future internet

# Bài báo
# Nghiên cứu về hoạt động tham gia trong MongoDB Preserving
# Các mô hình dữ liệu bộ sưu tập cho tương lai Ứng dụng Internet

**Antonio Celesti 1,2, * , †, Maria Fazio 1,3 và Massimo Villari 1,3**

1 Khoa MIFT, Đại học Messina, 98166 Messina, Ý; mfazio@unime.it (MF); mvillari@unime.it(MV)
2 Phòng thí nghiệm DỮ LIỆU LỚN — CINI, Via Volturno, 58, 00185
3 Rome, Ý IRCCS Centro Neurolesi “Bonino Pulejo”, Contrada Casazza, SS113, 98124 Messina, Ý
* Thư từ: acelesti@unime.it; Tel.: +39-0906-768-577
† Địa chỉ hiện tại: Viale F. Stagno d'Alcontres, 31 98166 Messina, Italy.

Nhận: ngày 14 tháng 1 năm 2019; Được chấp nhận: ngày 22 tháng 3 năm 2019; Xuất bản: 27 tháng 3,2019

**Tóm tắt**: Hiện tại, chúng tôi đang quan sát thấy sự bùng nổ dữ liệu cần đư ợc lưu trữ và xử lý qua Internet, và đư ợc đặc trưng bởi khối lượng lớn, tốc độ và sự đa dạng. Vì lý do này, các nhà phát triển phần mềm đã bắt đầu xem xét các giải pháp NoSQL để lưu trữ dữ liệu. Tuy nhiên, các hoạt động tầm thường trong Hệ thống quản lý cơ sở dữ liệu quan hệ (DBMS) truyền thống có thể trở nên rất phức tạp trong các DBMS NoSQL. Đây là trường hợp của hoạt động nối để thiết lập kết nối giữa hai hoặc nhiều cấu trúc DB, mà cấu trúc của chúng không có sẵn rõ ràng trong nhiều cơ sở dữ liệu NoSQL. Do đó, mô hình dữ liệu phải được thay đổi hoặc một tập hợp các hoạt động phải được thực hiện để giải quyết các truy vấn cụ thể trên dữ liệu. Do đó, câu hỏi mở là: các giải pháp NoSQL hoạt động như thế nào khi chúng phải thực hiện các hoạt động nối trên dữ liệu không được hỗ trợ nguyên bản?

Chất lượng của các giải pháp NoSQL trong những trường hợp như vậy là gì? Trong bài báo này, chúng tôi giải quyết các vấn đề như vậy một cách cụ thể khi xem xét một trong những DB định hư ớng tài liệu NoSQL chính hiện có trên thị trư ờng: MongoDB. Đặc biệt, chúng ta thảo luận về một cách tiếp cận để thực hiện các hoạt động nối ở lớp ứng dụng trong MongoDBcho phép chúng ta bảo toàn các mô hình dữ liệu. Chúng tôi phân tích hiệu suất của phương pháp đề xuất thảo luận về chi phí đư ợc giới thiệu so với các DB giống SQL.

**Từ khóa**: internet tương lai; dữ liệu lớn; NoSQL; MongoDB; tham gia

## 1. Giới thiệu

Hiện tại, chúng tôi đang quan sát thấy sự bùng nổ dữ liệu không đồng nhất đến từ phương tiện truyền thông xã hội, đường truyền giải trí, dịch vụ Điện toán đám mây và Internet vạn vật (IoT). Dữ liệu đó cần đư ợc lưu trữ và xử lý cho các dịch vụ quảng cáo của các ứng dụng Internet trong tương lai. Số lượng khổng lồ các tập dữ liệu phức tạp và phi cấu trúc này, thường được biểu thị bằng thuật ngữ Dữ liệu lớn, đư ợc đặc trưng bởi khối lượng lớn, tốc độ và sự đa dạng, và không thể đư ợc quản lý bằng cơ sở dữ liệu giống như Ngôn ngữ truy vấn có cấu trúc (SQL) truyền thống (DB), do tổ chức cấu trúc tĩnh [1]. Thiết kế các hệ thống DB xử lý hiệu quả các yêu cầu của các ứng dụng Dữ liệu lớn là một nhiệm vụ đầy thách thức. Một mặt, chúng ta phải quản lý một lượng lớn dữ liệu, mặt khác cần phải thực hiện các nghiên cứu phân tích về chúng. Sự phát triển không ngừng của Dữ liệu lớn khiến tính khả dụng của dữ liệu cao trở thành một yếu tố quan trọng. Nội dung dữ liệu thư ờng đư ợc chia sẻ thông qua các cổng web và ngư ời dùng, bao gồm cả nhà cung cấp và đối tác, có thể ở các khu vực địa lý khác nhau. Vì những lý do này, việc áp dụng phư ơng pháp phân phối dữ liệu theo địa lý, đặt dữ liệu gần với ngư ời dùng, đang trở thành một giải pháp chiến lư ợc hơn bao giờ hết. Xem xét tất cả các động cơ trư ớc đó, có thể nói rằng các ứng dụng xử lý dữ liệu giống

SQL truyền thống không đủ khả năng quản lý và thu thập loại thông tin dữ liệu này. Vì mục đích này, các tổ chức đang xem xét

NoSQL (viết tắt của "Not Only SQL") các giải pháp đư ợc coi là hiệu quả hơn Hệ thống quản lý DataBase giống Sql (DBMS) truyền thống trong quản lý dữ liệu.

Hệ quản trị cơ sở dữ liệu NoSQL đã đư ợc áp dụng thành công để lư u trữ Dữ liệu lớn trong nhiều miền ứng dụng, như ng hiện tại, điều quan trọng đối với các công ty kinh doanh là phải biết cách phân tích và xử lý dữ liệu để đư a ra các quyết định kinh doanh chiến lư ợc và quan trọng. Tính sẵn sàng cao, khả năng mở rộng và độchính xác trong phân tích Dữ liệu lớn có thể dẫn đến việc ra quyết định tự tin hơn và các quyết định tốt hơn đồng nghĩa với hiệu quả hoạt động cao hơn cùng với giảm chi phí và rủi ro. Thật vậy, điều quan trọng không phải là số lượng dữ liệu đối với các công ty, mà là họ có thể làm gì với dữ liệu. Tuy nhiên, vì Dữ liệu lớn có thể vừa có cấu trúc vừa không có cấu trúc, nên các hoạt động nhỏ trong Hệ thống quản lý cơ sở dữ liệu quan hệ (DBMS) truyền thống có thể trở nên rất phức tạp trong các DBMS NoSQL. Đây là trường hợp của thao tác nối để thiết lập kết nối giữa hai hoặc nhiều cấu trúc cơ sở dữ liệu mà cấu trúc của chúng không có sẵn rõ ràng trong nhiều cơ sở dữ liệu NoSQL. Do đó, các câu hỏi mở là: các giải pháp NoSQL hoạt động như thế nào khi chúng phải thực hiện các thao tác nối trên dữ liệu không đư ợc hỗ trợ nguyên bản? Chất lư ợng của các giải pháp NoSQLtrong những trư ờng hợp như vậy là gì?

Trong bài báo này, chúng tôi phân tích hành vi của một trong những DB hư ớng tài liệu NoSQL chính, MongoDB [2], xử lý cụ thể vấn đề liên kết bên trong. MongoDB lư u trữ các tài liệu BSON (nghĩa là các bản ghi dữ liệu) trong các bộ sư u tập phi cấu trúc và các bộ sư u tập trong DB. Cộng đồng mã nguồn mở MongoDB ,nhận thức đư ợc rằng hoạt động tổng hợp giữa các bộ sư u tập đư ợc yêu cầu cao bởi các nhà phát triển, đã giới thiệu toán tử tra cứu $ trong phiên bản 3.2. toán tử $lookup có thể thực hiện thao tác Left Outer Equi-Join (gọi ngắn gọn là "left join") với hai hoặc nhiều tập hợp. Tuy nhiên, nó cung cấp dư ới dạng đầu ra là tất cả các tài liệu của tập hợp bên trái ngay cả khi nó không khớp với bất kỳ trư ờng tài liệu nào của tập hợp bên phải. Do đó, truy vấn có thể chứa một lư ợng lớn tài liệu vô dụng trở thành một vấn đề, đặc biệt là trong bối cảnh Dữ liệu lớn. "Nối bên trong" thư ờng đư ợc sử dụng trong các cơ sở dữ liệu kiểu SQL truyền thống giải quyết loại vấn đề này, như ng nó không đư ợc hỗ trợ trong MongoDB cũng như trong hầu hết các DB hư ớng tài liệu NoSQL. Việc tổng hợp hiệu quả giữa các dữ liệu trong các bộ sư u tập phải đư ợc thực hiện

bởi các ứng dụng bên ngoài được kết nối với MongoDB. Để giải quyết vấn đề như vậy, chúng tôi trình bày mộttriển khai lớp ứng dụng của liên kết bên trong để khắc phục vấn đề đư ợc mô tả. Trong các thử nghiệm của mình, chúng tôi phân tích hiệu suất của giải pháp liên kết bên trong đư ợc đề xuất, đồng thời thảo luận về chi phí hoạt động đư ợc giới thiệu so với các cơ sở dữ liệu giống như SQL, xem xét các bộ dữ liệu tư ơng đư ơng.

Bài báo đư ợc tổ chức như sau. Phần 2 mô tả các công việc liên quan. Tổng quan về sự khác biệt giữa các giải pháp giống SQL và NoSQL đư ợc cung cấp trong Phần 3. Đặc biệt, các lợi thế của các giải pháp NoSQLđư ợc thảo luận, cũng phân tích các giới hạn của chúng về cấu trúc và hoạt động so với các giải pháp giống SQL, đặc biệt là xem xét MongoDB và MySQL như các mô hình DB. Tổng quan về các tính năng của MongoDB đư ợcđư a ra trong Phần 4. Trong Phần 5, chúng tôi trình bày giải pháp nối bên trong lớp ứng dụng mà chúng tôi đã triển khai cho MongoDB. Một phân tích so sánh giữa MongoDB và MySQL xem xét cụ thể cả hai hoạt động nốibên trong đư ợc trình bày trong Phần 6. Phần 7 kết thúc bài báo.

## 2. Bối cảnh và công việc liên quan

Với sự ra đời của các ứng dụng Internet trong tương lai, ngày càng có nhu cầu chuyển từ SQL-Like sangNoSQL DBMS (các) DBMS để có thể quản lý hiệu quả một khối lư ợng lớn dữ liệu không đồng nhất trong các lĩnhvực khác nhau bao gồm thành phố thông minh [3], công nghiệp 4.0 [4] , tiếp thị [5], ngư ời máy [6], giao thông vận tải [7], chăm sóc sức khỏe [8,9], bộ gen [10,11], v.v. Do đó, nhiều công trình khoa học và kỹ thuật đã đư ợc đề xuất trong tài liệu cho đến nay tập trung vào phân tích so sánh giữa các giải pháp cơ sởdữ liệu SQL-like và NoSQL.

