cả hai Database. Transaction được định nghĩa là đơn vị logic của quá trình xử lý Database được hình thành bởi một chương trình đang thực thi. Transaction của Database SQL dựa trên các thuộc tính ACID nghiêm ngặt. Trong đó ACID là chữ viết tắt của Atomicity, Consistence, Isoation và Durability. Nhưng các nhà thiết kế Database NoSQL đã đưa ra quyết định rằng thuộc tính ACID quá hạn chế để đạt được yêu 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. CAP là viết tắt của Consistency, Availability and Partition tolerance. Định lý nói rằng các nhà thiết kế có thể đạt được bất kỳ hai thuộc tính nào trong số này cùng một lúc trong môi trường phân tán. Các nhà thiết kế có thể đảm bảo Consistency, Availability and Partition tolerance, tức là Database dựa trên CA. Nếu nhà thiết kế sử dụng availability và partition tolerance với chi phí của Consistency, thì đó là Database dựa trên AP. Và nếu đảm bảo Consistency và Partition tolerance với chi phí availability thì Database dựa trên CP. Giao dịch của NoSQL có thể được phân loại như sau:

* Lo lắng về consistency và availability (CA): Loại hệ thống Database này đảm bảo mức độ ưu tiên của nó nhiều hơn đối với availability và consistency của dữ liệu bằng cách sử dụng phương pháp sao chép. Một phần của Database không bận tâm về partition tolerance . Trường hợp xảy ra phân vùng giữa các nút, dữ liệu sẽ không đồng bộ. The relational database,, hệ thống Database Vertica và Greenplum thuộc loại Database như vậy.
* Lo lắng về consistency và partition tolerance (CP): Ưu tiên của hệ thống database như vậy là đảm bảo tính nhất quán của dữ liệu. Nhưng nó không hỗ trợ cho availability tốt. Dữ liệu được lưu trữ trong các nút phân tán. Khi một nút ngừng hoạt động, dữ liệu sẽ không khả dụng để duy trì tính nhất quán giữa các nút. Nó duy trì partition tolerance bằng cách ngăn đồng bộ hóa lại dữ liệu. Hypertable, BigTable, HBase là một vài hệ thống database quan tâm đến CP.
* Lo lắng về availability và partition tolerance (AP): Ưu tiên của hệ thống database như vậy là chủ yếu đảm bảo availability của dữ liệu và partition tolerance. Ngay cả khi có lỗi giao tiếp giữa các nút, các nút vẫn trực tuyến. Sau khi partition 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 vài database tuân theo nguyên tắc này. Sau đó, định lý CAP được mở rộng thành PACELC. PACELC là viết tắt của partition, availability, consistency, else, latency, consistency. Theo mô hình này, sự đánh đổi giữa availability và consistency không chỉ dựa trên partition tolerance mà còn phụ thuộc vào sự tồn tại của network partition. Nó cho thấy độ trễ là một trong những yếu tố quan trọng, vì hầu hết các hệ thống database phân tán đều sử dụng công nghệ sao chép để đảm bảo tính khả dụng. eBay sau đó đã giới thiệu một định lý mới được gọi là định lý BASE. BASE nhằm mục đích đạt được availability thay vì consistency của database. BASE là chữ viết tắt của basically available, soft state và eventually consistent.
* Basically Available: Basically available nói rằng ngay cả khi một phần của database không khả dụng, các phần khác của database vẫn tiếp tục hoạt động như mong đợi. Trong trường hợp nút bị lỗi, hoạt động sẽ tiếp tục trên bản sao của dữ liệu được lưu trữ trong một số nút khác.
* Soft State: Soft state nói rằng 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: Eventual consistency cho biết sau bất kỳ lần cập nhật dữ liệu nào, dữ liệu có thể không nhất 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.

**COMPARISION OF NOSQL DATABASES**

Mongodb, http://www.mongodb.org.Không có bất kỳ quy tắc khó và nhanh nào để quyết định database 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 database. Sau đây là một vài sự thật có thể giúp ích trong việc lựa chọn cơ sở dữ liệu cho doanh nghiệp.

