HỌC VIỆN NÔNG NGHIỆP VIỆT NAM

**KHOA CÔNG NGHỆ THÔNG TIN**

------- 🙞🕮🙜 -------

****

**KHÓA LUẬN TỐT NGHIỆP**

**ĐỀ TÀI: Tìm hiểu về NoSQL và ứng dụng**

**HÀ NỘI – 2018**

HỌC VIỆN NÔNG NGHIỆP VIỆT NAM

**KHOA CÔNG NGHỆ THÔNG TIN**

------- 🙞🕮🙜 -------



**KHÓA LUẬN TỐT NGHIỆP**

**ĐỀ TÀI: Tìm hiểu về NoSQL và ứng dụng**

**Sinh viên thực hiện : PHẠM THU NGÂN**

**Khóa : K58**

**Ngành : CÔNG NGHỆ THÔNG TIN**

**Chuyên ngành : TIN HỌC**

**Gv hướng dẫn : Ths . HOÀNG THỊ HÀ**

**HÀ NỘI – 2018**

**LỜI CẢM ƠN**

Trước hết em xin chân thành cảm ơn ThS. Hoàng Thị Hà - Giảng viên Bộ môn công nghệ phần mềm, khoa Công nghệ thông tin, Học Viện Nông Nghiệp Việt Nam - người đã tận tình hướng dẫn chỉ bảo và giúp đỡ em rất nhiều trong quá trình làm đề tài khóa luận tốt nghiệp này để em có thể hoàn thành được đề tài của mình.

Em xin cảm ơn các thầy cô giáo của Khoa Công Nghệ Thông Tin - Học Viện Nông Nghiệp Việt Nam đã cho em kiến thức, kinh nghiệm quý báu trong suốt thời gian học tập để em có thể hoàn thành tốt đề tài.

Trong quá trình tìm hiểu và làm đề tài, do những nguyên nhân khách quan nên đề tài không tránh được những thiết sót. Em rất mong được thầy cô chỉ dẫn, đóng góp những ý kiến quý báu để em hoàn thiện hơn bài báo cáo của mình.

***Em xin chân thành cảm ơn!***

MỤC LỤC

[MỤC LỤC 4](#_Toc531628921)

[DANH MỤC HÌNH ẢNH 6](#_Toc531628922)

[DANH MỤC BẢNG 8](#_Toc531628923)

[DANH MỤC CHỮ VIẾT TẮT 8](#_Toc531628924)

[PHẦN I. MỞ ĐẦU 9](#_Toc531628925)

[1.1. Đặt vấn đề 9](#_Toc531628926)

[1.2. Mục đích và yêu cầu 10](#_Toc531628927)

[1.2.1. Mục đích 10](#_Toc531628928)

[1.2.2. Yêu cầu 10](#_Toc531628929)

[PHẦN II. TỔNG QUAN TÌNH HÌNH NGHIÊN CỨU TRONG VÀ NGOÀI NƯỚC 10](#_Toc531628930)

[2.1. Tình hình nghiên cứu trong nước. 10](#_Toc531628931)

[2.2. Tình hình nghiên cứu ngoài nước 11](#_Toc531628932)

[PHẦN III. NỘI DUNG VÀ PHƯƠNG PHÁP NGHIÊN CỨU 12](#_Toc531628933)

[3.1. Địa điểm và thời gian nghiên cứu (thực tập) 12](#_Toc531628934)

[3.2. Nội dung nghiên cứu 12](#_Toc531628935)

[3.3. Phương pháp nghiên cứu 13](#_Toc531628936)

[PHẦN IV. KẾT QUẢ VÀ THẢO LUẬN 13](#_Toc531628937)

[4.1. Tổng quan về NoSQL 13](#_Toc531628938)

[4.1.1. NoSQL là gì? 13](#_Toc531628939)

[4.1.2. Đặc điểm 14](#_Toc531628940)

[4.1.3. Một số khái niệm mới trong NoSQL 15](#_Toc531628941)

[4.1.4. Một số thuật ngữ liên quan 15](#_Toc531628942)

[4.1.5. Ưu – nhược điểm của NoSQL 17](#_Toc531628943)

[4.1.6. So sánh NoSQL và cơ sở dữ liệu quan hệ 18](#_Toc531628944)

[4.1.7. Các dạng NoSQL cơ bản 19](#_Toc531628945)

[4.1.8. Làm sao để chọn một giải pháp cơ sở dữ liệu tốt? 23](#_Toc531628946)

[4.1.9. Tìm hiểu một số loại NoSQL phổ biến 26](#_Toc531628947)

[4.2. Hệ quản trị cơ sở dữ liệu MongoDB 31](#_Toc531628948)

[4.2.1. Hệ quản trị cơ sở dữ liệu MongoDB là gì? 31](#_Toc531628949)

[4.2.2. Các thành phần chính của MongoDB 31](#_Toc531628950)

[4.2.3. So sánh giữa CSDL quan hệ và MongoDB 33](#_Toc531628951)

[4.2.4. Các kiểu dữ liệu trong MongoDB 33](#_Toc531628952)

[4.2.5. Một số câu lệnh thường dùng 36](#_Toc531628953)

[4.2.6. MongoDB hoạt động như thế nào 38](#_Toc531628954)

[4.2.7. Ưu và nhược điểm của MongoDB 39](#_Toc531628955)

[4.2.8. Khi nào nên và không nên sử dụng MongoDB 41](#_Toc531628956)

[4.2.9. Các công cụ trực quan 41](#_Toc531628957)

[4.2.10. Cài đặt MongoDB 41](#_Toc531628958)

[4.3. Demo ứng dụng của NoSQL 42](#_Toc531628959)

[4.3.1. Phát biểu bài toán 42](#_Toc531628960)

[4.3.2. Đặc tả yêu cầu bài toán 43](#_Toc531628961)

[4.3.3. Phân tích, thiết kế hệ thống 53](#_Toc531628962)

[PHẦN V. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 79](#_Toc531628963)

[5.1. Kết luận 79](#_Toc531628964)

[5.2. Hướng phát triển 79](#_Toc531628965)

[TÀI LIỆU THAM KHẢO 80](#_Toc531628966)

DANH MỤC HÌNH ẢNH

[Hình 1: Cơ sở dữ liệu NoSQL 11](#_Toc531936290)

[Hình 2: Phân loại NoSQL 17](#_Toc531936291)

[Hình 3: Key - values Database 18](#_Toc531936292)

[Hình 4: Column - family Databse 19](#_Toc531936293)

[Hình 5: Document Database 20](#_Toc531936294)

[Hình 6: Graph Database 21](#_Toc531936295)

[Hình 7: Thành phần chính của MongoDB 29](#_Toc531936296)

[Hình 8: Kiến trúc của MongoDB 36](#_Toc531936297)

[Hình 9: Tải file cài đặt MongoDB 41](#_Toc531936298)

[Hình 10: Bắt đầu cài đặt MongoDB 42](#_Toc531936299)

[Hình 11: Đồng ý với thỏa thuận 42](#_Toc531936300)

[Hình 12: Chọn kiểu setup 43](#_Toc531936301)

[Hình 13: Thư mục lưu trữ 44](#_Toc531936302)

[Hình 14: Cài đặt service 44](#_Toc531936303)

[Hình 15: Cài đặt MongoDB Compass 45](#_Toc531936304)

[Hình 16. Isntall 46](#_Toc531936305)

[Hình 17: Kết thúc 46](#_Toc531936306)

[Hình 18: Chèn php\_mongo.dll vào php.ini 47](#_Toc531936307)

[Hình 19: Kết nối cơ sở dữ liệu MongoDB và php 50](#_Toc531936308)

[Hình 20: Add new data 51](#_Toc531936309)

[Hình 21: Trang chủ 53](#_Toc531936310)

[Hình 22: Update data 53](#_Toc531936311)

[Hình 23: Delete data 56](#_Toc531936312)

[Hình 24: Giao diện khi mở Robo 3T lần đầu 57](#_Toc531936313)

[Hình 25: Tạo kết nối đến MongoDB server 58](#_Toc531936314)

[Hình 26: Kiểm tra kết nối 58](#_Toc531936315)

[Hình 27: Kết nối đến MongoDB server 59](#_Toc531936316)

[Hình 28: Giao diện Robo 3T khi kết nối lần đầu 60](#_Toc531936317)

[Hình 29: Tạo cơ sở dữ liệu 60](#_Toc531936318)

[Hình 30: Đặt tên cho cơ sở dữ liệu 61](#_Toc531936319)

[Hình 31: Tạo cơ sở dữ liệu thành công 61](#_Toc531936320)

[Hình 32: Tạo Collection 62](#_Toc531936321)

[Hình 33: Đặt tên cho Collection 62](#_Toc531936322)

[Hình 34: Insert Document 63](#_Toc531936323)

[Hình 35: Giao diện thực hiện Insert Document 64](#_Toc531936324)

[Hình 36: Xem Document 65](#_Toc531936325)

[Hình 37:Giao diện Document đã tạo thành công 65](#_Toc531936326)

DANH MỤC BẢNG

[Bảng 1: So sánh NoSQL và CSDL quan hệ 17](#_Toc531936259)

[Bảng 2: So sánh giữa CSDL quan hệ với MongoDB 31](#_Toc531936260)

[Bảng 3: Các kiểu dữ liệu trong MongoDB 32](#_Toc531936261)

[Bảng 4: Một số câu lệnh MongoDB thường dùng 35](#_Toc531936262)

DANH MỤC CHỮ VIẾT TẮT

|  |  |  |
| --- | --- | --- |
| STT | Chữ viết tắt | Ý nghĩa |
| 1 | CSDL | Cơ sở dữ liệu |
| 2 | RDBMS | Relational Database Management System: Hệ quản trị cơ sở dữ liệu quan hệ |

PHẦN I. MỞ ĐẦU

1.1. Đặt vấn đề

Ngày nay với kỷ nguyên công nghệ đang bùng nổ, thành công của Internet đã khiến cho số lượng người dùng truy cập vào cùng một hệ thống ngày càng tăng. Điển hình như Facebook, tính đến tháng 1 năm 2018 có 2,2 tỷ người dùng. Để giải quyết vấn đề như trên thì chúng ta đã mở rộng các hệ thống máy chủ siêu lớn, phân thành nhiều cụm đặt tại khắp nơi trên thế giới. Nhưng với tốc độ phát triển nhanh như hiện nay thì việc tăng số lượng máy chủ thôi vẫn chưa đủ. Ta cần xem xét và nâng cấp các giải pháp lưu trữ cho tương lai.

Với một số lượng thông tin lưu trữ khổng lồ thì chúng ta cần phải có hệ thống máy chủ cơ sở dữ liệu rất mạnh mẽ nếu không máy chủ sẽ quá tải. Các hệ thống có số lượng truy cập và lưu trữ lên hàng triệu đến hàng tỉ thì hiệu năng tốt là việc bắt buộc. Các hệ cơ sở dữ liệu quan hệ (RDBMS) hiện nay thì vấn đề hiệu năng thường không tốt trong trường hợp này. RDBMS hiện tại bộc lộ những yếu kém trong tác vụ như đánh chỉ mục một lượng lớn dữ liệu, phân trang, hoặc phân phối luồng dữ liệu media (phim, ảnh nhạc...). Cơ sở dữ liệu quan hệ được thiết kế cho những mô hình dữ liệu không quá lớn trong khi các dịch vụ mạng xã hội lại có một lượng dữ liệu cực lớn và cập nhật liên tục do số lượng người dùng quá nhiều.