Trong [12], một phép so sánh đư ợc đề xuất giữa cả hai loại cơ sở dữ liệu (SQL và NoSQL) để kiểm tra các hoạt động CRUD khác nhau, tức là Tạo, Đọc, Cập nhật và Xóa khác nhau về lượng dữ liệu được sử dụng.Do đó, nhờ phân tích của họ, các tác giả đã đưa ra một chỉ số đánh giá để chọn cơ sở dữ liệu nào sẽ sử dụng cho một tập dữ liệu nhất định. Tất cả các công việc trên đều đưa ra đánh giá riêng về hiệu suất của từng DBMS. Chúng đã đư ợc kiểm tra tất cả các hoạt động cơ bản (CRUD) nhưng  không có hoạt động thực hiện phân tích so sánh giữa các hoạt động Tham gia (SQL) và Tham gia (NoSQL), bởi vì mọi ngư ời đều đồng ý rằng hoạt động này không tồn tại trong cơ sở dữ liệu không quan hệ. Trong [13], để chạycác thử nghiệm so sánh giữa các loại cơ sở dữ liệu khác nhau, một ứng dụng ngân hàng sử dụng các loại cơsở dữ liệu khác nhau đư ợc thiết kế và triển khai. Các kết quả thử nghiệm đo thời gian của các hoạt động cơ bản được thực hiện trên mỗi cơ sở dữ liệu trong số chúng chứng minh rằng cơ sở dữ liệu NoSQL với SQL có các tính năng Foundationdb như vậy hóa ra tốt hơn SQL. Trong [14] , các tác giả đã thực hiện một phân tích đàm thoại cẩn thận về cơ sở dữ liệu NoSQL, so sánh sự khác biệt giữa các loại cơ sở dữ liệu NoSQL khác nhau , cuối cùng nêu bật những ư u điểm và như ợc điểm của chúng. Các tác giả trong [8,15] trình bàyHệ thống Thông tin Lư u trữ Mở (OAIS), khai thác Cơ sở dữ liệu hư ớng cột NoSQL (DB) Cassandra nổi tiếng.Họ đã thực hiện một số thử nghiệm về các giải pháp đư ợc đề xuất trong một tình huống sử dụng thực tế, sosánh hiệu suất với giải pháp MySQL truyền thống. Họ nhận thấy rằng trong một cơ sở hạ tầng không phân tán, Cassandra không hoạt động tốt so với MySQL, bởi vì lợi ích của việc quản lý họ cột không thể đư ợc trư ng bày. Tuy nhiên, lợi thế trong việc sử dụng Cassandra là rõ ràng khi Cassandra đư ợc Kết hợp với Apache Spark. Trong miền ứng dụng chăm sóc sức khỏe, tham khảo. [16] trình bày một giao thức để đánh giáđộ phức tạp tính toán của việc truy vấn hồ sơ sức khỏe điện tử tiêu chuẩn hóa quan hệ và phi quan hệ (EHR). Giao thức này cho thấy rằng các hệ thống SQL không thực tế cho các truy vấn một bệnh nhân vì thờigian phản hồi chậm hơn. Cơ sở dữ liệu NoSQL hiển thị độ dốc tuyến tính và MongoDB hoạt động nhanh hơn đáng kể so

với eXist DBMS. Trong đồng thời, MongoDB cũng hoạt động tốt hơn nhiều so với MySQL quan hệ.[ 17] trình bày phân tích so sánh giữa cơ sở dữ liệu NoSQL như HBase, MongoDB, BigTable, SimpleDB và cơ sở dữ liệu quan hệ như MySQL xác định giới hạn của chúng khi áp dụng vào thế giới thực. Các tác giả đã kiểm tra cụ thể các cơ sở dữ liệu trên để phân tích cả các truy vấn đơn giản và phức tạp hơn. Giải pháp nối bộ-tư ơng tự trong NoSQL sử dụng MapReduce đư ợc thảo luận trong [18] . Đặc biệt, số lư ợng so sánh để tìm tất cả các cặp tư ơng tự đã đư ợc giảm bớt bằng cách mở rộng kỹ thuật lọc tiền tố cho MapReduce Framework. Giải pháp này dẫn đến một thứ tự cải thiện đáng kể về hiệu suất so với các giải pháp hiện có hiệu quả nhất. Trong [19], một giải pháp dựa trên chỉ mục điều chỉnh một thuật toán tham gia xếp hạng tậptrung đư ợc thảo luận. Đặc biệt, các tác giả cung cấp (i) các thuật toán MapReduce chỉ ra cách xây dựng các chỉ số và cấu trúc thống kê, (ii) các thuật toán cho phép cập nhật trực tuyến các chỉ số này và (iii)các thuật toán xử lý truy vấn sử dụng chúng. Tất cả các thuật toán đư ợc triển khai và thử nghiệm trong Hadoop (HDFS) và HBase, sử dụng các truy vấn khác nhau trên các bảng có kích thư ớc khác nhau và các phân phối thuộc tính điểm số khác nhau. Vấn đề tối ư u hóa truy vấn đư ợc giải quyết trong [20], nơi các tác giả giải quyết việc thực thi hiệu quả các truy vấn JOIN trên ngôn ngữ truy vấn Hadoop, Hive, trên các khoDữ liệu lớn hạn chế. Một phư ơng pháp tích hợp dữ liệu mới để truy vấn dữ liệu riêng lẻ từ các hệ thống cơ sở dữ liệu quan hệ và NoSQL khác nhau đư ợc đề xuất trong [21]. Giải pháp như vậy không hỗ trợ kết hợpvà tổng hợp giữa các nguồn dữ liệu, mà nó chỉ thu thập dữ liệu đến từ các DBMS đư ợc tách biệt khác nhau theo các tùy chọn lọc và di chuyển chúng. Phư ơng pháp đư ợc đề xuất dựa trên phư ơng pháp tiếp cận siêu mô hình và nó bao gồm sự không đồng nhất về cấu trúc, ngữ nghĩa và cú pháp của các hệ thống nguồn. Trong [22], các tác giả trình bày một phần khung có thể tự động ánh xạ cơ sở dữ liệu quan hệ MySQL sang cơ sở dữ liệu MongoDB NoSQL. Đặc biệt, một thuật toán sử dụng siêu dữ liệu đư ợc lư u trữ trong các bảng hệ thống MySQL đư ợc sử dụng. Hoạt động tham gia đư ợc xử lý bằng các tài liệu nhúng. Trong [23], các tác giảchỉ ra cách có thể đạt đư ợc quyền truy cập SPARQL động vào dữ liệu không quan hệ khi xem xét MongoDB. Đặc biệt, họ sử dụng SparqlMap-M, là phần mở rộng của công cụ SPARQL-to-SQL thực hiện chuyển đổi một phầncác truy vấn SPARQL bằng cách sử dụng trừu tư ợng hóa quan hệ trên kho tài liệu.

Hơn nữa, dữ liệu trùng lặp trong kho tài liệu đư ợc sử dụng để giảm số lư ợng liên kết và tối ư u hóa tùychỉnh đư ợc giới thiệu. Mặc dù sự kiện, một giải pháp như vậy cho phép thực hiện các hoạt động nối, nó cần tạo ra một phần trừu tư ợng mới của toàn bộ mô hình dữ liệu mà không xét đến lợi thế của quyền truy cập trực tiếp vào MongoDB. Tham khảo [24] giới thiệu một khuôn khổ nhằm phân tích các ứng dụng dữ liệu bán cấu trúc bằng cách sử dụng cơ sở dữ liệu NoSQL MongoDB. Khung đề xuất tập trung vào các khía cạnh chính cần thiết cho phân tích dữ liệu bán cấu trúc về thu thập dữ liệu, phân tích cú pháp dữ liệu và dự đoán  dữ liệu. Một phân tích hiệu suất cho các hoạt động chọn + tìm nạp cần thiết cho phân tích, của MySQL và MongoDBđư ợc thực hiện khi cơ sở dữ liệu NoSQL MongoDB hoạt động tốt hơn cơ sở dữ liệu MySQL. Mặc dù MongoDB đư ợc chấp nhận rộng rãi và sự hiện diện của nhiều công trình khoa học và kỹ thuật tập trung vào các khía cạnh khác nhau của nó, theo hiểu biết của chúng tôi, tại thời điểm viết bài, cho đến nay chư a có công trình nào đối mặt trực tiếp với vấn đề tham gia nội bộ.

## 3. SQL-Like so với NoSQL DBMS

Hiện nay, hầu hết các giải pháp DBMS chủ yếu dựa trên mô hình quan hệ và cho phép soạn thảo các truy vấn phức tạp trên dữ liệu bằng Ngôn ngữ truy vấn có cấu trúc (SQL). Vì lý do này, chúng còn được gọi là cơ sở dữ liệu giống như SQL. Giải pháp Hệ quản trị cơ sở dữ liệu quan hệ (RDBMS) đầu tiên xuất hiện vào những năm 1970 và trong bốn thập kỷ, chúng đã thống trị thị trường. Tuy nhiên,

nhu cầu ngày càng tăng gần đây về cơ sở dữ liệu lớn hơn, do sự thành công của Điện toán đám mây, phương tiện truyền thông xã hội và IoT đã vượt qua các giới hạn của RDBMS. Trong bối cảnh này, NoSQL (viết tắt của Not Only SQL) đã nhanh chóng thu hút sự chú ý của cả giới học thuật và ngành công nghiệp như một giải pháp đầy hứa hẹn để giải quyết "vấn đề Dữ liệu lớn". Trên thực tế, vì các giải pháp NoSQL có thể quản lý các mô hình dữ liệu không có lược đồ hoặc không có lược đồ nên chúng có thể dễ dàng lưu trữ một lượng lớn dữ liệu không đồng nhất và không có cấu trúc đến từ một số nguồn dữ liệu khác nhau. Điều này là không thể trong các giải pháp RDBMS, trong đó các mô hình dữ liệu tĩnh phải được xác định trước.Mô hình quan hệ dựa trên đại số quan hệ và phép tính quan hệ, đồng thời đại diện cho một công cụ mạnh mẽ cho phép các nhà phát triển viết các truy vấn phức tạp trên dữ liệu được biểu thị bằng các phương tiện quan hệ. Lược đồ xác định tên của quan hệ, xác định các cột đại diện cho các thuộc tính hoặc trường. Hàng, còn được gọi là bộ hoặc bản ghi, đại diện cho thể hiện của quan hệ. Ràng buộc toàn vẹn là thuộc tính mà các quan hệ phải được thỏa mãn để biểu diễn thông tin chính xác trong các thể hiện dữ liệu.Các ràng buộc toàn vẹn có thể được phân loại theo các thành phần cơ sở dữ liệu có liên quan:

• ràng buộc miền, hạn chế miền của thuộc tính;

•các ràng buộc trong quan hệ, cần có sự nhất quán trong khóa chính và khóa ngoại.