* Nếu các ứng dụng chỉ lưu trữ và truy xuất các mục dữ liệu không rõ ràng đối với hệ thống quản lý database và các đốm màu bằng cách sử dụng khóa làm mã định danh, thì kho lưu trữ key-value là lựa chọn tốt nhất. Nhưng nếu ứng dụng thích truy vấn database với một số giá trị thuộc tính khác với khóa, nó sẽ không thành công. Ngoài ra, trong khi cập nhật hoặc đọc một trường riêng lẻ trong kho lưu trữ key-value bản ghi là một lỗi.
* Khi các ứng dụng có nhiều tính chọn lọc hơn và cần lọc các bản ghi dựa trên các lĩnh vực non-key, hoặc truy xuất hoặc cập nhật các lĩnh vực riêng lẻ trong một bản ghi dưới dạng nó, thì tài liệu database là một giải pháp hiệu quả. Kho lưu trữ dữ liệu tài liệu cung cấp khả năng truy vấn tốt hơn so với kho lưu trữ dữ liệu key-value.
* Khi các ứng dụng cần lưu trữ các bản ghi có hàng trăm hoặc hàng nghìn lĩnh vực, nhưng truy xuất một tập hợp con của các lĩnh vực đó trong hầu hết các truy vấn mà nó thực hiện, thì trong trường hợp đó, lưu trữ dữ liệu column-family là một lựa chọn hiệu quả. Các kho lưu trữ dữ liệu như vậy phù hợp với các tập dữ liệu lớn có quy mô lớn.
* Nếu các ứng dụng cần lưu trữ và xử lý thông tin trên dữ liệu được liên kết chặt chẽ với mối quan hệ rất phức tạp giữa các thực thể, thì graph database là lựa chọn tốt nhất. Trong graph database, các thực thể và mối quan hệ giữa các thực thể được coi trọng như nhau.

**Bảng 1. SO SÁNH CÁC CƠ SỞ DỮ LIỆU NOSQL KHÁC NHAU**

Bảng 1 trình bày danh sách các database, các mô hình dữ liệu tương ứng của chúng, cùng với mô hình giao dịch và ngôn ngữ truy vấn được sử dụng bởi các database này. Cassandra cho facebook, HBase cho Google, DynamoDB cho Amazon là một vài database 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 database như Neo4j, Riak và MongoDB được phát triển để phục vụ các tổ chức khác. Về mô hình giao dịch, hầu hết các database như DynamoDB, Riak, Cassandra và Voldermort ưu tiên tính khả dụng hơn consistency. Trong khi Nội các Tokyo, Hbase thích consistency hơn availability. database 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 ad-hoc. Mặc dù database NoSQL được đề cập trong Table hỗ trợ truy vấn ad-hoc nhưng trình độ chuyên môn lập trình viết truy vấn cần cao hơn nhiều so với relational database

**CONCLUSION**

Trong lĩnh vực database, database NoSQL được coi là khá mới. Tuy nhiên, những điều này đang được phát triển dựa trên lý thuyết đã biết và hiện có. Các hệ thống database NoSQL vẫn còn nhiều hạn chế. Không có tiêu chuẩn chung cũng như không có bất kỳ ngôn ngữ truy vấn phổ biến và quen thuộc nào để truy vấn database NoSQL. Mỗi database hoạt động theo một cách khác nhau và làm những việc khác nhau. Tương đối những cơ sở dữ liệu này là chưa trưởng thành và không ngừng phát triển. Database 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 lưu trữ dữ liệu. Bài báo này mô tả giới hạn của database quan hệ cùng với các loại mô hình dữ liệu NoSQL khác nhau. Vì không có sẵn đánh giá để tìm ra công cụ phù hợp, bài viết này so sánh sức mạnh và hạn chế của từng mô hình dữ liệu. Hạn chế của database 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.

**REFERENCES**

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., So sánh cơ sở dữ liệu đám mây: SimpleDB của Amazon và Bigtable của Google, Hội nghị quốc tế về các xu hướng gần đây trong hệ thống thông tin, IEEE, trang 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 BasicApplication 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>.