Thế hệ cơ sở dữ liệu mới - NoSQL – với hệ hống lưu trữ phân tán và mô hình lưu trữ cặp: key - value. Hệ cơ sở dữ liệu này có thể lưu trữ, xử lý từ lượng nhỏ đến hàng petabytes dữ liệu với khả năng chịu tải, chịu lỗi cao nhưng chỉ đòi hỏi về tài nguyên phần cứng thấp. Vì vậy ở trong khóa luận tốt nghiệp này, em quyết định chọn đề tài: “Tìm hiểu về NoSQL và ứng dụng” với mục đích có thể tìm hiểu rõ ràng hơn về loại cơ sở dữ liệu mới này và đi sâu nghiên cứu về một loại NoSQL phổ biến nhất hiện nay - MongoDB.

1.2. Mục đích và yêu cầu

1.2.1. Mục đích

- Tìm hiểu tổng quan về NoSQL: kiến trúc, đặc điểm, ưu khuyết điểm…; biết cách mà NoSQL giải quyết vấn đề hiệu suất cao với lượng dữ liệu lớn như thế nào; đồng thời biết trường hợp nào có thể sử dụng cơ sở dữ liệu dạng NoSQL.

- Tìm hiểu chi tiết về MongoDB và biết cách ứng dụng MongoDB vào một bài toán cụ thể.

1.2.2. Yêu cầu

- Hiểu biết tổng quan về NoSQL: kiến trúc, phân loại, đặc điểm…

- Hiểu biết về MongoDB và biết sử dụng cơ sở dữ liệu này.

- Từ kiến thức đã tìm hiểu được xây dựng một ứng dụng demo sử dụng hệ quản trị cơ sở dữ liệu MongoDB.

- Đưa ra kết quả và đánh giá.

PHẦN II. TỔNG QUAN TÌNH HÌNH NGHIÊN CỨU TRONG VÀ NGOÀI NƯỚC

2.1. Tình hình nghiên cứu trong nước.

Hiện nay tại Việt Nam, khái niệm cơ sở dữ liệu dạng NoSQL không phải là mới nhưng cũng không quá cũ. Đã có một số đề tài tìm hiểu về NoSQL và một số hệ cơ sở dữ liệu dạng NoSQL (đặc biệt là MongoDB) cũng đang được rất nhiều công ty sử dụng như: SamSung, FPT, Five9, BiTech,…

2.2. Tình hình nghiên cứu ngoài nước

NoSQL đang được sử dụng ngày càng nhiều trong các ứng dụng dữ liệu lớn và ứng dụng nền web thời gian thực.

Dựa trên các bảng xếp hạng phổ biến tháng 12 năm 2018, các cơ sở dữ liệu NoSQL phổ biến nhất là MongoDB xếp thứ 5 và Redis xếp thứ 7[2].

PHẦN III. NỘI DUNG VÀ PHƯƠNG PHÁP NGHIÊN CỨU

3.1. Địa điểm và thời gian nghiên cứu (thực tập)

- Địa điểm nghiên cứu: Khoa Công nghệ thông tin, Học viện Nông nghiệp Việt Nam - Gia Lâm, Hà Nội.

- Thời gian nghiên cứu: Bắt đầu từ 06/08/2018 đến 25/12/2018

3.2. Nội dung nghiên cứu

- Tìm hiểu chung về NoSQL

- Tìm hiểu trường hợp áp dụng cơ sở dữ liệu dạng NoSQL, trường hợp nào không phù hợp với NoSQL.

- Phân tích làm rõ ưu khuyết điểm của việc áp dụng cơ sở dữ liệu NoSQL.

- So sánh giữa việc sử dụng cơ sở dữ liệu RDBMS hoặc XML và cơ sở dữ liệu NoSQL trên cùng một ứng dụng.

- So sánh hiệu suất giữa một cơ sở dữ liệu dạng NoSQL và cơ sở dữ liệu dạng RDBMS để làm rõ hiệu suất hoạt động của NoSQL.

- Tập trung tìm hiểu chi tiết cơ sở dữ liệu loại Document database.

- Sau đó tìm hiểu chi tiết về hệ quản trị cơ sở dữ liệu MongoDB.

- Xây dựng một chương trình demo sử dụng NoSQL, MongoDB.

3.3. Phương pháp nghiên cứu

- Phương pháp nghiên cứu tài liệu và cài đặt thử nghiệm.

PHẦN IV. KẾT QUẢ VÀ THẢO LUẬN

4.1. Tổng quan về NoSQL

4.1.1. NoSQL là gì?



Hình 1: Cơ sở dữ liệu NoSQL

NoSQL là 1 dạng cơ sở dữ liệu nguồn mở không sử dụng câu lệnh SQL để truy vấn thông tin. NoSQL viết tắt bởi: None - Relational SQL, hay có nơi thường gọi là Not - Only SQL.

NoSQL được phát triển trên JavaScript Framework với kiểu dữ liệu là JSON và dạng dữ liệu theo kiểu key và value (1 đặc trưng về dữ liệu trong JSON). NoSQL ra đời như là 1 mảnh vá cho những khuyết điểm và thiếu sót cũng như hạn chế của mô hình dữ liệu quan hệ RDBMS về tốc độ, tính năng, khả năng mở rộng, memory cache,…

Khi sử dụng một dạng CSDL quan hệ nào đó như: SQL server, MySQL,...bạn có thể mất không ít công sức trong việc chỉnh sửa các bảng dữ liệu khi liên quan đến khóa chính và khóa ngoại hay một loạt các rắc rối khác trong quá trình làm việc. Nhưng với NoSQL bạn có thể mở rộng dữ liệu mà không cần lo đến việc tạo khóa chính, khóa ngoại, kiểm tra ràng buộc,...Vì NoSQL không hạn chế việc mở rộng dữ liệu nên tồn tại nhiều nhược điểm như: sự phụ thuộc của từng bản ghi, tính nhất quán, toàn vẹn dữ liệu,....nhưng chúng ta có thể chấp nhận những nhược điểm đó để khiến ứng dụng cải thiện hiệu suất cao hơn khi giải quyết những bài toán lớn về hệ thống thông tin, phân tán hay lưu trữ dữ liệu.

4.1.2. Đặc điểm

- NoSQL lưu trữ dữ liệu của mình theo dạng cặp giá trị “key – value”. Sử dụng số lượng lớn các node để lưu trữ thông tin – Mô hình phân tán dưới sự kiểm soát phần mềm.

- Chấp nhận dữ liệu bị trùng lặp do một số node sẽ lưu cùng thông tin giống nhau.

- Một truy vấn sẽ được gửi tới nhiều máy cùng lúc, do đó khi một máy nào đó không phục vụ được sẽ không ảnh hưởng lắm đến chất lượng trả về kết quả.

- Phi quan hệ – không có ràng buộc nào cho việc nhất quán dữ liệu.

- Tính nhất quán không theo thời gian thực: Sau mỗi thay đổi CSDL, không cần tác động ngay đến tất cả các CSDL liên quan mà được lan truyền theo thời gian.

4.1.3. Một số khái niệm mới trong NoSQL

**- Fields**: Tương đương với khái niệm Columns trong SQL.

**- Document**: Thay thế khái niệm row trong SQL. Đây cũng chính là khái niệm làm nên sự khác biệt giữa NoSQL và SQL, 1 document chứa số cột (fields) không cố định trong khi 1 row thì số cột (columns) là định sẵn trước.

**- Collection**: Tương đương với khái niệm table trong SQL. Một collection là tập hợp các document. Điều đặc biệt là một collection có thể chứa các document hoàn toàn khác nhau.

- **\_id:** Là trường bắt buộc có trong mỗi document. Trường \_id đại diện cho một giá trị duy nhất trong docume. Trường \_id cũng có thể được hiểu là khóa chính trong document. Nếu bạn thêm mới một document thì CSDL sẽ tự động sinh ra một \_id đại diện cho document đó và là duy nhất trong cơ sở dữ liệu đó.

**- Key – value**: Cặp từ khóa – giá trị được dùng để lưu trữ dữ liệu trong NoSQL.

**- Cursor** (con trỏ): Chúng ta sẽ sử dụng cursor để lấy dữ liệu từ database.

**- Index**: Trong các hệ cơ sở dữ liệu quan hệ, các cột được định nghĩa theo bảng còn với hệ cơ sở dữ liệu không ràng buộc thì các cột được định nghĩa ở mỗi document. Bởi thế, các document quản lí gần như tất cả, các collection không cần quản lí chặt chẽ những gì đang xảy ra trong nó nữa .

4.1.4. Một số thuật ngữ liên quan

**- Non - relational**: relational (ràng buộc) - thuật ngữ sử dụng chỉ đến các mối quan hệ giữa các bảng trong cơ sở dữ liệu quan hệ sử dụng mô hình khóa gồm 2 loại khóa: khóa chính và khóa phụ (primary key + foreign key) để ràng buộc dữ liệu nhằm thể hiện tính nhất quán dữ liệu từ các bảng khác nhau. Non - relational là khái niệm không sử dụng các ràng buộc dữ liệu cho nhất quán dữ liệu ở NoSQL.

**- Distributed storage**: mô hình lưu trữ phân tán các file hoặc dữ liệu ra nhiều máy tính khác nhau trong mạng LAN hoặc Internet dưới sự kiểm soát của phần mềm.

**- Eventual consistency** (nhất quán cuối): tính nhất quán của dữ liệu không cần phải đảm bảo ngay tức khắc sau mỗi phép write. Một hệ thống phân tán chấp nhận những ảnh hưởng theo phương thức lan truyền và sau một khoảng thời gian (không phải ngay tức khắc), thay đổi sẽ đi đến mọi điểm trong hệ thống, tức là cuối cùng (eventually) dữ liệu trên hệ thống sẽ trở lại trạng thái nhất quán.

**- Vertical scalable** (khả năng mở rộng chiều dọc): Khi dữ liệu lớn về lượng, phương pháp tăng cường khả năng lưu trữ và xử lý bằng việc cải tiến phần mềm và cải thiện phần cứng trên một máy tính đơn lẻ được gọi là khả năng mở rộng chiều dọc. Ví dụ việc tăng cường CPUs, cải thiện đĩa cứng, bộ nhớ trong một máy tính... cho RDBMS nằm trong phạm trù này. Khả năng mở rộng chiều dọc còn có một thuật ngữ khác scale up.

**- Horizontal scalable** (khả năng mở rộng chiều ngang):

+ Khi dữ liệu lớn về lượng, phương pháp tăng cường khả năng lưu trữ và xử lý là dùng nhiều máy tính phân tán. Phân tán dữ liệu được hỗ trợ bởi phần mềm tức hệ quản trị cơ sở dữ liệu.

+ Trong khi giá thành phần cứng ngày càng giảm, tốc độ xử lý, bộ nhớ ngày càng tăng thì horizontal scalable là một lựa chọn đúng đắn. Hàng trăm máy tính nhỏ được chập lại tạo thành một hệ thống tính toán mạnh hơn nhiều so với vi xử lý RISC truyền thống đơn lẻ. Mô hình này tiếp tục được hỗ trợ bởi các công nghệ kết nối Myrinet và InfiniBand. Từ đó chúng ta có thể quản lý, bảo trì từ xa, xây dựng batch procession (xử lý đồng loạt tập lệnh) tốt hơn. Do những đòi hỏi về tốc độ xử lý I/O cao, lượng cực lớn dữ liệu,...scale horizontally sẽ thúc đẩy các công nghệ lưu trữ mới phát triển giống như object storage devices (OSD).

4.1.5. Ưu – nhược điểm của NoSQL

**4.1.5.1. Ưu điểm**

- NoSQL là nguồn mở: hầu hết các sản phẩm nguồn mở đưa ra cho những người phát triển với nhiều lợi ích to lớn, đặc biệt là việc sử dụng miễn phí.

- Khả năng mở rộng linh hoạt: do không bị ràng buộc chặt về các mối quan hệ, cấu trúc lưu trữ nên khả năng mở rông của NoSQL rất linh hoạt.

- Các CSDL NoSQL khác nhau cho các dự án khác nhau: mỗi loại CSDL dạng NoSQL sẽ là giải pháp phục vụ cho một hoặc một vài vấn đề cụ thể.