Trong những năm gần đây, các giải pháp cơ sở dữ liệu NoSQL đã xuất hiện để giải quyết các giới hạn của RDBMS về mặt quản lý dữ liệu lớn, phi cấu trúc, phức tạp và động. Khác với các DB giống SQL, hầu hết các giải pháp NoSQL không thực thi các ràng buộc liên quan và miền. Chúng tôi có thể xác định bốn họ NoSQL DB chính, mỗi họ dựa trên một mô hình dữ liệu khác nhau:

1. key-Value-Store DB: dựa trên mô hình khóa-giá trị, trong đó mỗi bản ghi được đặc trưng bởi một khóa chính và một tập hợp các giá trị; nó còn được gọi là lưu trữ hàng, vì dữ liệu đại diện cho một bản ghi được lưu trữ cùng nhau. Ví dụ về NoSQL DB sử dụng mô hình dữ liệu này là Redis và Riak.
2. DB hướng theo cột: dữ liệu được lưu trữ trong các cột (nó bổ sung cho các DB lưu trữ dữ liệu thay thế dọc theo các hàng, cụ thể là các DB hướng theo hàng); mỗi thuộc tính của bảng được lưu trữ trong một tệp hoặc vùng riêng biệt trên hệ thống lưu trữ. Ví dụ về loại DB này là Cassandra.
3. DB hướng tài liệu: dữ liệu được lưu trữ trong một tài liệu; các tài liệu có thể được lồng vào nhau và do đó chứa các tài liệu, danh sách và mảng khác; danh mục này cũng bao gồm Cơ sở dữ liệu XML gốc, được gọi như vậy vì chúng xác định một mô hình logic cho tài liệu XML. Ví dụ là MongoDB và CoucheDB.
4. Graph DB: được thiết kế cho dữ liệu có các quan hệ được biểu diễn dưới dạng một biểu đồ bao gồm các phần tử được kết nối với nhau với một số lượng hữu hạn các quan hệ giữa chúng. Ví dụ bao gồm Neo4j và OrientDB.

Tại thời điểm này, câu hỏi đặt ra là: làm thế nào để chúng ta hiểu được liệu nhu cầu của người dùng cuối (ví dụ: một công ty, một nhà phát triển, v.v.) có được giải pháp NoSQL hoặc RDBMS đáp ứng tốt hơn hay không? Để trả lời câu hỏi này, chúng ta bắt đầu từ các định lý ACID và CAP. ACID đề cập đến bốn thuộc tính chính của một giao dịch:

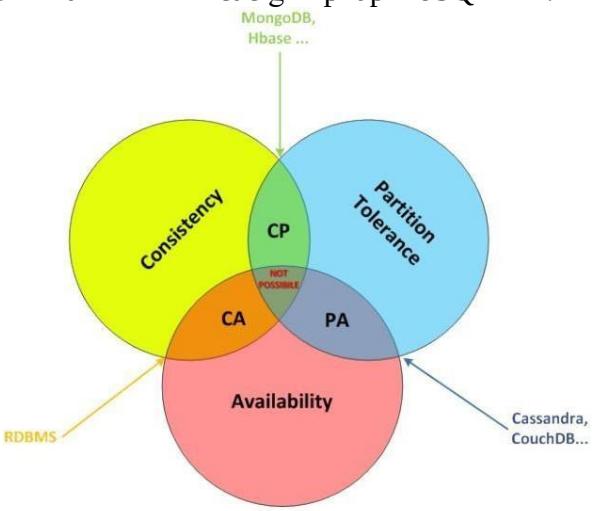
- **A as Atomicity**: các giao dịch phải là nguyên tử, không thể phân chia được; trong DBMS, hoạt động củaquá trình chuyển đổi ảnh hưởng đến DB một cách mạch lạc.

- **C as Consistency**: một giao dịch không đư ợc vi phạm các ràng buộc toàn vẹn trong DB; nếu phát sinh sự khôngnhất quán về dữ liệu, tất cả dữ liệu sẽ trở lại trạng tháitrư ớc khi giao dịch.
- **I as Isolation**: việc thực hiện một giao dịch phải độc lập với nhau.
- **D as Durability**: giaodịch, sau khi thực hiện cam kết thực hiện, sẽ có hiệu lực chính xác vàoDB và tất cả các thay đổi phải đư ợc áp dụng và không bị mất nữa.

Các giao dịch **ACID** là cơ sở của giải pháp RDBMS. Tuy nhiên, ngày nay, sự phát triển liên tục của Web và các dịch vụ phần mềm dẫn đến sự gia tăng lư ợng dữ liệu phải đư ợc quản lý bởi các ứng dụng phân tán. Điều nàyhạn chế việc áp dụng RDBMS không phân phối (nguyên bản). Để giải quyết một giới hạn như vậy, Eric Brewer đã trình bày định lý CAP [25]. Nó khẳng định rằng không thể có một hệ thống lư u trữ phân tán cung cấp đồng thờinhiều hơn hai trong số ba thuộc tính sau:

- **C as Consistency**: những thay đổi trong hệ thống lư u trữ phân tán phải đư ợc phản ánh trên tất cả các nút lưu trữ phân tán.
- **A as Availability**: hệ thống lưu trữ luôn sẵn sàng đáp ứng yêu cầu.
- **P as Partition-tolerance**: hệ thống lư u trữ có thể hoạt động và cũng có thể phản hồi các truy vấn nếu xảy ra sự cố trong mạng (ví dụ: sự cốnút lư u trữ hoặc một số thông báo bị mất).

Để hiểu rõ Định lý CAP, chúng ta hãy xem xét một nút lư u trữ. Một mình, nó chỉ có thể đảm bảo tính nhất quán và dung sai phân vùng, như ng không có sẵn; nếu chúng ta xemxét nhiều nút bảo toàn dữ liệu giống nhau, chúng đảm bảo tính khả dụng, như ng không thểđảm bảo tính nhất quán vì chúng có thể ở các trạng thái khác nhau tại một thời điểm cụ thể trước khi dữ liệu đư ợc cung cấp cho tất cả các nút. Tăng số lư ợng các nút, mức độ phức tạp của hệ thống tăng lên, và cả khả năng chịu đựng đối với các phân vùng. Khái niệm này đư ợc mô tả trong Hình 1. Định lý CAP là cơ sở của các giải pháp NoSQL DB.




Hình 1. Định lý CAP.

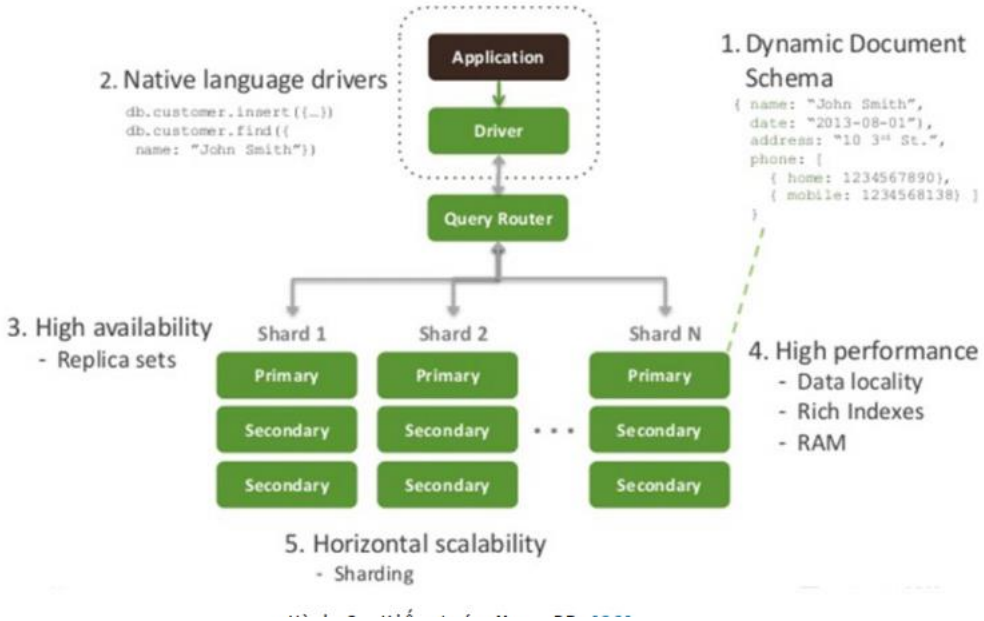
Bất cứ khi nào chúng ta áp dụng NoSQL DBs, cần phải tìm ra sự thỏa hiệp về các thuộc tính CAP nào mà chúngta muốn giải quyết, tùy thuộc vào yêu cầu của hệ thống cụthể. Ví dụ: nếu một hệ thống phải đảm bảo Tính dung nạp phân vùng cùng với Tính khả dụng đi đến tính nhất quán, các giải pháp NoSQL có thể bao gồm Riak, CoucheDB hoặcCassandra, trong khi nếu mục tiêu của hệ thống là đảm bảo Tính nhất quán và Tính khả dụng phân vùng gây bấtlợi về Tính khả dụng các giải pháp NoSQL khả thi là MongoDB hoặc Redis.

Một hệ thống có thể mở rộng khi nó tăng hiệu suất tương ứng với các tài nguyên mà nóquản lý.Việc gia tăng các nguồn lực có thể xảy ra theo hai cách: theo chiều dọc hoặc theo chiều ngang. Chia tỷ lệ dọc có nghĩa là tăng số lư ợng tài nguyên vật lý (ví dụ: CPU, RAM) trong một nút lưu trữ. Nằm ngang chia tỷ lệ có nghĩa là các nút mới đư ợc thêm vào hệ thống lư u trữ. Với chia tỷ lệ theochiều ngang, việc chia tỷ lệ linh hoạt theo yêu cầu của hệ thống lư u trữ sẽ dễ dàng hơn, trong khi tỷ lệ theo chiều dọc bị giới hạn ở khả năng của một nút. Để triển khai chia tỷ lệ theo chiều ngang của DB quan hệ, các bảng phải đư ợc phân phối trên các nút khác nhau, như ng chúng có thể đư ợc truy cập từ ngư ời dùng cuối thông qua các truy vấn sử dụng toán tử nối. Tuy nhiên, vấn đề chính của RDBMS phân tán quản lý dữ liệu lớn là sự phức tạp trong việc đảm bảo các ràng buộc toàn vẹn tham chiếu trên cơ sở của các hệ thống quan hệ. Việc triển khai mở rộng quy mô theo chiều ngang củaNoSQL DB có nghĩa là thêm nhiều phiên bản nút hơn trong hệ thống và DBMS sẽ tự động truyền dữ liệu qua các nút khi cần thiết. Giải pháp này hư ớng đến dữ liệu lớn vì không phải giữ tính toàn vẹn giữa các dữ liệu. Do đó, các DB NoSQL dễ dàng mở rộng quy mô, như ng chúng không cung cấp các hoạt động nối để thu thập dữ liệu thuộc các cấu trúc DB khác nhau.

Trong bài báo này, chúng tôi giải quyết những hạn chế như vậy của NoSQL DB, và đặc biệt, chúng tôi cung cấp một giải pháp hữu ích đã đư ợc thử nghiệm trên MongoDB, một DB định hư ớng tài liệu NoSQL nổi tiếng.MongoDB phiên bản 3.2 đã giới thiệu khái niệm kết nối textitleft giữa các tập hợp bằng toán tử tra cứu.
Hiện tại, nó không cho phép ngư ời dùng thực hiện các thao tác kết hợp theta hoặc tự nhiên.Để hoàn thành khoảng cách như vậy, chúng tôi trình bày hoạt động tham gia MongoDB textitinner đư ợc thựchiện ở lớp ứng dụng.

## 4. Tổng quan về MongoDB: Các tính năng và giới hạn chính

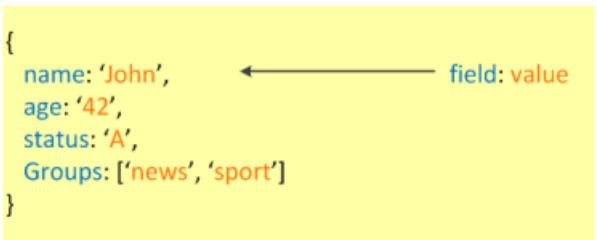
Cái tên MongoDB xuất phát từ “humongous” để xác định thứ gì đó “to lớn”. Nó cung cấp hiệu suất cao trong các hoạt động viết và đọc. Trên thực tế, các thao tác viết (theo mặc định) là cháy và quên.Điều đó có nghĩa là trình điều khiển sẽ không xác nhận sự thành công của hoạt động viết. Tuy nhiên, nếu cần, có thể đặt chế độ "ghi an toàn" để ghi buộc hệ thống phải gửi phản hồi. Ngoài ra, MongoDB cho phép kích hoạt tính năng “ghi nhật ký”, tạo một bản ghi nhật ký cho mỗi khách hàng bắt đầu thao tác ghi. Loại đệm trong bộ nhớ này cho phép ngăn chặn bất kỳ lỗi nào do mất dữ liệu.Bất cứ khi nào các tệp tạp chí được đồng bộ hóa, thì dữ liệu sẽ được ghi vào DB. Một tính năng quan trọng khác là khả năng mở rộng dễ dàng của hệ thống lưu trữ: MongoDB cung cấp các công cụ hữu ích để phân phối dữ liệu hiệu quả trên nhiều máy, chẳng hạn như sharding và sao chép. Trong hình 2 , kiến trúc tham chiếu của MongoDB được hiển thị. Nó mô tả cách các ứng dụng tương tác với DB thông qua một Trình điều khiển cụ thể cần thiết để chèn dữ liệu và thực hiện các truy vấn.




Hình 2. Kiến trúc MongoDB [26].

4.1. Hoạt động CRUD

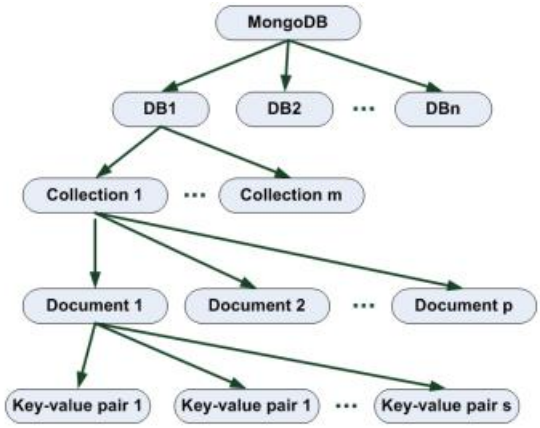
Phù hợp với DBMS truyền thống, MongoDB cho phép thực hiện các hoạt động CRUD (Tạo, Đọc, Cập nhật và Xóa) trên dữ liệu. Nó lưu trữ dữ liệu bên trong các tài liệu không có lược đồ. Điều này có nghĩa là có thể tạo các tài liệu lồng nhau (ví dụ, các tài liệu được bao gồm trong các tài liệu khác hoặc trong các mảng), do đó giảm nhu cầu thực hiện các thao tác nối. Hình 3 cho thấy một ví dụ về tài liệu MongoDB.Tài liệu MongoDB có cú pháp cấu trúc JSON và được lưu trữ dưới dạng tài liệu BSON (Binary JSON).Lược đồ tổ chức dữ liệu miễn phí được giới thiệu với JSON mang lại lợi thế về cấu trúc dữ liệu rất năng động và linh hoạt, đồng thời giúp phát triển mô hình dữ liệu dễ dàng hơn so với các mẫu DBMS quan hệ cứng nhắc. Tính linh hoạt này cũng được hỗ trợ bởi một ngôn ngữ truy vấn phong phú.Thật vậy, MongoDB cung cấp nhiều tính năng như: chỉ mục phụ, cập nhật, nâng cấp ("cập nhật" nếu tài liệu tồn tại, nếu không thì "chèn") và các tổng hợp đơn giản.



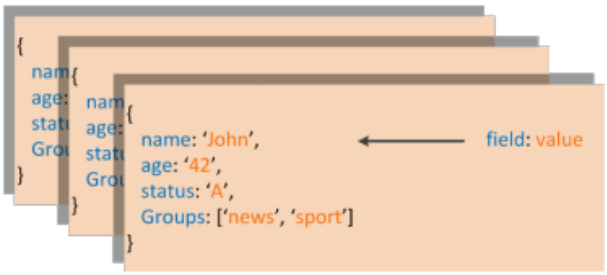

Hình 3. Ví dụ về tài liệu MongoDB.

Tổ chức hợp lý của dữ liệu và tài liệu trong MongoDB được hiển thị trong Hình 4. Bộ sưu tập là một nhóm tài liệu, tương đương với một bảng trong DB giống như SQL. Mỗi bộ sưu tập chỉ thuộc về một DB. Các bộ sưu tập không thực thi một lược đồ dữ liệu; trên thực tế, các tài liệu trong một bộ sưu tập có thể có các trường khác nhau. Tuy nhiên, thông thường, các tài liệu trong một bộ sưu tập có mục đích liên quan và

chia sẻ một số trường. Hình 5 cho thấy một ví dụ về bộ sưu tập MongoDB. MongoDBDB được hình thành bởi một nhóm các bộ sưu tập.



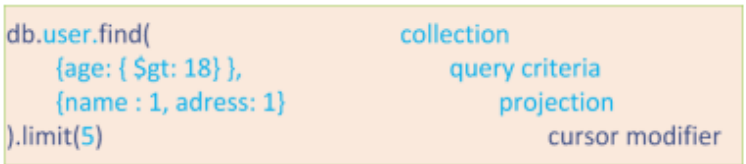

Hình 4. MongoDB logica tổ chức dữ liệu.




Hình 5. Ví dụ về bộ sưu tập MongoDB.

4.1.1. Thao tác đọc trong MongoDB

Thao tác đọc trả về các tài liệu đư ợc lư u trữ trong MongoDB. Truy vấn chỉ định tiêu chí hoặc điềukiện để xác định một hoặc nhiều tài liệu trong bộ sư u tập, đồng thời chỉ định trư ờng mục tiêu chiếu.Đối với loại hoạt động này, MongoDB cung cấp phương thức db.collection.find() chấp nhận các tiêu chí và phép chiếu làm tham số để trả về một con trỏ tới tập kết quả của các tài liệu đã truy xuất. Từ phiên bản 3.2, MongoDBhỗ trợ thao tác nối trái giữa các tập hợp khác nhau bằng toán tử tra cứu, như ng nó không hỗ trợ thao tác nối bêntrong giữa các tập hợp.Để so sánh độ phức tạp của ngôn ngữ truy vấn trong các DB khác nhau, Hình 6a,b hiển thị một ví dụ về thao tác đọc tư ơng ứng trong MongoDB và trong một DBMS giống như SQL.




(a) Thao tác đọc trong MongoDB.


(b) Thao tác đọc trong DBMS giống như SQL.

Hình 6. So sánh tư ơng ứng giữa các thao tác đọc trong MongoDB và DBMS giống SQL.

4.1.2. Thao tác viết trong MongoDB

Trong MongoDB, các thao tác viết là nguyên tử trên một tài liệu của một tập hợp cụ thể. Có ba lớp hoạt động ghi trong MongoDB: insert, liên quan đến thu thập, cập nhật, thay đổi dữ liệu hiện có của tậphợp và loại bỏ, sẽ xóa dữ liệu khỏi tập hợp.Về hoạt động chèn, MongoDB cung cấp phương thức db.collection.insert (). Đối với các hoạt động cập nhật,MongoDB cung cấp các phương thức db.collection.update () và db.collection.save (). Điều đầu tiên chấp nhận tiêu chí truy vấn để xác định tài liệu cần cập nhật. Một phương thức như vậy cũng hỗ trợ "upert" rằng nó có thể hoạt động bằng cách đặt tham số thứ ba thành true, tham số này theo mặc định là false. Nếu "upert" tìm thấy tài liệu đã chọn, nó sẽ thực hiện cập nhật tài liệu đó, nếu không nó sẽ tạo một tài liệu mới. Thay vào đó, phư ơng thức db.collection.save (document) thể hiện sự kết hợp giữa các thao tác chèn và cập nhật. Cuối cùng, thao tác xóa sẽ xóa tài liệu khỏi bộ sưu tập bằng phương thức db.collection.remove (). Nó có thể chấpnhận các tiêu chí truy vấn để xác định tài liệu nào phải đư ợc xóa.

4.1.3. Giới hạn của mối quan hệ dữ liệu

Bằng cách thao tác cấu trúc của tài liệu, có thể xác định mối quan hệ giữa các dữ liệu trong một bộ sưu tập. Do đó, các mối quan hệ giữa các dữ liệu đã được thực hiện trong một bộ sưu tập bằng các kỹ thuật tham chiếu và tài liệu nhúng:

• Tham chiếu cho phép quản lý các mối quan hệ dữ liệu bao gồm các tham chiếu hoặc liên kết từ tài liệu này sang tài liệu khác. Các ứng dụng khác nhau có thể sử dụng các tham chiếu này để truy cập dữ liệu liên quan. Về nguyên tắc, các mẫu dữ liệu này được chuẩn hóa. MongoDB sử dụng hai phương pháp để thực hiện kiểu quan hệ như vậy giữa các phần dữ liệu khác nhau:

1. Tham chiếu thủ công, cho phép lưu trư ờng của một tài liệu vào một tài liệu khác làm tài liệu tham khảo,vì vậy ứng dụng có thể chạy một truy vấn thứ hai để trả về dữ liệu nhúng.
2. DBRef, bao gồm các trường sau:(a) $ ref chứa tên của bộ sưu tập nơi tài liệu được tham chiếu cư trú. (b)$ id chứa giá trị của trư ờng _id của tài liệu được tham chiếu. (c) $ db (tùy chọn) chứa tên của cơ sở dữ liệu nơi tài liệu cư trú.

Ví dụ:

{ "_id" : ObjectId("5126bb f 64aed4da f 9e2ab76500), ngư ời sáng lập: { "$ref" : "người sáng lập", "$id" : ObjectId("5126bc054aed4da f 9e2ab76600), "$db" : "ngư ời dùng" } }

Trong ví dụ DBRef trỏ đến một tài liệu trong bộ sưu tập người sáng lập có trong trư ờng _id giá trị ObjectId("5126bc054aed4da f 9e2ab76600) và nằm

trong cơ sở dữ liệu người dùng.