- NoSQL được các công ty lớn sử dụng như: Facebook, Amazon, BBC, Google đều có sử dụng CSDL dạng NoSQL.

- NoSQL phù hợp với công nghệ đám mây: những yêu cầu về lưu trữ của công nghệ đám mây với NoSQL là một sự trùng khớp tự nhiên.

**4.1.5.2. Nhược điểm**

- Hỗ trợ không đồng đều cho các doanh nghiệp: trong khi các nhà cung cấp chủ chốt của RDBMS như: SQL server, Oracle, IBM,… thường đưa ra sự hỗ trợ tốt cho khách hàng thì các nhà cung cấp nguồn mở mới thành lập không thể mong đợi sẽ cung cấp được sự hỗ trợ có thể so sánh được.

- Chưa đủ “chín” cho các doanh nghiệp: dù NoSQL đã được triển khai tại một số công ty lớn thì chúng vẫn phải đối mặt với một vấn đề về sự tin cậy chính với nhiều doanh nghiệp. Vấn đề lớn của NoSQL là thiếu về độ chín muồi và các vấn đề về tính không ổn định, trong khi đó tính chín muồi, hỗ trợ đầy đủ các chức năng và tính ổn định của các RDBMS đã được thiết lập từ lâu.

- Hạn chế về tri thức nghiệp vụ: các CSDL NoSQL không có nhiều sự đeo bám tới các công cụ BI (tri thức nghiệp vụ) thường được sử dụng, trong khi những yêu cầu và phân tích hiên đại đơn giản nhất thì cũng liên quan khá nhiều tới sự linh thôngvề lập trình.

- Thiếu sự tinh thông: tính mới mẻ của NoSQL là không có nhiều lập trình viên và người quản trị biết về công nghệ này. Như vậy sẽ rất khó khăn cho các công ty tìm người có hiểu biết phù hợp.

- Vấn đề về tính tương thích: mỗi CSDL NoSQL có các giao diện lập trình ứng dụng API riêng, các giao diện truy vấn riêng... Sự thiếu hụt các tiêu chuẩn sẽ gây ra rất nhiều khó khăn cho việc chuyển từ nhà cung cấp này sang một nhà cung cấp khác khi cớ nhu cầu.

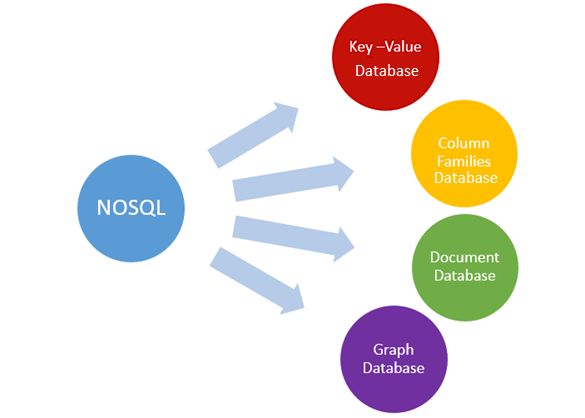
4.1.6. So sánh NoSQL và cơ sở dữ liệu quan hệ

Bảng 1: So sánh NoSQL và CSDL quan hệ

|  |  |  |
| --- | --- | --- |
| Đặc điểm | CSDL quan hệ | CSDL NoSQL |
| Hiệu suất | Kém hơn  SQL Relational giữa các table | Cực tốt Bỏ qua SQL Bỏ qua các ràng buộc dữ liệu |
| Khả năng mở rộng | Hạn chế về lượng | Hỗ trợ một lượng lớn các node |
| Hiệu suất đọc - ghi | Kém do thiết kế để đảm bảo sự vào/ra liên tục của dữ liệu | Tốt với mô hình xử lý lô và những tối ưu về đọc-ghi dữ liệu. |
| Thay đổi số node trong hệ thống | Phải shutdown cả hệ thống. Việc thay đổi số node phức tạp. | Không cần phải shutdown cả hệ thống. Việc thay đổi số node đơn giản, không ảnh hưởng đến hệ thống. |
| Phần cứng | Đòi hỏi cao về phần cứng. | Đòi hỏi thấp hơn về giá trị và tính đồng nhất của phần cứng |

4.1.7. Các dạng NoSQL cơ bản

Cơ sở dữ liệu NoSQL được phân loại theo cách mà nó lưu trữ dữ liệu và gồm có 4 loại chính:

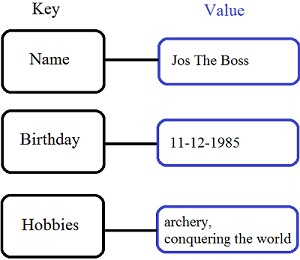


Hình 2: Phân loại NoSQL

**4.1.7.1. Key - value Database**

Dữ liệu lưu trữ trong database dưới dạng cặp key – value. Để truy vấn dữ liệu trong database, ta dựa vào key để lấy value ra. Các database dạng này có tốc độ truy vấn rất nhanh.

* Ví dụ hệ quản trị cơ sở đặc trưng: Riak, Redis, MemCache, Project Voldemort, CouchBase
* Ưu điểm: Tìm kiếm rất nhanh
* Nhược điểm: Lưu dữ liệu không theo khuôn dạng (schema) nhất định
* Ứng dụng: Do tốc độ truy xuất nhanh, key - value database thường được dùng để làm cache cho ứng dụng (Tiêu biểu là Redis và MemCache). Ngoài ra, nó còn được dùng để lưu thông tin trong sessions, profiles/preferences của user…



Hình 3: Key - values Database

**4.1.7.2. Column - Family Database**

Dữ liệu được lưu trong database dưới dạng các cột, thay vì các hàng như SQL. Mỗi hàng sẽ có một key/id riêng. Điểm đặt biệt là các hàng trong một bảng sẽ có số lượng cột khác nhau. Câu lệnh truy vấn của nó khá giống SQL.

* Ví dụ hệ quản trị cơ sở đặc trưng: Cassandra (Phát triển bởi Facebook), HyperTable, Apache HBase
* Ưu điểm: Tìm kiếm nhanh, Phân tán dữ liệu tốt
* Nhược điểm: Hỗ trợ được với rất ít phần mềm
* Ứng dụng: Column-Family Database được sử dụng khi ta cần ghi một số lượng lớn dữ liệu, big data. Nó còn được ứng dụng trong 1 số CMS và ứng dụng e-commerce.

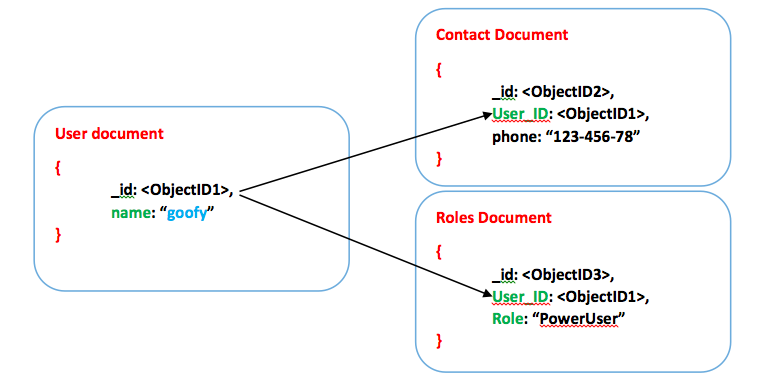


Hình 4: Column - family Databse

**4.1.7.3. Document Database**

Mỗi object sẽ được lưu trữ trong database dưới dạng một document. Dữ liệu sẽ được lưu trữ dưới dạng BSON/JSON/XML dưới database. Dữ liệu không schema cứng như SQL, do đó ta có thể thêm/sửa field, thay đổi table, … rất nhanh và đơn giản. Database dạng này có tốc độ truy vấn nhanh, có thể thực hiện các câu truy vấn phức tạp, dễ mở rộng (scalability). Mỗi database có một kiểu truy vấn riêng (RavenDB dùng Lucene, MongoDB lại dùng query document).

* Ví dụ hệ quản trị cơ sở đặc trưng: MongoDB, RavenDB, CouchDB, TerraStone, OrientDB
* Ưu điểm: Dùng khi dữ liệu nguồn không được mô tả đầy đủ
* Nhược điểm: Hiệu năng truy vấn, Không có cú pháp chuẩn cho câu truy vấn dữ liệu
* Ứng dụng: Do nhanh và linh động, document database thường đóng vài trò làm database cho các ứng dụng prototype, big data, e-commerce, CMS. Ngoài ra, ta còn dùng nó để lưu log hoặc history.

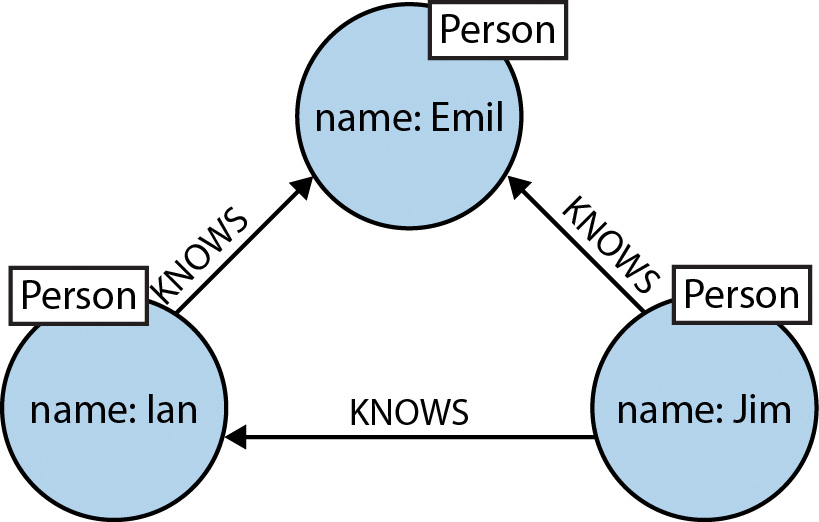


Hình 5: Document Database

**4.1.7.4. Graph Database**

Những CSDL này áp dụng lý thuyết đồ thị trong khoa học máy tính để lưu trữ và truy xuất dữ liệu. Chúng tập trung vào tính rời rạc giữa các phần dữ liệu. Các phần tử đơn vị dữ liệu được biểu thị như một nút và liên kết với các thành phần khác bằng các cạnh.

* Ví dụ hệ quản trị cơ sở đặc trưng: Neo4j, InfiniteGraph, OrientDB, HYPERGRAPHDB
* Ưu điểm: Ứng dụng các thuật toán trên đồ thị như đường đi ngắn nhất, liên thông,…
* Nhược điểm: Phải duyệt nội bộ đồ thị, để trả lời lại các truy vấn. Không dễ để phân tán
* Ứng dụng: Khi cần truy vấn các mối quan hệ, graph database truy vấn nhanh và dễ hơn nhiều so với database. Nó được dùng trong các hệ thống: mạng nơ ron, chuyển tiền bạc, mạng xã hội (tìm bạn bè), giới thiệu sản phẩm (dựa theo sở thích/lịch sử mua sắm của người dùng)…



Hình 6: Graph Database

4.1.8. Làm sao để chọn một giải pháp cơ sở dữ liệu tốt?

Mỗi loại cơ sở dữ liệu dạng NoSQL đều có những điểm mạnh và điểm yếu riêng của nó. Một câu hỏi chúng ta thường hay gặp là: “Tôi muốn sử dụng công nghệ NoSQL X cho việc Y thì làm sao?”. Với câu hỏi này, chúng ta thường gặp phải vấn đề là:

- Cố gắng áp dụng các khái niệm, kỹ thuật, kinh nghiệm của mô hình cơ sở dữ liệu quan hệ truyền thống vào trong NOSQL.

- Cố gắng sử dụng một loại cơ sở dữ liệu NoSQL trên toàn bộ ứng dụng mà có thể có những phần khác nhau của ứng dụng không phù hợp với cơ sở dữ liệu NoSQL này.