- Embedded cho phép quản lý các mối quan hệ dữ liệu xem xét các tài liệu lồng vào nhau. MongoDB, trên thực tế, cho phép kết hợp nhiều tài liệu hơn trong một tài liệukhác dư ới dạng một trư ờng hoặc dư ới dạng ma trận. Các mô hình dữ liệu này không đư ợc chuẩn hóa và cho phép các ứng dụng truy xuất và thao tác các dữ liệu liên quan bằngmột thao tác duy nhất trên bộ sư u tập. MongoDB cũng sử dụng ký hiệu dấu chấm để truy cập các phần tử của một mảng và các trư ờng trong một tài liệu hợp nhất. Để truy cập một phần tử của mảng, bạn nối tên của mảng với chỉ số của phần tử thông qua dấu chấm, đặt tất cả trong dấu ngoặc kép: " < array > . <index > Để truy cập trường của một tài liệu con có ký hiệu, bạn nối tên của tài liệu phụ với tên của trường thông qua dấu chấm, đặt trong dấu ngoặc kép: " < tài liệu con > . < f lĩnh vực >

Với cả hai cách tiếp cận, quan hệ giữa các dữ liệu được quản lý bên trong chính mô hình dữ liệu. Do đó, nếu một truy vấn mới yêu cầu một kiểu quan hệ mới giữa các dữ liệu không được lập kế hoạch trước đó, thì toàn bộ mô hình dữ liệu phải được sửa đổi. Để giải quyết vấn đề như vậy, bắt đầu từ phiên bản 3.2, cộng đồng mã nguồn mở MongoDB đã giới thiệu hoạt động liên hợp giữa các bộ sưu tập. Đặc biệt MongoDB 3.2 đã được cải tiến với toán tử $ lookup. Những cải tiến này muốn mở rộng các tùy chọn để thựchiện phân tích trên cơ sở dữ liệu hoạt động tiếp tục duy trì các câu trả lời hiệu quả và nhanh chóng cho các truyvấn. Trong các phiên bản MongoDb trước, phép nối giữa hai bộ sưu tập đã được các nhà phát triển thực hiện ở lớp ứng dụng. Nói cách khác khi xem xét hai bộ sưu tập, chúng tôi có bộ sưu tập đầu tiên (chúng ta có thể gọi nó là "bộ sưu tập bên trái") chứa "_id" được lưu trữ các tài liệu của nhiều bộ sưu tập khác (bộ sưu tập bên phải). Bộ sưu tập bên trái chứa các tham chiếu để truy cập bộ sưu tập bên phải. Trước MongoDB 3.2, liên kết này đã đư ợc triển khai trong mã ứng dụng. MongoDB 3.2, thay vào đó, giới thiệu toán tử $ lookup hiện có thể được đưa vào như một giai đoạn trong một đường ống tổng hợp. Toán tử này cho phép kết hợp dữ liệu của các tập hợp bên trái và bên phải trong cơ sở dữ liệu. Như ng nhà điều hành mới này chỉ xem xét một trường hợp cụ thể về tiềm năng của "tham gia" cụ thể là "Tham gia trang bị bên ngoài bên trái". Left Outer Equi-Join là một "liên kết bên ngoài". Nó cung cấp một kết quả truy vấn trong đó tất cả các tài liệu của bộ sưu tập có liên quan. Hơn nữa, đây là một giải pháp đơn hướng "left→right" mà theo quan điểm của chúng tôi là một hạn chế vì người dùng phải luôn chú ý đến thứ tự của các toán hạng "nối". Trên thực tế, Left Outer Equi-Join cung cấp đầu ra tất cả các tài liệu của bộ sư u tập bên trái ngay cả khi nó không có bất kỳ khớp nào trên các trư ờng tài liệu của bộ sưu tập bên phải. Do đó, kết quả của truy vấn có thể chứa một lư ợng lớn tài liệu vô dụng, đặc biệt là trong bối cảnh Dữ liệu lớn. Thay vào đó, với công việc của chúngtôi, chúng tôi cung cấp một giải pháp cấp cao cho "liên kết bên trong" còn đư ợc gọi là "tham gia theta" bao gồm trong một sản phẩm Descartes đư ợc theo sau bởi một phép toán lựa chọn. Theo cách này, ngư ời dùng có thể quản lý các bộ sưu tập theo cách hiệu quả hơn, đồng thời có được một bộ sưu tập đã kết hợp chỉ bao gồm các tài liệu phù hợp với một điều kiện cụ thể trên một hoặc nhiều trư ờng. Mục tiêu chính của bài báo này là nghiên cứu chi phí được giới thiệu bởi một triển khai lớp ứng dụng thực hiện một phép nối bên trong trên các bộ sư u tập khác nhau.

## 5. Tham gia bên trong lớp ứng dụng cho MongoDB

Như đã biết, phép nối tự nhiên là một mệnh đề trong ngôn ngữ SQL kết hợp các bộ của hai hoặc nhiều quan hệ (bảng) trong một sản phẩm cartesian, theo các tiêu chí cụ thể được áp dụng cho các thuộc tính. Mặc dù không được hỗ trợ, phép nối tự nhiên có thể được thực thi bởi ứng dụng bên thứ ba bên ngoài được viết với bất kỳ ngôn ngữ lập trình nào được Giao diện chương trình ứng dụng MongoDB hỗ trợ (API), ngay cả khi phải trả giá bằng một

số hoạt động bổ sung. Liệt kê 1 cho thấy một ví dụ về truy vấn nối bên trong xem xét các bảng "Kỳ thi" và "Lớp học" có các thể hiện được mô tả tương ứng trong Hình 7 và 8 (chỉ 20 hàng đầu tiên).

```
SELECT e.id, e.Serial Number, e.Grade,
       c.Course, c.Professor, c.CourseNumb
FROM Exam e, Class c
WHERE e.CourseNumb=c.CourseNumb).
```

Liệt kê 1. Ví dụ về truy vấn nối bên trong.

```
mysql> SELECT * from Exam ;
+----+-------------+-------+-----------+
| id |Serial Number| Grade |CourseNumb|
+----+-------------+-------+-----------+
|  1 | 5000        |    18 |         8 |
|  2 | 5001        |    22 |         6 |
|  3 | 5002        |    24 |         8 |
|  4 | 5003        |    24 |         2 |
|  5 | 5004        |    30 |         4 |
|  6 | 5005        |    21 |         6 |
|  7 | 5006        |    25 |         5 |
|  8 | 5007        |    18 |         4 |
|  9 | 5008        |    27 |         9 |
| 10 | 5009        |    20 |         5 |
| 11 | 5000        |    18 |         3 |
| 12 | 5001        |    20 |         6 |
| 13 | 5002        |    28 |         7 |
| 14 | 5003        |    30 |         8 |
| 15 | 5004        |    20 |         2 |
| 16 | 5005        |    23 |         3 |
| 17 | 5006        |    28 |         3 |
| 18 | 5007        |    28 |         5 |
| 19 | 5008        |    23 |         6 |
| 20 | 5009        |    28 |         4 |
```

Hình 7. Bảng ví dụ "Exam" trên MySQL.

```
mysql> SELECT * from Class;
+----------+--------------------------+-------------+
|CourseNumb| Course                   | Professor   |
+----------+--------------------------+-------------+
|        1 | Computer Systems         | Villari     |
|        2 | Numerical Elaboration    | Serrano     |
|        3 | Telecommunication Systems| Serrano     |
|        4 | Power Electronics        | De Caro     |
|        5 | Operating Systems        | Scarpa      |
|        6 | Automated Checks         | Xibilia     |
|        7 | Computer Science Foundation| Scarpa    |
|        8 | Calculus                 | Di Bella    |
|        9 | Computers                | Iannizzotto |
+----------+--------------------------+-------------+
```

Hình 8. Bảng ví dụ "Class" trên MySQL.

Hình 9 cho thấy một ví dụ về kết quả của thao tác nối bên trong SQL được thực hiện trên MySQL xem xét các bảng “Bài kiểm tra” và “Lớp học” được lọc bởi trường “CourseNumb”.

| Id | Serial Number | Grade | Course | Professor | CourseNumb |
|---|---|---|---|---|---|
| 1 | 5000 | 18 | Calculus | Di Bella | 8 |
| 2 | 5001 | 22 | Automated Checks | Xibilia | 6 |
| 3 | 5002 | 24 | Calculus | Di Bella | 8 |
| 4 | 5003 | 24 | Numerical Elaboration | Serrano | 2 |
| 5 | 5004 | 30 | Power Electronics | De Caro | 4 |
| 6 | 5005 | 21 | Automated Checks | Xibilia | 6 |
| 7 | 5006 | 25 | Operating Systems | Scarpa | 5 |
| 8 | 5007 | 18 | Power Electronics | De Caro | 4 |
| 9 | 5008 | 27 | Computers | Iannizzotto | 9 |
| 10 | 5009 | 20 | Operating Systems | Scarpa | 5 |
| 11 | 5000 | 18 | Telecomunication Systems | Serrano | 3 |
| 12 | 5001 | 20 | Automated Checks | Xibilia | 6 |
| 13 | 5002 | 28 | Computer Science Foundation | Scarpa | 7 |
| 14 | 5003 | 30 | Calculus | Di Bella | 8 |
| 15 | 5004 | 20 | Numerical Elaboration | Serrano | 2 |
| 16 | 5005 | 23 | Telecomunication Systems | Serrano | 3 |
| 17 | 5006 | 28 | Telecomunication Systems | Serrano | 3 |

Hình 9. Kết quả của thao tác Inner Join được thực hiện trên MySQL khi xem xét “Exam” và “Class” quan hệ được lọc bởi trường “CourseNumb”.

```
> db.Exam.find()
{ "_id" : ObjectId("5311df3044aec0c560cda759"), "Serial Number": 5051, "Grade": 26,"CourseNumb": 8 }
{ "_id" : ObjectId("5311df3044aec0c560cda75a"), "Serial Number": 5052, "Grade": 23,"CourseNumb": 5 }
{ "_id" : ObjectId("5311df3044aec0c560cda75b"), "Serial Number": 5053, "Grade": 27,"CourseNumb": 3 }
{ "_id" : ObjectId("5311df3044aec0c560cda75c"), "Serial Number": 5054, "Grade": 27,"CourseNumb": 1 }
{ "_id" : ObjectId("5311df3044aec0c560cda75d"), "Serial Number": 5055, "Grade": 29,"CourseNumb": 9 }
{ "_id" : ObjectId("5311df3044aec0c560cda75e"), "Serial Number": 5056, "Grade": 22,"CourseNumb": 4 }
{ "_id" : ObjectId("5311df3044aec0c560cda75f"), "Serial Number": 5057, "Grade": 28,"CourseNumb": 6 }
{ "_id" : ObjectId("5311df3044aec0c560cda760"), "Serial Number": 5058, "Grade": 24,"CourseNumb": 9 }
{ "_id" : ObjectId("5311df3044aec0c560cda761"), "Serial Number": 5059, "Grade": 22,"CourseNumb": 4 }
{ "_id" : ObjectId("5311df3044aec0c560cda762"), "Serial Number": 5060, "Grade": 28,"CourseNumb": 2 }
{ "_id" : ObjectId("5311df3044aec0c560cda763"), "Serial Number": 5061, "Grade": 29,"CourseNumb": 4 }
{ "_id" : ObjectId("5311df3144aec0c560cda764"), "Serial Number": 5062, "Grade": 27,"CourseNumb": 8 }
{ "_id" : ObjectId("5311df3144aec0c560cda765"), "Serial Number": 5063, "Grade": 25,"CourseNumb": 8 }
{ "_id" : ObjectId("5311df3144aec0c560cda766"), "Serial Number": 5064, "Grade": 19,"CourseNumb": 2 }
{ "_id" : ObjectId("5311df3144aec0c560cda767"), "Serial Number": 5065, "Grade": 19,"CourseNumb": 1 }
{ "_id" : ObjectId("5311df3144aec0c560cda768"), "Serial Number": 5066, "Grade": 27,"CourseNumb": 3 }
{ "_id" : ObjectId("5311df3144aec0c560cda769"), "Serial Number": 5067, "Grade": 28,"CourseNumb": 4 }
{ "_id" : ObjectId("5311df3144aec0c560cda76a"), "Serial Number": 5068, "Grade": 23,"CourseNumb": 1 }
```

Hình 10. Ví dụ về bộ sưu tập Bài kiểm tra.

```
> db.Class.find()
{ "_id" : ObjectId("529b111d10dc0424c9e9f8e4"), "NumCorso" : 1,
"Course" : "Computer Systems", "Professor " : "Villari" }
{ "_id" : ObjectId("529b112c10dc0424c9e9f8e5"), "NumCorso" : 2,
"Course" : "Numerical Elaboration", "Professor " : "Serrano" }
{ "_id" : ObjectId("529b113c10dc0424c9e9f8e6"), "NumCorso" : 3,
"Course" : "Telecommunication Systems" , "Professor " : "Serrano" }
{ "_id" : ObjectId("529b115110dc0424c9e9f8e7"), "NumCorso" : 4,
"Course" : "Power Electronics", "Professor " : "De Caro" }
{ "_id" : ObjectId("529b116410dc0424c9e9f8e8"), "NumCorso" : 5,
"Course" : "Operating Systems", "Professor " : "Scarpa" }
{ "_id" : ObjectId("529b117310dc0424c9e9f8e9"), "NumCorso" : 6,
"Course" : "Automated Checks", "Professor " : "Xibilia" }
{ "_id" : ObjectId("529b118610dc0424c9e9f8ea"), "NumCorso" : 7,
"Course" : "Computer Science Foundation", "Professor " : "Scarpa" }
{ "_id" : ObjectId("529b119010dc0424c9e9f8eb"), "NumCorso" : 8,
"Course" : "Calculus", "Professor " : "Di Bella" }
{ "_id" : ObjectId("529b119c10dc0424c9e9f8ec"), "NumCorso" : 9,
"Course" : "Computers", "Professor " : "Iannizzotto" }
```

Hình 11. Ví dụ về tập hợp Lớp.

Thủ tục Inner Join cho phép thực hiện thao tác Inner Join ở lớp ứng dụng xem xét hai tập hợp MongoDB được hiển thị trong Thuật toán 1. Nó lấy làm đầu vào ba đối tượng DBCollection ,tương ứng đại diện cho hai tập hợp sẽ được kết hợp, collection1 và collection2, và tập hợp kết quả , tức là, đã tham gia vào Bộ sư u tập. Hơn nữa, Thuật toán 1 lấy đầu vào cũng là một Chuỗi, tức là joinField, đại diện cho tên của trư ờng để khớp các tài liệu của collection1 và các tài liệu của collecition2. Trong dòng mã 2, con trỏ DBCursor1 được tạo bao gồm tất cả các tài liệu của collection1. Các dòng mã 3–7 trích xuất tài liệu từ collection1 trong mảng DBObject [] 1. Trong dòng mã 8, con trỏ DBCursor2 đư ợc tạo bao gồm tất cả các tài liệu của collection2. Các dòng mã 9–13 trích xuất tài liệu từ collection2 trong DBObject [] array2 Các dòng mã 14–29 điều chỉnh hoạt động nối bên trong con collection1 và collection2 có tài liệu khớp với trường joinField. Vì thủ tục như vậy thực hiện thao tác kết hợp bên trong trên hai tập hợp MongoDB ở lớp ứng dụng, nó giới thiệu chi phí do các thao tác đọc / ghi trong MongoDB.

**Algorithm 1** Procedure performing an Inner Join operation among two collections in MongoDB.

```
1: procedure void join(DBCollection collection1, DBCollection collection2, DBCollection collollectionJoined, String joinField )
2:     DBCursor cursor1 ← collection1.find() ;
3:     int size1 ← cursor1.size();
4:     DBObject[] array1 ← new DBObject[size1];
5:     for i = 0 to i < size1 do
6:         array1[i] ← cursor1.next();
7:     end for
8:     DBCursor cursor2 ← collection2.find();
9:     int size2 ← cursor2.size();
10:    DBObject[] array2 ← new DBObject[size2];
11:    for i = 0 to i < size2 do
12:        array2[i] ← cursor2.next();
13:    end for
14:    DBObject obj1 ← (DBObject) JSON.parse("{}");
15:    for i = 0 to i < size1 do
16:        if array1[i].containsField(joinField) then
17:            obj1.putAll(array1[i]);
18:            for j = 0 to i < size2 do
19:                if array2[i].containsField(joinField) then
20:                    if array1[i].get(joinField).equals(array2[j].get(joinField)) then
21:                        obj1.putAll(array2[i]) ;
22:                        obj1.removeField("_id") ;
23:                        collectionJoined.insert(obj1);
24:                        obj1 ← JSON.parse("{}");
25:                    end if
26:                end if
27:            end for
28:        end if
29:    end for
30:    end procedure
```

Thủ tục Tham gia bên trong MongoDB cho phép thực thi một truy vấn tư ơng tự như truy vấn đư ợc thể hiện trong Liệt kê 1. Kết quả của hoạt động Tham gia bên trong MongoDB xem xét các bộ sư u tập "Bài kiểm tra" và "Lớp học"được lọc bởi trư ờng "CourseNumb" đư ợc hiển thị trong Hình 12

```
b.JoinSubject.find()
_id" : ObjectId("5311c56044ae806088cecec6"),"Serial Number" : 5003, "Grade" : 23,"CourseNumb": 9, "Course" :
Computers ",  "Professor"  : "Iannizzotto" }
_id" : ObjectId("5311c56044ae806088cecec7"),"Serial Number" : 5004, "Grade" : 26,"CourseNumb": 2, "Course" :
Numerical Elaboration ",  "Professor"  : "Serrano" }
_id" : ObjectId("5311c56044ae806088cecec8"),"Serial Number" : 5005, "Grade" : 20,"CourseNumb": 8, "Course" :
Calculus", "Professor"  : "Di Bella" }
_id" : ObjectId("5311c56044ae806088cecec9"),"Serial Number" : 5006, "Grade" : 24,"CourseNumb": 2, "Course" :
Numerical Elaboration ",  "Professor"  : "Serrano" }
_id" : ObjectId("5311c56044ae806088cececa"),"Serial Number" : 5007, "Grade" : 18,"CourseNumb": 1, "Course" :
 "Computer Systems", "Professor"  : "Villari" }
_id" : ObjectId("5311c56044ae806088cececb"),"Serial Number" : 5008, "Grade" : 24,"CourseNumb": 7, "Course" :
 Computer Science Foundation","Professor"  : "Scarpa" }
_id" : ObjectId("5311c56044ae806088cececc"),"Serial Number" : 5009, "Grade" : 27,"CourseNumb": 3, "Course" :
" Telecommunication Systems", "Professor"  : "Serrano" }
_id" : ObjectId("5311c56044ae806088cececd"),"Serial Number" : 5000, "Grade" : 18,"CourseNumb": 1, "Course" :
 "Computer Systems", "Professor"  : "Villari" }
_id" : ObjectId("5311c56044ae806088cecece"),"Serial Number" : 5001, "Grade" : 19,"CourseNumb": 1, "Course" :
 "Computer Systems", "Professor"  : "Villari" }
_id" : ObjectId("5311c56044ae806088cececf"),"Serial Number" : 5002, "Grade" : 22,"CourseNumb": 4, "Course" :
"Electronic Power", "Professor"  : "De Caro" }
```

**Hình 12**. Kết quả của MongoDB Inner Join đã thực hiện MongoDB, xem xét “Exam” và “Class”bộ sưu tập được lọc bởi trường "CourseNumb".

## 6. Thí nghiệm

Mục tiêu của Phần này là so sánh hiệu suất MongoDB và MySQL về thời gian phản hồi và đặc biệt, hiệu suất bị ảnh hưởng như thế nào bởi các hoạt động kết hợp. Để cung cấp loại này của phân tích, chúng tôi đã nghiên cứu cả thao tác chèn và truy vấn trên cùng một mô hình dữ liệu. Đặc biệt, chúng tôi đã phân tích lược đồ cơ sở dữ liệu (Class-Exam) được thảo luận trong Phần trước. Về MySQL, thông qua lệnh SQL CREATE DATABASE mysqldb, chúng tôi đã tạo cơ sở dữ  liệu có tên mysqldb, sau đó chúng tôi tạo bảng Exam bao gồm các thuộc tính id, Serial Number, Grade và CourseNumb và bảng Class bao gồm các thuộc tính CourseNumb, Course và Professor. Một ràng buộc khóa ngoại cũng được tạo trong Bảng lớp để thuộc tính Exam.CourseNumb tham chiếu đến thuộc tính Class.CourseNumb. Về MongoDB, vì nó là không có lược đồ, nên không phải tạo lược đồ cơ sở dữ liệu ưu tiên, vì cấu trúc dữ liệu thu thập được tạo khi tài liệu đầu tiên được chèn

Cấu trúc dữ liệu đã xác định cho các thử nghiệm, chúng tôi đã đánh giá thời gian phản hồi của cơ sở dữ liệu cho hai loại hoạt động là: (1) chèn dữ liệu và (2) truy xuất dữ liệu bằng phép nối bên trong.

Việc đánh giá việc chèn dữ liệu cho phép chúng tôi mô tả hành vi của hai DB, nêu bật những ưu điểm và nhược điểm của hai cách tiếp cận. Việc đánh giá truy xuất dữ liệu với phép nối bên trong cho phép chúngtôi đưa ra những cân nhắc của mình về việc triển khai phép nối bên trong đư ợc đề xuất cho MongoDB.