Trong một ứng dụng, chúng ta có thể sử dụng key-value database để lưu trữ thông tin phiên làm việc (session), sử dụng graph database để phục vụ những truy vấn xã hội và document database để lưu trữ các thực thể. Nếu chúng ta lưu trữ dữ liệu theo một loại cơ sở dữ liệu NoSQl duy nhất thì việc này giống như chúng ta muốn lưu trữ tất cả code trên một file duy nhất. Chúng ta có thể làm được việc này nhưng có vẻ vụng về, không được tối ưu lắm. Điều nên làm là cố gắng phân chia ứng dụng thành từng phần mà mỗi phần thích hợp với một mô hình truy cập dữ liệu để đem lại hiệu quả cao nhất.

Kết luận: Trong một ứng dụng chúng ta có thể sử dụng nhiều công nghệ lưu trữ dữ liệu khác nhau để làm cho ứng dụng của chúng ta hoạt động tốt nhất và mỗi phần khác nhau của ứng có thể sử dụng công nghệ khác nhau sao cho phù hợp với mục đích của chúng ta. Điều đó cũng nói lên rằng: trong hệ thống sử dụng nhiều công nghệ lưu trữ, một công nghệ lưu trữ dữ liệu mới chỉ thực sự có nghĩa khi mà lợi ích nó mang lại lớn hơn chi phí phải trả để sử dụng công nghệ đó. Nếu chúng ta cần hỗ trợ lưu trữ các trường dữ liệu người dùng tự định nghĩa thì chúng ta nhanh chóng sử dụng document database hơn là cố gắng thực hiện điều đó với RDBMS.

Trong khóa luận tốt nghiệp này, em chọn Document Database làm cơ sở dữ liệu NoSQL để xây dựng ứng dụng chính vì:

- Về cơ bản thì cốt lõi của Document Database là key-value database được lưu trữ theo một định dạng được biết đến. Do đó, document database cũng đáp ứng được yêu cầu của một key-value database khi cần truy cập dữ liệu theo khóa.

- Dữ liệu trong document database được lưu trữ dưới định dạng mà cơ sở dữ liệu hiểu được. Các định dạng có thể là XML, JSON, Binary JSON(BSON) miễn sao cơ sở dữ liệu hiểu được cấu trúc nội bộ của document. Thực tế thì hầu hết các ứng đều sử dụng JSON (hoặc BSON) hoặc XML. Đây đều là những định dạng được sử dụng rất phổ biến và con người có thể đọc được.

- Cơ sở dữ liệu hiểu được định dạng của dữ liệu thì nó có thể thực hiện thao tác trên dữ liệu này phía máy chủ và dễ dàng hơn để viết các công cụ quản lý dữ liệu vì có thể hiển thị và chỉnh sửa dữ liệu.

- Document database có lược đồ tùy ý. Chúng ta không cần phải định nghĩa trước lược đồ và tuân thủ theo lược đồ này. Điều này cho phép chúng ta lưu trữ dữ liệu phức tạp tùy ý. Có thể lưu trữ dữ liệu dạng cây, tập hợp hay dạng từ điển một cách dễ dàng.

- Lợi ích chính của việc sử dụng document database là ngoài việc nó có tất cả lợi ích của key-value database thì chúng ta không bị giới hạn bởi việc truy vấn theo khóa.

- Bên cạnh việc có thể truy vấn dữ liệu, document database còn có thể:

+ Thực hiện phép chiếu dữ liệu của một document sang một định dạng khác.

+ Chạy phép tính tập hợp trên một tập hợp các document.

+ Cập nhật một phần dữ liệu

- Lợi ích quan trọng của việc sử dụng document database là làm việc với các documents.

- Document không hỗ trợ mối quan hệ. Mỗi document là độc lập và chúng ta sẽ dễ dàng phân tán dữ liệu hơn so với RDBMS bởi vì chúng ta không cần lưu trữ tất cả các quan hệ trên cùng một mảnh của hệ thống và không cần hỗ trợ phép join trên hệ thống phân tán.

4.1.9. Tìm hiểu một số Tool NoSQL phổ biến

**4.1.9.1. RavenDB**

RavenDB được viết trên C# bởi Hibernating Rhinos với giấy phép GNU AGPL v3.0. RavenDB là một giải pháp NoSQL trên nền tảng .NET được xây dựng dựa trên kiến trúc client-server. Dữ liệu được lưu trữ trên một thực thể máy chủ và những yêu cầu dữ liệu có thể được gửi tới máy chủ này từ một hoặc nhiều máy người dùng khác nhau.

Những yêu cầu gửi tới máy chủ được thực hiện bằng cách sử dụng những Client API có sẵn trong bất kỳ ứng dụng .NET hoặc ứng dụng SilverLight, hoặc bằng cách truy cập trực tiếp tới Server’s RESTful API. Nếu là một .NET developer thì sử dụng .NET Client API là cách dễ nhất để làm việc với RavenDB vì nó cung cấp một lượng lớn các tính năng và nhiều API hỗ trợ. RESTful API làm cho RavenDB có thể được truy cập từ nhiều nền tảng khác nhau như truy vấn AJAX trong trang web hoặc là các ứng dụng Non-Windows được viết bằng Ruby-on-Rail.

Các đặc điểm chính của RavenDB:

- Mặc định an toàn dữ liệu: Hỗ trợ ACID (Atomicity, Consistency, Isolation, Durability), No locking, Automatic batching, client/server chatter projection.

- .Net client API: hỗ trợ tốt cho việc lập trình trên nền tảng .NET

- REST API: Tất cả dữ liệu đều có một địa chỉ duy nhất được lấy qua HTTP. Giao thức REST sử dụng các phương thức của HTTP như GET, POST, PUT và DELETE.

- Dễ dàng triển khai ứng dụng một cách nhanh chóng.

- Kiến trúc phân tán: mở rộng ứng dụng một cách dễ dàng bằng cách sử dụng tính năng mạnh mẽ của RavenDB cho việc mở rộng là Sharding và Replication. Có thể kết hợp cả hai tính năng này trong cùng ứng dụng. Có hỗ trợ multi-database.

- Hỗ trợ nhiều gói tiện ích hữu dụng như: Versioning, Expiration, IndexReplication, Authorization, Authentication. Chúng ta có thể tự viết các gói mở rộng cho RavenDB bằng cách sử dụng Triggers và Responders.

**4.1.9.2. Hadoop**

Hadoop là một framework nguồn mở viết bằng Java cho phép phát triển các ứng dụng phân tán có cường độ dữ liệu lớn một cách miễn phí. Nó cho phép các ứng dụng có thể làm việc với hàng ngàn node khác nhau và hàng petabyte dữ liệu. Hadoop lấy được phát triển dựa trên ý tưởng từ các công bố của Google về mô hình MapReduce và hệ thống file phân tán Google File System (GFS). Map/Reduce là mô hình mà ứng dụng sẽ được chia nhỏ ra thành nhiều phân đoạn khác nhau, và các phần này sẽ được chạy song song trên nhiều node khác nhau. Thêm vào đó, Hadoop cung cấp 1 hệ thống file phân tán (HDFS) cho phép lưu trữ dữ liệu lên trên nhiều node. Cả Map/Reduce và HDFS đều được thiết kế sao cho framework sẽ tự động quản lý được các lỗi, các hư hỏng về phần cứng của các node. Hadoop giúp các nhà phát triển ứng dụng phân tán tập trung tối đa vào phần logic của ứng dụng, bỏ qua được một số phần chi tiết kỹ thuật phân tán bên dưới (phần này do Hadoop tự động quản lý).

**4.1.9.3. CouchDB**

CouchDB được viết bằng Erlang với mục tiêu là tạo ra một cơ sở dữ liệu bền vững, chịu lỗi cao, dễ dàng trong việc sử dụng. Dùng cách lưu trữ thông thường là JSON với giấy phép Apache 2.0. Với CouchDB thì mỗi một cơ sở dữ liệu là một tập các văn bản riêng biệt. Mỗi văn bản tự bảo quản chính nó và tự nó bao gồm mô hình của nó (các trường, loại của mỗi trường). Mỗi một ứng dụng có thể thực thi rất nhiều cơ sở dữ liệu, ví dụ như chúng ta dùng một cơ sở dữ liệu để lưu thông tin người dùng điện thoại và cái còn lại là lưu trên server. Trên mỗi văn bản(bản ghi) còn bao gồm các thông tin về phiên bản, khiến cho việc dễ dàng đồng bộ các dữ liêu với nhau khi cơ sở dữ liệu bị mất kết nối một thời gian giữa các thiết bị.

CouchDB dử dụng MVCC (multi-Version Concurency Control ) để tránh việc deadlock cơ sở dữ liệu trong suốt quá trình ghi. Tức là trong khi ghi dữ liệu, chúng ta vẫn có thể đọc dữ liệu vì CouchDB sinh ra một bản copy và chúng ta đọc trên bản copy đó. Sau khi ghi xong nó sẽ tiến hành nhập dữ liệu giữa các thiết bị và xóa bản ghi cũ đi. Dùng giao thức HTTP theo RESTful với cách thiết kế có khả năng chịu lỗi cao với việc dùng views đi kèm với map/reduce mang lại một tốc độ cao. Thích hợp cho rất nhiều các thiết bị khác nhau như máy chủ, máy bàn hay điện thoại thông minh.

CouchDB được sử dụng tốt nhất cho các hệ thống thỉnh thoảng thay đổi dữ liệu như các hệ thống CMS, các hệ thống cho phép triển khai nhiều trang web.

Các đặc điểm chính của CouchDB:

- Lưu trữ theo hướng văn bản (document storage) .

- Sử dụng ngữ nghĩa ACID: Cho phép điều khiển việc đồng bộ việc ghi và đọc cường độ rất cao mà không lo bị xung đột.

- Sử dụng Map/Reduce và các chỉ mục: Mỗi view được tạo ra bởi một hàm javascript mà thực thi cả 2 hành động map và reduce. Hàm đó làm cho các văn bản kết hợp với nhau thành một giá trị đơn nhất và trả về kết quả đó.

- Kiến trúc phân tán có nhân bản: CouchDB được thiết kế với khả năng nhân bản 2 chiều với các dữ liệu offline. Tức là ta có thể chỉnh sửa dữ liệu offline và sau đó đồng bộ chúng sau khi có kết nối trở lại.

- REST API: Tất cả dữ liệu đều có một địa chỉ duy nhất được lấy qua HTTP. Giao thức REST sử dụng các phương thức của HTTP như GET, POST, PUT và DELETE với 4 chức năng cơ bản (Tạo, đọc, ghi, xóa, sửa)

- Built for Offline: Có khả năng nhân bản dữ liệu cho từng thiết bị và tự động đồng bộ dữ liệu khi th

**4.1.9.4. Cassandra**

Cassandra là một hệ quản trị cơ sở dữ liệu nguồn mở, được viết bằng Java với mục tiêu chính là trở thành Best of BigTable. Cassandra được thiết kế với khả năng xử lý một khối dữ liệu cực lớn được trải ra trên rất nhiều máy chủ trong khi cung cấp một dịch vụ có tính sẵn sàng cao và không hỏng. Nó là một giải pháp NoSQL bước đầu được phát triển bởi Facebook.

Cassandra cung cấp một cấu trúc lưu trữ theo dạng key/value với khả năng điều hướng tính nhất quán. Các khóa ánh xạ đến nhiều giá trị, cái mà được gộp thành các nhóm cột. Các nhóm cột được cố định khi cơ sở dữ liệu Cassandra được tạo ra, nhưng các cột có thể được thêm vào nhóm đó bất cứ lúc nào. Hơn nữa, các cột được thêm vào chỉ để làm các khóa xác định, bởi vậy các khóa khác nhau có thể có số lượng cột khác nhau. Giá trị từ các nhóm cột cho mỗi một khóa được lưu trữ cùng nhau. Điều đó khiến cho Cassandra là một hệ quản trị dữ liệu lai giữa hướng cột hoặc là hướng bản ghi.

Cassandra được dùng tốt nhất khi bạn ghi nhiều hơn bạn đọc, ví dụ ở đây là hệ thống logging nhiều như các mạng xã hội, hệ thống ngân hàng, tài chính chứng khoán. Với tốc độ ghi nhanh hơn tốc độ đọc, nó thích hợp cho việc phân tích dữ liệu thời gian thực.

Các đặc điểm nổi bật:

- Tính phân cấp: Mỗi node trong một cụm có cùng một luật. Dữ liệu được phân tán dọc theo các cụm đó (do đó mỗi node lại có một dữ liệu khác nhau), nhưng không có master bởi mỗi một node có thể phục vụ bất kì một yêu cầu nào.

- Hỗ trợ nhân bản và nhân bản nhiều trung tâm dữ liệu: Việc mô phỏng có thể được cấu hình. Cassandra được thiết kế cho các hệ thống phân tán, có thể triển khai một số lượng lớn các node trên nhiều trung tâm dữ liệu khác nhau. Kiến trúc phân phối các đặc trưng khóa của Casandra thích hợp cho việc triển khai nhiều tập dữ liệu. Xử lý dữ liệu dư thừa, đề phòng việc hỏng hóc.

- Tính đàn hồi: Thông lượng đọc và ghi đều tăng tuyến tính khi các máy mới thêm vào vì giảm được thời gian chết hoặc bị gián đoạn giữa các ứng dụng

- Tính dung lỗi: Dữ liệu được nhân bản ra thành nhiều node cho khả năng dung lỗi. Việc nhân bản giữa các trung tâm dữ liệu khác nhau cũng được hỗ trợ. Các node lỗi có thể được thay thế mà không mất thời gian chờ đợi.

- Tính điều hướng nhất quán: Đọc và ghi đưa ra một yêu cầu về tính nhất quán với việc "việc ghi không bao giờ bị lỗi".

- Hỗ trợ Map/Reduce: Cassandra có tích hợp thêm cả Hadoop đồng nghĩa với việc hỗ trợ map/reduce.

- Có truy vấn theo ngôn ngữ riêng: CQL (viết tắt của Cassandra Query Language) là một thay thể của SQL – giống với các giao thức RPC truyền thống. Nó được điều khiển bởi Java và Python .

4.2. MongoDB

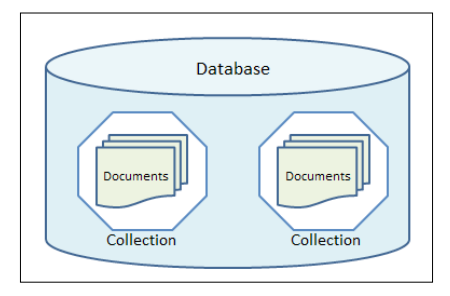
4.2.1. MongoDB là gì?

MongoDB là một cơ sở dữ liệu nguồn mở và là cơ sở dữ liệu dạng NoSQL hàng đầu, được hàng triệu người sử dụng. MongoDB được viết bằng C++.

Ngoài ra, MongoDB là một cơ sở dữ liệu đa nền tảng, hoạt động trên các khái niệm Collection và Document, nó cung cấp hiệu suất cao, tính khả dụng cao và khả năng mở rộng dễ dàng.

MongoDB lưu trữ dữ liệu theo hướng tài liệu (document), các dữ liệu được lưu trữ trong document kiểu JSON nên truy vấn sẽ rất nhanh.

4.2.2. Các thành phần chính của MongoDB



Hình 7: Thành phần chính của MongoDB

- **Database** – Nơi chứa các Collection, giống với cơ sở dữ liệu RDMS chúng chứa các bảng. Mỗi Database có một tập tin riêng lưu trữ trên bộ nhớ vật lý. Một mấy chủ MongoDB có thể chứa nhiều Database.

- **Collection** – Là nhóm của nhiều document trong MongoDB. Collection có thể được hiểu là một bảng tương ứng trong cơ sở dữ liệu RDBMS (Relational Database Management System). Collection nằm trong một cơ sở dữ liệu duy nhất. Các collection không phải định nghĩa các cột, các hàng hay kiểu dữ liệu trước.

- **Document** – Một bản ghi thuộc một Collection thì được gọi là một Document. Các Document lần lượt bao gồm các trường tên và giá trị. Ví dụ về một Document:

|  |
| --- |
| *{*  *"name" : "John Smith",*  *"phone" : {*  *"home" : "555-123-4567",*  *"cell" : "555-321-7654”*  *},*  *"address" : {*  *"street" : "123 Main Street",*  *"city" : "San Francisco",*  *"state" : "CA",*  *"zip" : 94115*  *}*  *}* |

4.2.3. So sánh giữa CSDL quan hệ và MongoDB

Bảng 2: So sánh giữa CSDL quan hệ với MongoDB

|  |  |
| --- | --- |
| CSDL quan hệ | MongoDB |
| Database | Database |
| Table | Collection |
| Tuple/Row | Document |
| Column | Field |
| Table Join | Embedded Documents |
| Primary Key | Primary Key (mặc định là \_id) |

4.2.4. Các kiểu dữ liệu trong MongoDB

MongoDB hỗ trợ các kiểu dữ liệu:

Bảng 3: Các kiểu dữ liệu trong MongoDB

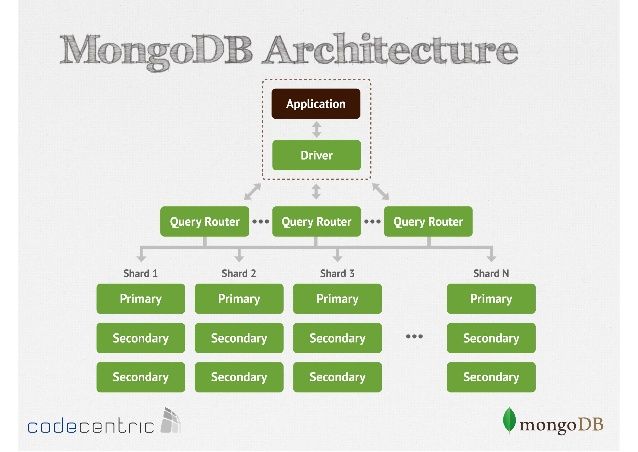
|  |  |  |  |
| --- | --- | --- | --- |
| Type | Number | Alias | Description |
| Double | 1 | Double | Kiểu dữ liệu này được sử dụng để lưu các giá trị số thực dấu chấm động. |
| String | 2 | String | Đây là kiểu dữ liệu được sử dụng phổ biến nhất để lưu trữ dữ liệu. Chuỗi trong MongoDB phải là UTF-8 hợp lệ. |
| Object | 3 | Object | Kiểu dữ liệu này được sử dụng cho các Document được nhúng vào. |
| Array | 4 | Array | Kiểu dữ liệu này được sử dụng để lưu giữ các mảng hoặc danh sách hoặc nhiều giá trị vào trong một key. |
| Binary data | 5 | binData | Kiểu dữ liệu này được sử dụng để lưu trữ dữ liệu nhị phân |
| Undefined | 6 | Undefined | Kiểu dữ liệu này được sử dụng để lưu trữ các giá trị không xác định. |
| ObjectId | 7 | objectId | Kiểu dữ liệu này được sử dụng để lưu trữ ID của Document. |
| Boolean | 8 | Bool | Kiểu dữ liệu này được sử dụng để lưu trữ một giá trị Boolean (true/false). |
| Date | 9 | Date | Kiểu dữ liệu này được sử dụng để lưu trữ date và time hiện tại trong định dạng UNIX time. Bạn có thể xác định date time riêng cho bạn bằng việc tạo đối tượng Date và truyền ngày, tháng, năm vào trong đó. |
| Null | 10 | Null | Kiểu dữ liệu này được sử dụng để lưu một giá trị Null. |
| Regular Expression | 11 | Regex | Kiểu dữ liệu này được sử dụng để lưu trữ Regular Expresion. |
| DBPointer | 12 | dbPointer | Kiểu dữ liệu này được sử dụng để lưu trữ |
| JavaScript | 13 | Javascript | Kiểu dữ liệu này được sử dụng để lưu trữ dữ liệu JavaScript mà không có phạm vi. |
| Symbol | 14 | Symbol | Kiểu dữ liệu này được sử dụng giống như một chuỗi, tuy nhiên, nói chung nó được dành riêng cho các ngôn ngữ mà sử dụng kiểu symbol cụ thể. |
| JavaScript (with scope) | 15 | javascriptWithScope | Kiểu dữ liệu này được sử dụng để lưu trữ dữ liệu JavaScript với một phạm vi nhất định. |
| 32-bit integer | 16 | Int | Kiểu dữ liệu này được sử dụng để lưu một giá trị số nguyên 32 bit. |
| Timestamp | 17 | Timestamp | Giúp thuận tiện cho việc ghi chép hoặc đánh dấu thời điểm một Document được sửa đổi hoặc được thêm vào. |
| 64-bit integer | 18 | Long | Kiểu dữ liệu này được sử dụng để lưu một giá trị số nguyên 64 bit. |
| Decimal128 | 19 | Decimal | Kiểu dữ liệu này được sử dụng để lưu một giá trị số nguyên 128 bit. |
| Min key | -1 | minKey | Kiểu dữ liệu này được sử dụng để so sánh một giá trị với các phần tử BSON thấp nhất. |
| Max key | 127 | maxKey | Kiểu dữ liệu này được sử dụng để so sánh một giá trị với các phần tử BSON cao nhất. |

4.2.5. Một số câu lệnh thường dùng

Bảng 4: Một số câu lệnh MongoDB thường dùng

|  |  |  |
| --- | --- | --- |
| Câu lệnh | SQL | MongoDB |
| Create table | CREATE TABLE people (id MEDIUMINT NOT NULL AUTO\_INCREMENT, user\_id Varchar(30), age Number, status char(1), PRIMARY KEY (id)) | db.people.insertOne({User\_id: “abc123”, Age: 55, Status: “A”}) |
| Drop table | DROP TABLE people | db.people.drop() |
| Insert records into tables | INSERT INTO people(user\_id, age, status) VALUES ("bcd001", 45, "A") | db.people.insertOne( { user\_id: "bcd001", age: 45, status: "A" }) |
| Select | SELECT \*FROM people | db.people.find() |
|  | SELECT id,user\_id, status FROM people | db.people.find( { }, { user\_id: 1, status: 1 } ) |
|  | SELECT \* FROM people WHERE status = "A" | db.people.find( { status: "A" } ) |
|  | SELECT \* FROM people WHERE status = "A" AND age = 50 | db.people.find( { status: "A", age: 50 } ) |
|  | SELECT \* FROM people WHERE status = "A" OR age = 50 | db.people.find( { $or: [ { status: "A" } , { age: 50 } ] } ) |
|  | SELECT \* FROM people WHERE user\_id like "%bc%" | db.people.find( { user\_id: /bc/ } )  db.people.find( { user\_id: { $regex: /bc/ } } ) |
|  | SELECT COUNT(user\_id) FROM people | db.people.count( { user\_id: { $exists: true } } )  db.people.find( { user\_id: { $exists: true } } ).count() |
| Update records | UPDATE people SET status = "C" WHERE age > 25 | db.people.updateMany( { age: { $gt: 25 } }, { $set: { status: "C" } } ) |
|  | UPDATE people SET age = age + 3 WHERE status = "A" | db.people.updateMany( { status: "A" } , { $inc: { age: 3 } } ) |
| Delete Records | DELETE FROM people WHERE status = "D" | db.people.deleteMany( { status: "D" } ) |
|  | DELETE FROM people | db.people.deleteMany({}) |

4.2.6. MongoDB hoạt động như thế nào



Hình 8: Kiến trúc của MongoDB

MongoDB hoạt động dưới một tiến trình ngầm service, luôn mở một cổng (Cổng mặc định là 27017) để nghe các yêu cầu truy vấn, thao tác từ các ứng dụng gửi vào sau đó mới tiến hành xử lý.