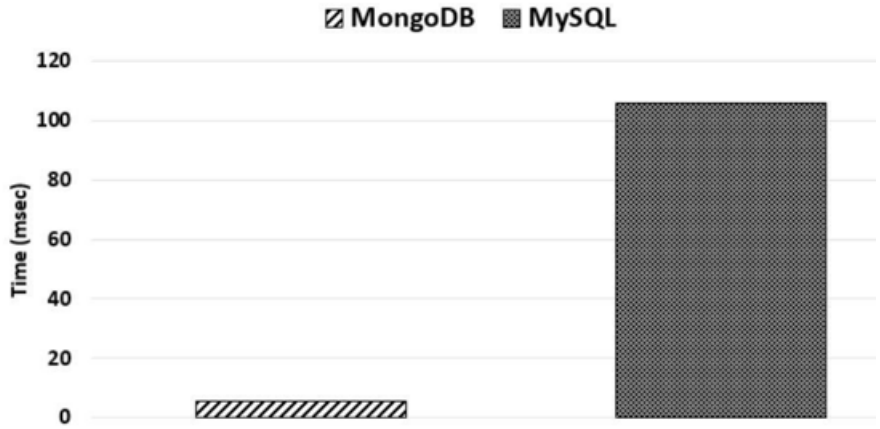
Bước đánh giá đầu tiên liên quan đến thao tác chèn dữ liệu. Đặc biệt, chúng tôi thiết lập một vùng thử nghiệm nơi một ứng dụng Java bên ngoài chèn cùng một dữ liệu:

- một tài liệu (đại diện cho một kỳ thi) vào bộ sư u tập "Kỳ thi" của cơ sở dữ liệu mymongodb;
- một bản ghi (đại diện cho một kỳ thi) vào bảng "Kỳ thi" của cơ sở dữ liệu mysqldb.

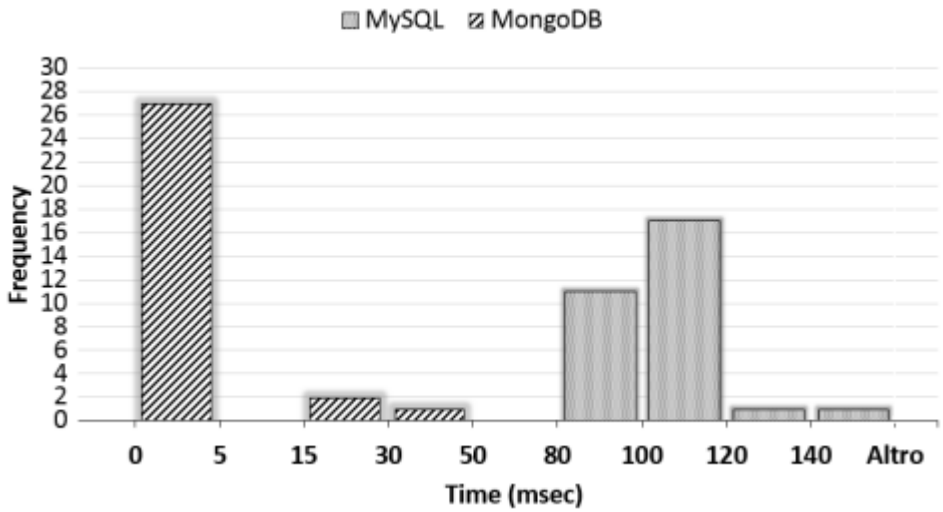
Mỗi thử nghiệm đánh giá thời gian phản hồi cho các hoạt động chèn và thử nghiệm được lặp lại 30 lần để tính giá trị trung bình với khoảng tin cậy 95%. Cụ thể, từng phần dữ liệu đã bị xóa khỏi mỗi cơ sở dữ liệu trước khi được chèn lại.

Hình 13 cho thấy biểu đồ báo cáo thời gian chèn trung bình đo được. Chúng ta có thể thấy rằng việc chèn các tài liệu trong MongoDB rất hiệu quả so với việc chèn các bản ghi trong MySQL. Điều này là do tổ chức dữ liệu không có sơ đồ trong MongoDB cho phép lưu trữ thông tin giảm bớt quá trình xử lý và kiểm soát. Đặc biệt, vì MongoDB không hiển thị chi phí điều khiển do các ràng buộc về khóa ngoại nên nó mang lại hiệu suất tốt hơn so với MySQL. Do đó, thao tác chèn một tài liệu trong mymongodb mất khoảng 5 mili giây trong khi thao tác tư ơng tự trong mysqldb mất khoảng 105 mili giây; vì vậy thời gian chèn trên MongoDB là khoảng 4% trên MySQL. Biểu đồ hiển thị trong Hình 14 cho thấy sự phân bố của các phép đo trên thời gian chèn để cho thấy các giá trị đo đư ợc lan truyền như thế nào trong phạm vi phép đo. Đây là thông tin bổ sung liên quan đến việc ước tính thời gian chèn

trung bình đánh giá hành vi ổn định / không ổn định của hệ thống xung quanh giá trị trung bình. Đặc biệt, Hình 14 cho thấy hoạt động của MongoDB ổn định hơn hoạt động của MongoDB với thời gian đo được nhiều nhất trong khoảng [0–5] ms. Trên thực tế, chúng ta có thể quan sát bằng cách nào khi nhìn vào biểu đồ tần số khi MongoDB 27 tác vụ chèn lặp lại 30 lần trải qua thời gian phản hồi trung bình trong khoảng từ 0 đến 5 ms. Trong khi đó, liên quan đến MySQL, chúng tacó thể quan sát thấy rằng trong 17 nhiệm vụ chèn lặp đi lặp lại là 30, chúng tôi đã trải qua thời gian phản hồi trung bình từ 100 đến 120 mili giây.




Hình 13. Thời gian chèn trung bình của các tài liệu và bản ghi tương ứng trong MongoDB và MySQL.



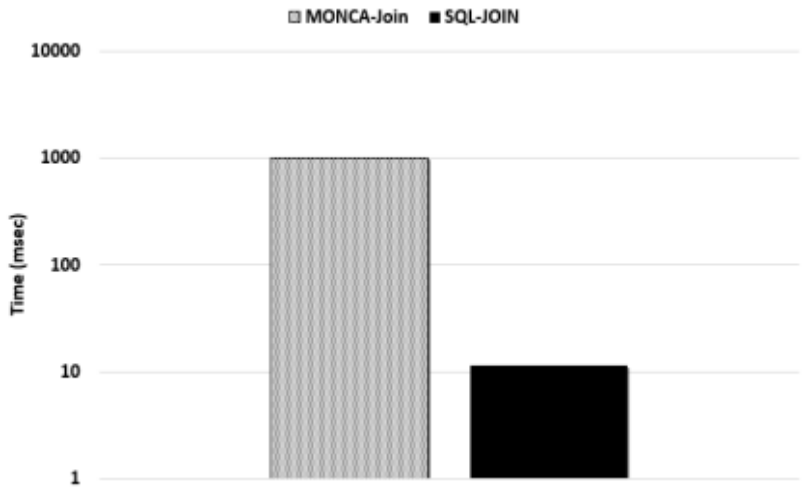

Hình 14. Biểu đồ tần suất chèn của các tài liệu và bản ghi tương ứng trong MongoDB và MySQL.

Bước đánh giá thứ hai có liên quan đến thao tác nối bên trong. Thao tác nối bên trong được thựchiện trong MongoDB bằng cách sử dụng Thuật toán 1 được triển khai bằng ngôn ngữ Java ở lớp ứng dụng xem xét cùng một tập dữ liệu đã thảo luận trước đó. Đặc biệt, chúng tôi đã xem xét 1000 bản ghi/tài liệu trong bảng/bộ sưu tập “Class” và 9 bản ghi/tài liệu trong bảng/bộ sưu tập “Exam” choMySQL/MongoDB.Như đã thảo luận trước đây, giá trị gia tăng của Thuật toán 1 là cung cấp chức năng mới không có sẵn cho MongoDB. Ở đây, chúng tôi điều tra tác động của nóvề hiệu suất. Trên giường thử nghiệm, chúng tôi đã thực hiện:

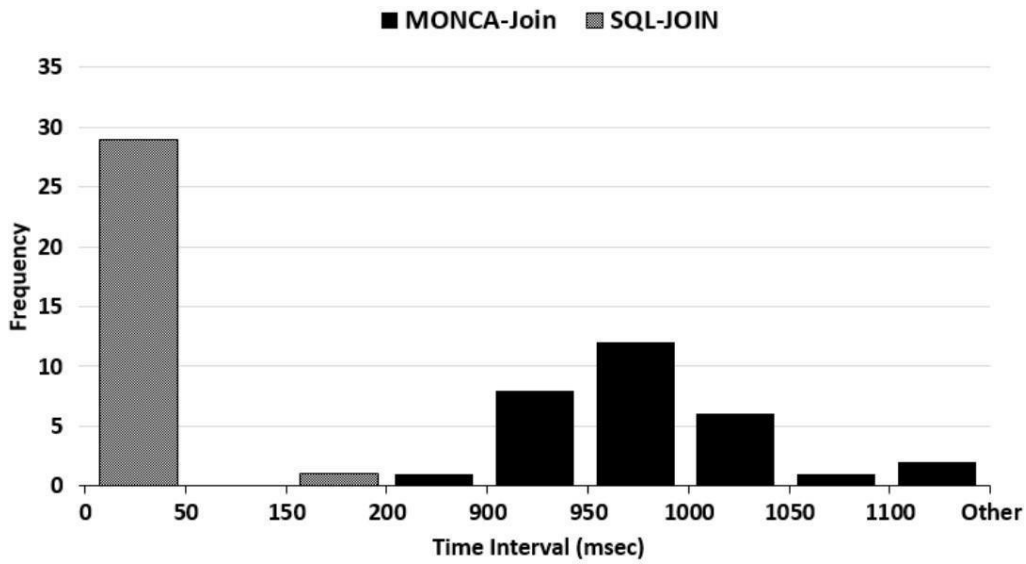
- 30 thao tác nối bên trong lặp đi lặp lại trong MongoDB giữa các bộ sưu tập “Exam” và “Class” một cơ sở dữ liệu mymongodb.
- 30 thao tác nối bên trong lặp đi lặp lại trong MySQL giữa các bảng “Exam” và

"Class" của một cơ sở dữ liệu mysqldb.

Hình 15 cho thấy chi phí hoạt động được giới thiệu bởi thao tác nối bên trong được thực thi ở lớp ứng dụng so với thao tác nối bên trong gốc được thực thi trực tiếp trong MySQL. Thời gian phản hồi trungbình cho MySQL là khoảng 11 ms, trong khi thời gian phản hồi trung bình cho MongoDb là khoảng 1000 ms. Hình 16 nhấn mạnh rằng hoạt động nối bên trong ổn định hơn trong MySQL, trong khi thời gian phản hồi trong MongoDB trải rộng trên một tập hợp lớn các giá trị, do đó làm cho hiệu suất rất thay đổi trong các truy vấn.




Hình 15. So sánh thời gian phản hồi trung bình giữa MySQL và MongoDB (tầng ứng dụng) tham gia hoạt động.




Hình 16. Biểu đồ tần suất hiển thị thời gian phản hồi của MySQL và MongoDB (tầng ứng dụng) tham gia hoạt động.