Mỗi một bản ghi của MongoDB được tự động gắn thêm một field có tên “\_id” thuộc kiểu dữ liệu ObjectId mà nó quy định để xác định được tính duy nhất của bản ghi này so với bản ghi khác, cũng như phục vụ các thao tác tìm kiếm và truy vấn thông tin về sau. Trường dữ liệu “\_id” luôn được tự động đánh index (chỉ mục) để tốc độ truy vấn thông tin đạt hiệu suất cao nhất.

Mỗi khi có một truy vấn dữ liệu, bản ghi được ghi đệm(cache) lên bộ nhớ Ram, để phục vụ lượt truy vấn sau diễn ra nhanh hơn mà không cần phải đọc từ ổ cứng.

Khi có yêu cầu thêm/sửa/xóa bản ghi, để đảm bảo hiệu suất của ứng dụng mặc định MongoDB sẽ chưa cập nhật xuống ổ cứng ngay, mà sau 60 giây MongoDB mới thực hiện ghi toàn bộ dữ liệu thay đổi từ RAM xuống ổ cứng.

4.2.7. Ưu và nhược điểm của MongoDB

**4.2.7.1. Ưu điểm**

- MongoDB là nguồn mở và miễn phí, có cộng đồng phát triển rất lớn.

- Hiệu năng cao:

+ Tốc độ truy vấn (find, update, insert, delete) của MongoDB nhanh hơn hẳn so với các hệ quản trị cơ sở dữ liệu quan hệ (RDBMS).

+ Thử nghiệm cho thấy tốc độ insert, tốc độ insert của MongoDB có thể nhanh tới gấp 100 lần so với MySQL.

**MongoDB có hiệu năng cao vì:**

+ MongoDB lưu dữ liệu dạng JSON, khi bạn insert nhiều đối tượng thì nó sẽ là insert một mảng JSON gần như với trường hợp insert 1 đối tượng

+ Dữ liệu trong MongoDB không có sự ràng buộc lẫn nhau như trong RDBMS, khi insert, delete hay update nó không cần phải mất thời gian kiểm tra xem có thỏa mãn các bảng liên quan như trong RDBMS.

+ Dữ liệu trong MongoDB được đánh chỉ mục (đánh index) nên khi truy vấn nó sẽ tìm rất nhanh.

+ Khi thực hiện insert, find… MongoDB sẽ khóa các thao tác khác lại, ví dụ khi nó thực hiện find(), trong quá trình find mà có thêm thao tác insert, update thì nó sẽ dừng hết lại để chờ find() xong .

* Dữ liệu linh hoạt: MongoDB là document database, dữ liệu lưu dưới dạng JSON, không bị bó buộc về số lượng field, kiểu dữ liệu…
* MongoDB có sẵn các method để thực hiện create, read, update, delete dữ liệu.
* Tính sẵn có: MongoDB hỗ trợ replica set nhằm đảm bảo việc sao lưu và khôi phục dữ liệu.
* Khả năng mở rộng Horizontal Scalability: Trong MongoDB có một khái niệm cluster là cụm các node chứa dữ liệu giao tiếp với nhau, khi muốn mở rộng hệ thống ta chỉ cần thêm một node mới vào cluster.

**4.2.7.2. Nhược điểm**

- MongoDB không có các tính chất ràng buộc như trong RDBMS nên dễ bị làm sai dữ liệu.

- Không hỗ trợ join giống như RDBMS nên khi viết function join trong code ta phải làm bằng tay khiến cho tốc độ truy vấn bị giảm.

- Sử dụng nhiều bộ nhớ: do dữ liệu lưu dưới dạng key-value, các collection chỉ khác về value do đó key sẽ bị lặp lại. Không hỗ trợ join nên sẽ bị dư thừa dữ liệu.

- Bị giới hạn kích thước bản ghi: mỗi document không được có kích thước > 16Mb và không mức độ các document con trong 1 document không được > 100.

4.2.8. Khi nào nên và không nên sử dụng MongoDB

**4.2.8.1. Khi nào nên sử dụng MongoDB**

Sử dụng MongoDB trong trường hợp:

* Hệ thống realtime (thời gian thực) yêu cầu phản hồi nhanh
* Các hệ thống bigdata với yêu cầu truy vấn nhanh.
* Các hệ thống có tần suất write/insert lớn
* Sử dụng làm search engine.

**4.2.8.2. Khi nào không nên sử dụng MongoDB**

* Các ứng dụng cần sử dụng nhiều transaction (như ngân hàng) do MongoDB không có cơ chế transaction (giao dịch) để phục vụ cho các ứng dụng ngân hàng
* Các ứng dụng cần SQL (sử dụng joins).

4.2.9. Các công cụ trực quan

Một số công cụ trực quan làm việc với MongoDB điển hình như:

* RoboMongo
* UMongo
* MongoExplorer
* RockMongo

4.2.10. Cài đặt MongoDB

* **Yêu cầu hệ thống**

+ Yêu cầu phần cứng: Hệ thống có kiến trúc 64 bit x86

+ Yêu cầu phần mềm: Windows 7 / Server 2008 R2 hoặc mới hơn.

* **Cài đặt MongoDB (phiên bản MongoDB Community Server )**

+ Tải MongoDB từ link (tải file .msi):

https://www.mongodb.com/download-center/community

+ Chạy file vừa tải về.

+ Làm theo hướng dẫn cài đặt MongoDB

Trình hướng dẫn sẽ hướng dẫn bạn cài đặt MongoDB và MongoDB Compass.

Bạn có thể chọn loại thiết lập **Hoàn thành** hoặc **Tùy chỉnh** . Nếu bạn chọn chọn Cài đặt **tùy chỉnh** , bạn có thể chỉ định thư mục cài đặt. MongoDB không có phụ thuộc hệ thống khác. Bạn có thể cài đặt và chạy MongoDB từ bất kỳ thư mục nào.

Bắt đầu từ MongoDB 4.0, bạn có thể cài đặt theo mặc định hoặc cấu hình và khởi động MongoDB như một server trong khi cài đặt, và MongoDB server được bắt đầu khi cài đặt thành công.

4.3. Demo ứng dụng của NoSQL

4.3.1. Phát biểu bài toán

Để vận dụng được lý thuyết về cơ sở dữ liệu NoSQL vào một bài toán cụ thể, em lựa chọn xây dựng một module quản lý user.

Module này cần lưu trữ các dữ liệu: Name, email, address,age.

4.3.2. Thực hiện xây dựng ứng dụng demo

**4.3.2.1. Cài đặt môi trường**

Chương trình demo ứng dụng của NoSQL được xây dựng trên ngôn ngữ php và lưu trữ dữ liệu bằng MongoDB.

Yêu cầu cài đặt:

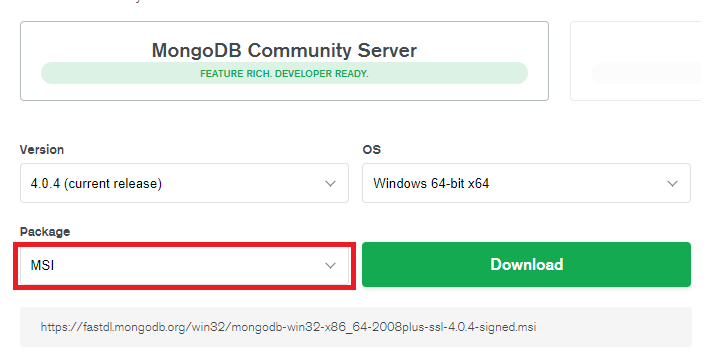
+ XAMPP

+ MongoDB

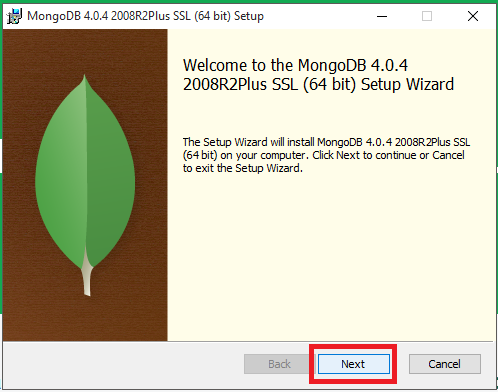
+ Robo3T

***Cài đặt MongoDB:***

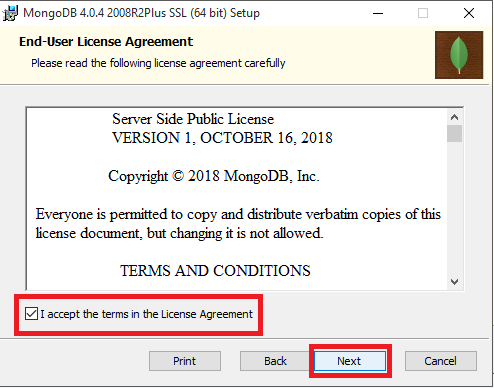
Đầu tiên, tải file cài đặt MongoDB:



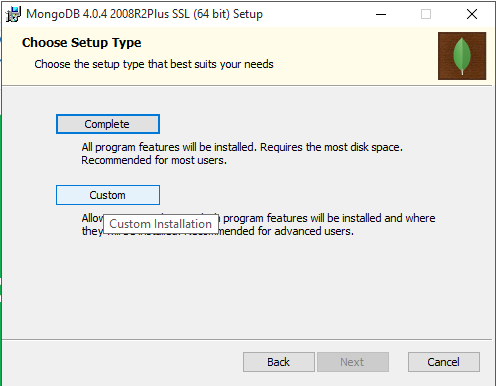
Hình 9: Tải file cài đặt MongoDB



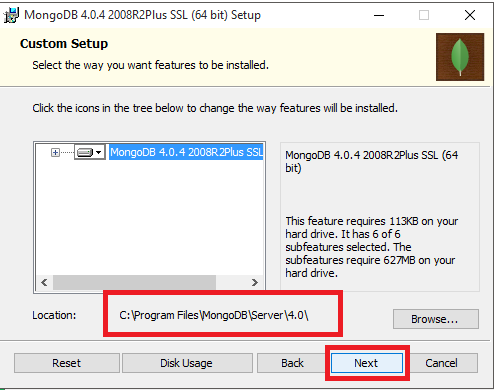
Hình 10: Bắt đầu cài đặt MongoDB



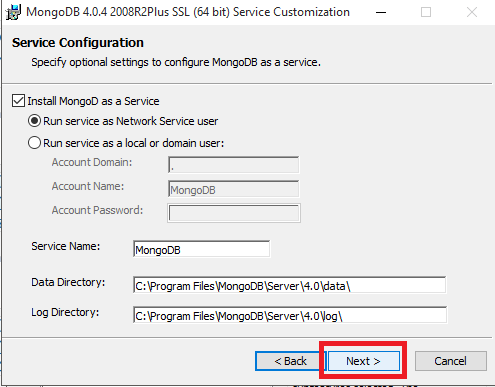
Hình 11: Đồng ý với thỏa thuận



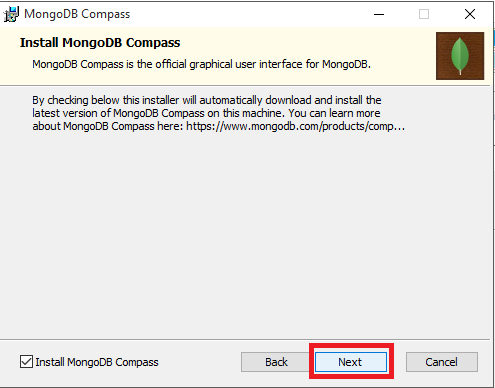
Hình 12: Chọn kiểu setup



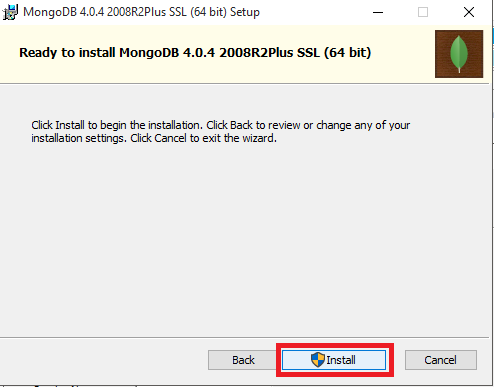
Hình 13: Thư mục lưu trữ



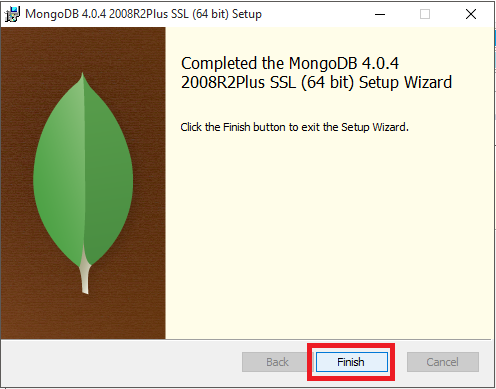
Hình 14: Cài đặt service



Hình 15: Cài đặt MongoDB Compass



Hình 16. Isntall



Hình 17: Kết thúc

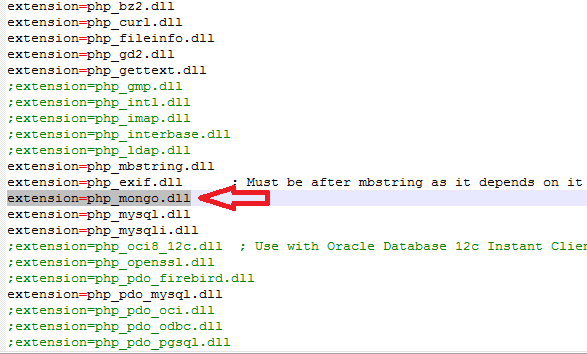
**Cài đặt PHP – MongoDB Driver**

Để có thể sử dụng MongoDB với PHP, phải cần có PHP-MongoDB driver. Tải xuống trình điều khiển từ url điều khiển PHP:

<https://s3.amazonaws.com/drivers.mongodb.org/php/index.html>

Tải xuống bản phát hành mới nhất của nó. Sau đó giải nén kho lưu trữ và đặt php\_mongo.dll trong thư mục mở rộng PHP (mặc định: C:\xampp\php\ext) thêm dòng sau vào tệp php.ini của bạn:

|  |
| --- |
| extension = php\_mongo.dll |



Hình 18: Chèn php\_mongo.dll vào php.ini

Bây giờ chúng ta sẽ tạo một cơ sở dữ liệu MongoDB mới có tên mongo và tạo một collection với tên: *users* sau đó thêm một số dữ liệu vào collection đó.

Tạo chỉ mục duy nhất cho trường email:

|  |
| --- |
| db.users.createIndex({"email":1}, {unique:true}) |

Trường “email” là duy nhất để các mục trùng lặp cho email sẽ tạo ra thông báo lỗi. Mỗi người dùng được nhận dạng duy nhất bằng "email". ID người dùng trong MongoDB được tạo tự động dưới dạng ObjectId 12 byte.

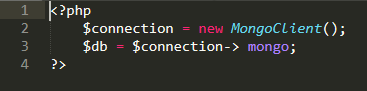
Chèn dữ liệu vào collection *users:*

|  |
| --- |
| >db.users.insert{[  "\_id" : ObjectId("5c09714844f087a81c00002c"),  "name" : "Phạm Thu Ngân",  "email" : "ptngan7194@gmail.com",  "address" : "Hải Dương",  "age" : "25"  },  {  "\_id" : ObjectId("5c098b8444f087ac1c00002e"),  "name" : "Phạm Nguyễn Anh Chi",  "email" : "chi432@gmail.com",  "address" : "Hà Nội",  "age" : "29"  },  {  "\_id" : ObjectId("5c098bce44f087ac1c00002f"),  "name" : "Nguễn Mạnh Hùng",  "email" : "hungnguyen432@gmail.com",  "address" : "Hồ Chí Minh",  "age" : "19"  }  ]} |

Truy vấn collection (hiển thị có cấu trúc):

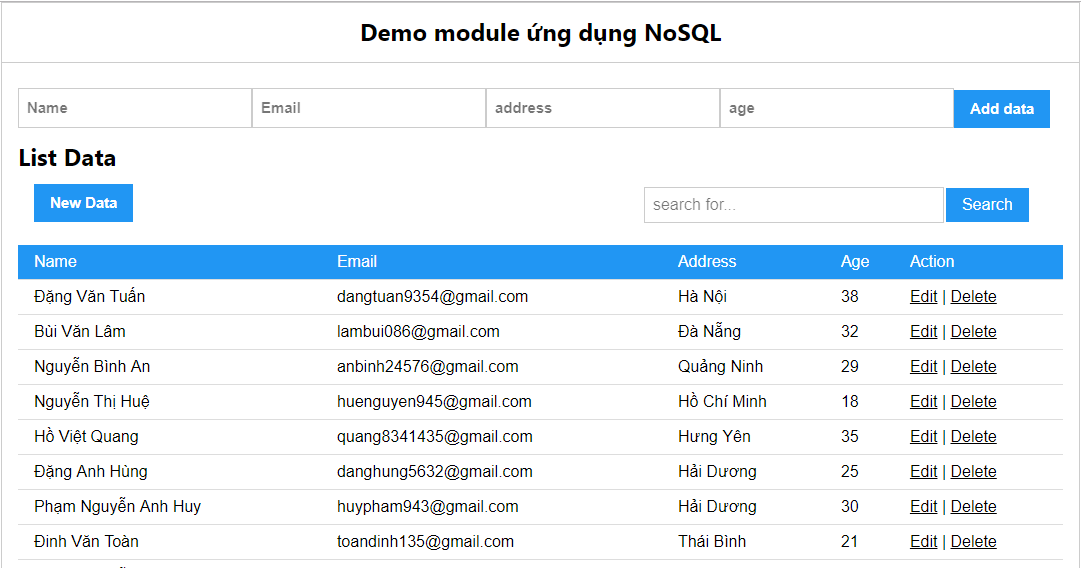
|  |
| --- |
| >db.users.find().pretty()  {  "\_id" : ObjectId("5c09714844f087a81c00002c"),  "name" : "Phạm Thu Ngân",  "email" : "ptngan7194@gmail.com",  "address" : "Hải Dương",  "age" : "25"  },  {  "\_id" : ObjectId("5c098b8444f087ac1c00002e"),  "name" : "Phạm Nguyễn Anh Chi",  "email" : "chi432@gmail.com",  "address" : "Hà Nội",  "age" : "29"  },  {  "\_id" : ObjectId("5c098bce44f087ac1c00002f"),  "name" : "Nguễn Mạnh Hùng",  "email" : "hungnguyen432@gmail.com",  "address" : "Hồ Chí Minh",  "age" : "19"  } |

Tạo một tệp chứa mã kết nối cơ sở dữ liệu MongoDB: config.php.



Hình 19: Kết nối cơ sở dữ liệu MongoDB và php

Để thêm dữ liệu vào cơ sở dữ liệu, chúng ta cần một biểu mẫu.



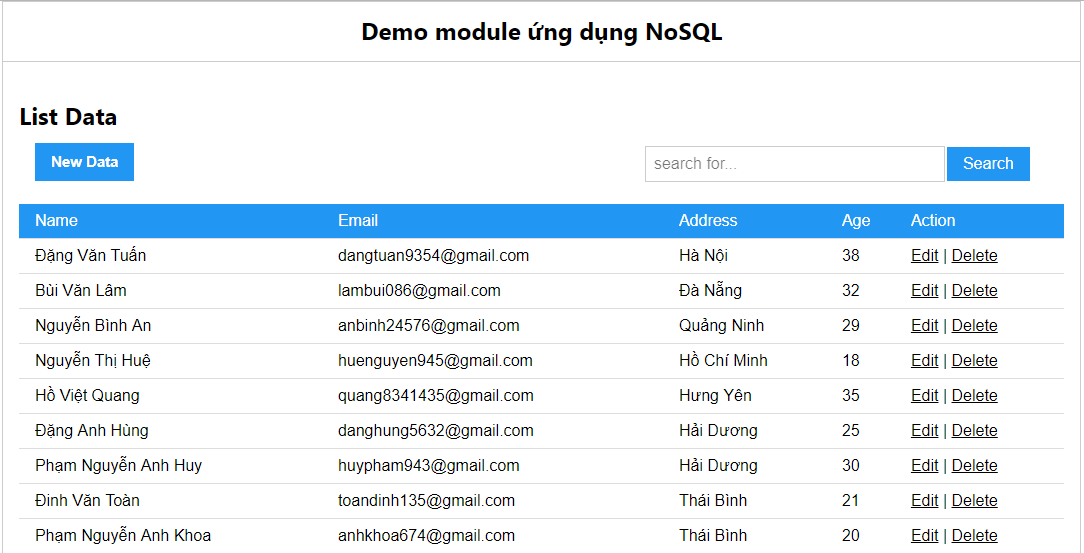
Hình 20: Add new data

Nhập dữ liệu cần thêm vào biểu mẫu. Dữ liệu nhập vào biểu mẫu sẽ được lấy ra và xử lý:

|  |
| --- |
| // Lấy dữ liệu ra sau đó truyền vào mảng  $user = array (  'name' => $\_POST['name'],  'email' => $\_POST['email'],  'address' => $\_POST['address'],  'age' => $\_POST['age']  );  $errorMessage = '';  foreach ($user as $key => $value) {  if (empty($value)) {  $errorMessage .= $key . ' field is empty<br /> ' ;  }  }  if ($errorMessage) {  // Thông báo lỗi:  echo '<span style="color:red">'.$errorMessage.'</span>';  echo "<br/><a href='javascript:self.history.back();'>Go Back</a>";  } else {  //Chèn dữ liệu vào collection 'users'  $db->users->insert($user);  header('Location:../index.php?a=file&ac=them') } |

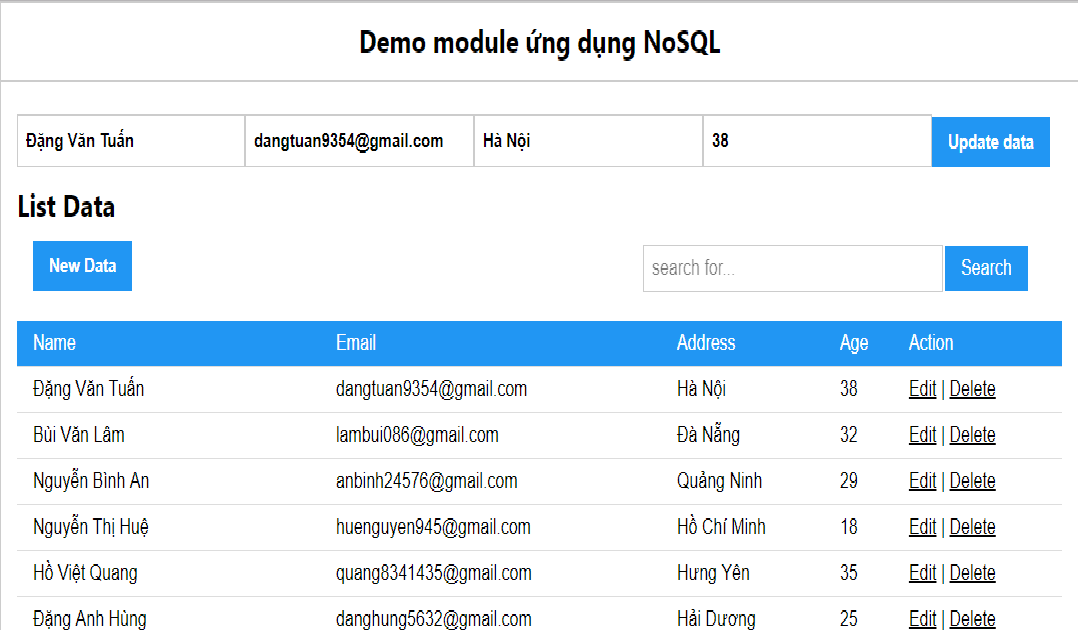
Thực hiện xác nhận đơn giản để kiểm tra xem tên, email và tuổi đã nhập có trống hay không. Nếu tất cả đều được điền thì dữ liệu sẽ được chèn vào bảng collection trong cơ sở dữ liệu.