## 7. Thảo luận và kết luận

Cơ sở dữ liệu NoSQL ngày càng trở nên phổ biến do tính linh hoạt và khả năng mở rộng của chúng trong việc quản lý dữ liệu lớn không đồng nhất đến từ mạng xã hội,Điện toán đám mây, IoT và các kịch bản mới nổi khác. Mặc dù, hầu hết các giải pháp cơ sở dữ liệu NoSQL phổ biến đều cung cấp hiệu suất tốt về cả việc chèn và truy xuất dữ liệu, nhưng thường thì chúng không hỗ trợ các thao tác nối một cách tự nhiên. Đây là trường hợp của MongoDB gần đây đãgiới thiệu khái niệm nối trái giữa các bộ sưu tập, như ng tại thời điểm viết bài báo này, nó không hỗ trợ thao tác nối bên trong phổ biến. Do đó, để truy xuất các loại thông tin khác nhau cùng một lúc,dữ liệu phải đư ợc lưu trữ trong một bộ sư u tập lớn đư ợc tổ chức trong các đối tượng json lồng nhau. Ngay cả khi hoạt động, điều này hạn chế các kiến trúc sư dữ liệu trong việc tổ chức dữ liệu, chẳng hạn như giải quyết tính độc lập logic của dữ liệu và sự khác biệt của các chính sách truy cập dữ liệu ở cấp bộ sư u tập hoặc hạn chế tích hợp với các hệ thống cũ. Trong bài báo này, chúng tôi đã trình bày một thuật toán mới để khắc phục loại hạn chế này bằng cách triển khai chức năng nối bên trong cũng trong NoQuery MongoDB. Điều này làm cho thiết kế của DB linh hoạt hơn, mang lại cơ hộimới cho kiến trúc sư dữ liệu trong việc tổ chức dữ liệu thành DBs.

Sự đóng góp của bài báo này cho trạng thái nghệ thuật gồm hai phần: (1) chúng tôi đã triển khai một chức năng mới không có sẵn trong MongoDB (và trong nhiều DB NoSQL khác), chứng tỏ tính linh hoạt hơn đối với các kiến trúc sư dữ liệu trong việc tổ chức dữ liệu; (2) chúng tôi phân tích tác động của hoạt động nối bên trong hiệu suất của MongoDB và chúng tôi so sánh hiệu suất đó với hiệu suất của MySQL. Cụ thể, các thử nghiệm chứng minh rằng mặc dù các tác vụ chèn trong MongoDB thuận tiện hơn trong MySQL, nhưng các tác vụ truy xuất dữ liệu yêu cầu các thao tác nối bên trong giữa các bộ sư u tập khác nhau có thời gian phản hồi lớn hơn khoảng 10 lần so với thời gian nhận được trong MySQL.

Từ kết quả thử nghiệm, rõ ràng là cách tiếp cận thiết kế tốt nhất để tổ chức dữ liệu trong MongoDB là có một bộ sưu tập với các đối tư ợng json lồng nhau. Tuy nhiên, đây không phải lúc nào cũng là một cách tiếp cận tốt, chẳng hạn như nếu cần phân biệt các chính sách truy cập dữ liệu ở cấp độ thu thập. Giờ đây, các kiến trúc sư dữ liệu có cơ hộiquản lý dữ liệu theo cách linh hoạt hơn, nhưng họ phải cân bằng giữa tính linh hoạt với các vấn đề về hiệu suất. Với bài báo này, chúng tôi hy vọng sẽ kích thích cộng đồng nguồn mở MongoDB hướng tới việc tạo ra một toán tử tham gia bên trong bản địa. Vì những lý do này, từ quan điểm của các ứng dụng kế thừa Internet trong tư ơng lai mà một mặt cần chuyển từ DBMS giống SQL sang MongoDB, nhưng mặt khác yêu cầu bảo toàn các mô hình dữ liệu của chúng, một triển khai nguyên bản của hoạt động kết nối bên trong trong MongoDB rất cần thiết.Các nghiên cứu trong tương lai của chúng tôi sẽ được định hướng để fullfil một khoảng trống như vậy. Tuy nhiên, trong bước tiếp theo, chúng tôi dự định thực hiện một đánh giá lớn về thuật toán được đề xuất với các bộ dữ liệu khác nhau và trong các tình huống hoạt động khác nhau. Hơn nữa, chúng tôi cũng sẽ phân tích các hạn chế bổ sung của MongoDB và các DB NoSQL khác.

Đóng góp của tác giả: AC thiết kế kiến trúc của hệ thống; MF thực hiện thí nghiệm; MV đề xuấtnghiên cứu so sánh và giám sát bài báo.Kinh phí: Nghiên cứu này đư ợc tài trợ bởi Bộ Y tế Ý, tên dự án “Bệnh nhân chấn thư ơng não mắc phải nặng có đư ợc hư ởng lợi từ Phục hồichức năng từ xa không? Nghiêncứu phân tích hiệu quả chi phí ”, số tài trợ GR-2016-02361306.

Xung đột lợi ích: Các tác giả tuyên bố không có xung đột lợi ích.

**Ngư ời giới thiệu**


1. Mohamed, M.A.; Altrafi, O.G.; Ismail, M.O. Relational vs. nosql databases: A survey. Int. J. Comput. Inf. Technol. 2014, 3, 598–601.

2. MongoDB Atlas. Deploy a Fully Managed Cloud Database in Minutes. Available online: www.mongodb.org (accessed on 10 January 2019).

3. Carnevale, L.; Celesti, A.; Di Pietro, M.; Galletta, A. How to conceive future mobility services in smart cities according to the Fiware frontiercities experience. IEEE Cloud Comput. 2018, 5, 25–36. [CrossRef]

4. Wan, J.; Li, J.; Hua, Q.; Celesti, A.; Wang, Z. Intelligent equipment design assisted by Cognitive Internet of Things and industrial big data. Neural Comput. Appl. 2018. [CrossRef]

5. Galletta, A.; Carnevale, L.; Celesti, A.; Fazio, M.; Villari, M. A Cloud-Based System for Improving Retention Marketing Loyalty Programs in Industry 4.0: A Study on Big Data Storage Implications. IEEE Access 2017, 6, 5485–5492. [CrossRef]

6. Carnevale, L.; Calabro, R.; Celesti, A.; Leo, A.; Fazio, M.; Bramanti, P.; Villari, M. Towards Improving Robotic-Assisted Gait Training: Can Big Data Analysis Help us? IEEE Internet Things J. 2018. [CrossRef]

7. Celesti, A.; Galletta, A.; Carnevale, L.; Fazio, M.; Lay-Ekuakille, A.; Villari, M. An IoT cloud system for traffic monitoring and vehicular accidents prevention based on mobile sensor data processing. IEEE Sens. J. 2018, 18, 4795–4802. [CrossRef]

8. Celesti, A.; Fazio, M.; Romano, A.; Villari, M. A hospital cloud-based archival information system for the efficient management of HL7 big data. In Proceedings of the 39th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), Opatija, Croatia, 30 May–3 June 2016.

9. Mulfari, D.; Celesti, A.; Villari, M.; Puliafito, A. How cloud computing can support on-demand assistive services. In Proceedings of the 10th International Cross-Disciplinary Conference on Web Accessibility, Rio de Janeiro, Brazil, 13–15 May 2013. [CrossRef]

10. Celesti, A.; Celesti, F.; Fazio, M.; Bramanti, P.; Villari, M. Are Next-Generation Sequencing Tools Ready for the Cloud? Trends Biotechnol. 2017, 35, 486–489. [CrossRef] [PubMed]

11. Celesti, F.; Celesti, A.; Wan, J.; Villari, M. Why Deep Learning Is Changing the Way to Approach NGS Data Processing: A Review. IEEE Rev. Biomed. Eng. 2018, 11, 68–76. [CrossRef] [PubMed]

12. Gyorodi, C.; Gyorodi, R.; Pecherle, G.; Olah, A. A comparative study: MongoDB vs. MySQL. In Proceedings of the 13th International Conference on Engineering of Modern Electric Systems (EMES), Oradea, Romania, 11–12 June 2015; pp. 1–6. [CrossRef]

13. Katkar, M.; Kutchhii, S.; Kutchhii, A. Performance Analysis for NoSQL and SQL. Int. J. Innov. Emerg. Res. Eng. 2015, 2, 12–17.

14. Pankaj Sareen, P.K. NoSQL Database and its Comparison with SQL Database. Int. J. Comput. Sci. Commun. Netw. 2015, 5, 293–298.

15. Celesti, A.; Fazio, M.; Romano, A.; Bramanti, A.; Bramanti, P.; Villari, M. An OAIS-Based Hospital Information System on the Cloud: Analysis of a NoSQL Column-Oriented Approach. IEEE J. Biomed. Health Inform. 2018, 22, 912–918. [CrossRef] [PubMed

16. Sánchez-de-Madariaga, R.; Muñoz, A.; Castro, A.L.; Moreno, O.; Pascual, M. Executing Complexity-Increasing Queries in Relational (MySQL) and NoSQL (MongoDB and EXist) Size-Growing ISO/EN 13606 Standardized EHR Databases. J. Vis. Exp. 2018, 133, 57439. [CrossRef] [PubMed] Future Internet 2019, 11, 83 17 of 17

17. Sangeeta Gupta, G. Correlation and comparison of nosql specimen with relational data store. Int. J. Res. Eng. Technol. 2015, 4, 1–5.

18. Kim, C.; Shim, K. Supporting set-valued joins in NoSQL using MapReduce. Inf. Syst. 2015, 49, 52–64. [CrossRef]

19. Ntarmos, N.; Patlakas, I.; Triantafillou, P. Rank join queries in NoSQL databases. In Proceedings of the VLDB Endowment, Hangzhou, China, 1–5 September 2014; Volume 7, pp. 493–504.

20. Sahal, R.; Nihad, M.; Khafagy, M.H.; Omara, F.A. iHOME: Index-Based JOIN Query Optimization for Limited Big Data Storage. J. Grid Comput. 2018, 16, 345–380. [CrossRef]

21. Vathy-Fogarassy, A.; HugyÁk, T. Uniform data access platform for SQL and NoSQL database systems. Inf. Syst. 2017, 69, 93–105. [CrossRef]

22. Stanescu, L.; Brezovan, M.; Burdescu, D. Automatic mapping of MySQL databases to NoSQL MongoDB. In Proceedings of the 2016 Federated Conference on Computer Science and Information Systems, Gda ´nsk, Poland, 11–14 September 2016; pp. 837–840.

23. Unbehauen, J.; Martin, M. Executing SPARQL queries over mapped document stores with SparqlMap-M. In Proceedings of the 12th International Conference on Semantic Systems, Leipzig, Germany, 12–15 September 2016; pp. 137–144.

24. Hiriyannaiah, S.; Siddesh, G.M.; Anoop, P.; Srinivasa, K.G. Semi-structured data analysis and visualisation using NoSQL. Int. J. Big Data Intell. 2018, 5, 133–142. [CrossRef]

25. Brewer, E. Towards Robust Distributed Systems. In Proceedings of the Nineteenth ACM Symposium on Principles of Distributed Computing, Portland, OR, USA, 16–19 July 2000; pp. 7–10.

26. MongoDB Essentials. Available online: https://dinfratechsource.com/2018/11/10/mongodb-essentials/ (accessed on 10 January 2019) .