Dữ liệu từ cơ sở dữ liệu được tìm nạp và hiển thị trong index.php. Tệp này cũng chứa liên kết để thêm dữ liệu. Trên mỗi hàng dữ liệu được hiển thị, cũng có một liên kết để chỉnh sửa và xóa dữ liệu:



Hình 21: Trang chủ

Mỗi hàng / document của dữ liệu có thể được chỉnh sửa riêng biệt. ID hàng / document được chuyển vào URL của sua.php. ID xác định duy nhất mục nhập dữ liệu.



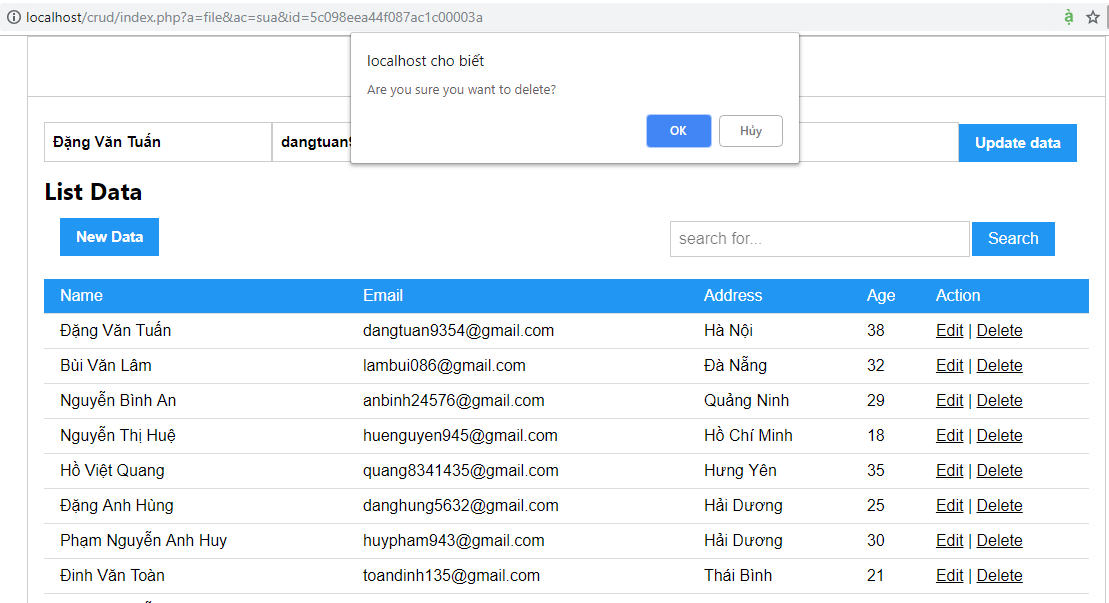
Hình 22: Update data

Trong đoạn mã bên dưới, đầu tiên, một mục nhập dữ liệu của hàng đơn được lấy dựa trên id. Dữ liệu được tìm, nạp và được hiển thị trong biểu mẫu chỉnh sửa. Khi người dùng chỉnh sửa dữ liệu và gửi biểu mẫu, thì một số xác thực đơn giản được thực hiện cho dữ liệu trống. Khi mọi thứ đều chính xác, thì mục nhập dữ liệu cụ thể đó được cập nhật trong cơ sở dữ liệu.

|  |
| --- |
| $id = $\_POST['id'];  $user = array (  'name' => $\_POST['name'],  'email' => $\_POST['email'],  'address' => $\_POST['address'],  'age' => $\_POST['age']  );  $errorMessage = “'';  foreach ($user as $key => $value) {  if (empty($value)) {  $errorMessage .= $key . ' field is empty<br />';  }  }  if ($errorMessage) {  // Thông báo lỗi  echo '<span style="color:red" onClick=\"return confirm('.$errorMessage.')\">&times;</span>';  } else {  //Update lại dữ liệu vào collection user  $db->users->update(  array('\_id' => new MongoId($id)),  array('$set' => $user)  ); |

Mỗi hàng / document của dữ liệu có thể được xóa riêng biệt. ID hàng / document được chuyển vào URL. Sau khi xóa, người dùng được chuyển hướng đến trang chủ (index.php).

|  |
| --- |
| //chuyển id vào URL  $id = $\_GET['id'];  //xóa document  $db->users->remove(array('\_id' => new MongoId($id)));  //quay lại trang index  header("Location:../index.php"); |



Hình 23: Delete data

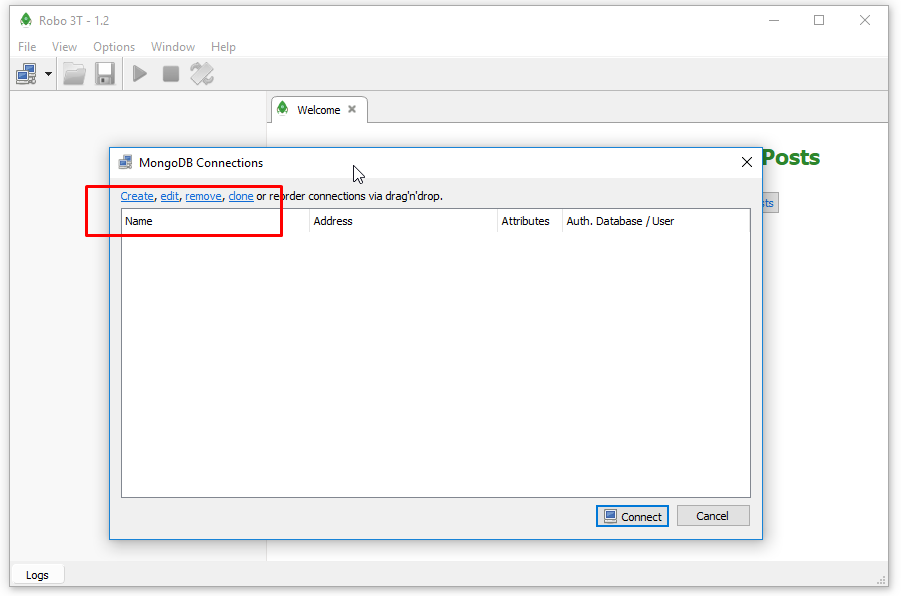
**4.3.1.2. Thực hiện tạo cơ sở dữ liệu sử dụng Robo3T (Robo Mongo):**

**Bước** 1: Tải file cài đặt Robo3T (.exe).

Link tải: https://robomongo.org/download

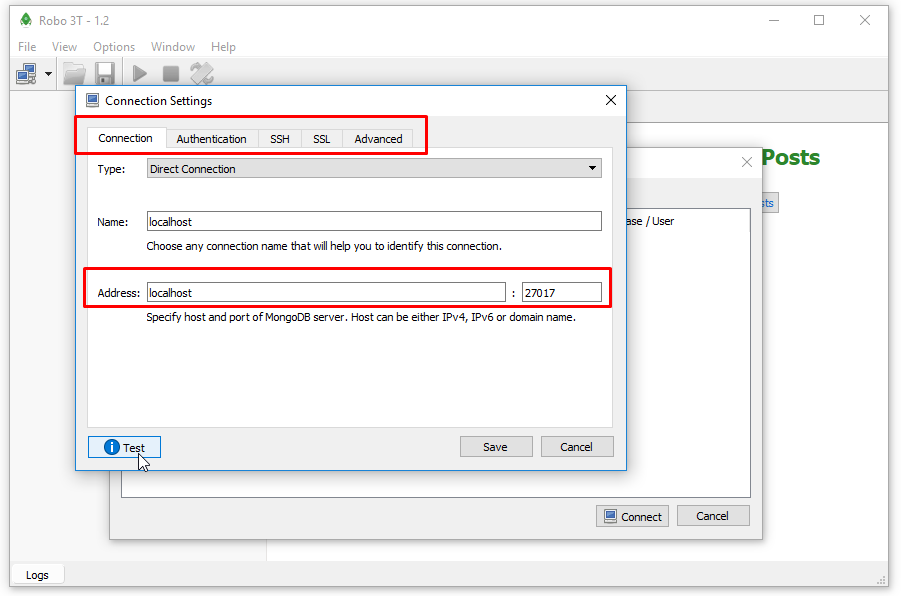
**Bước 2**: Cài đặt Robo 3T.

**Bước 3**: Mở Robo 3T và tạo kết nối với MongoDB server.



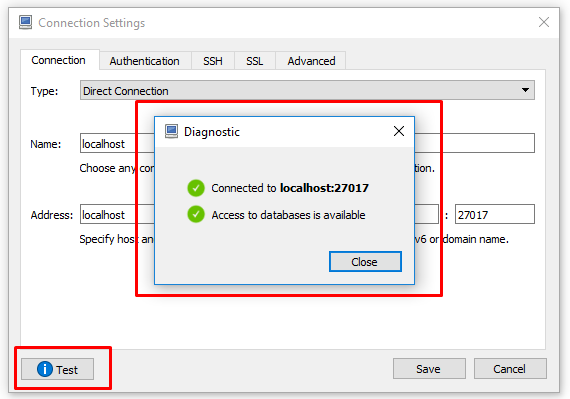
Hình 24: Giao diện khi mở Robo 3T lần đầu

Click vào Create để tạo kết nối tới MongoDB.



Hình 25: Tạo kết nối đến MongoDB server

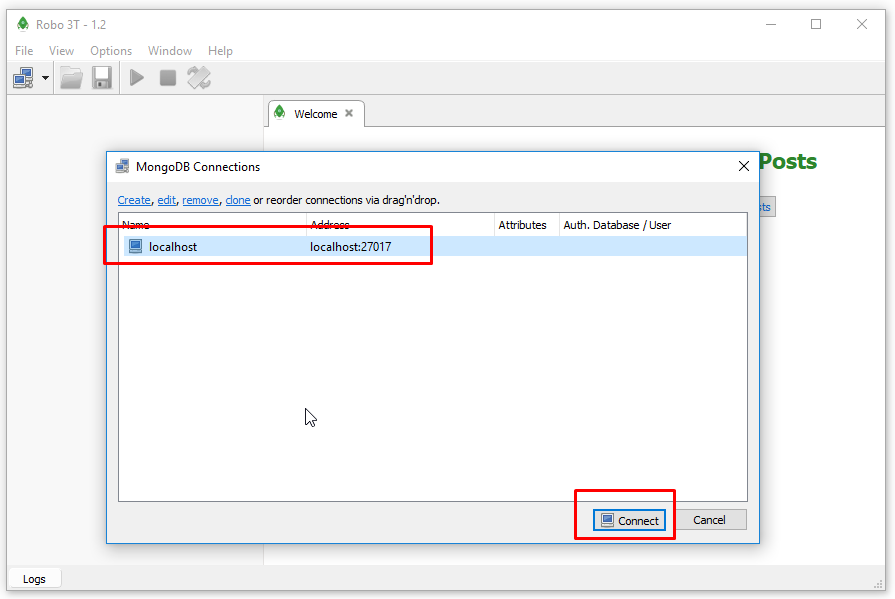
Click vào Test để kiểm tra kết nối.



Hình 26: Kiểm tra kết nối

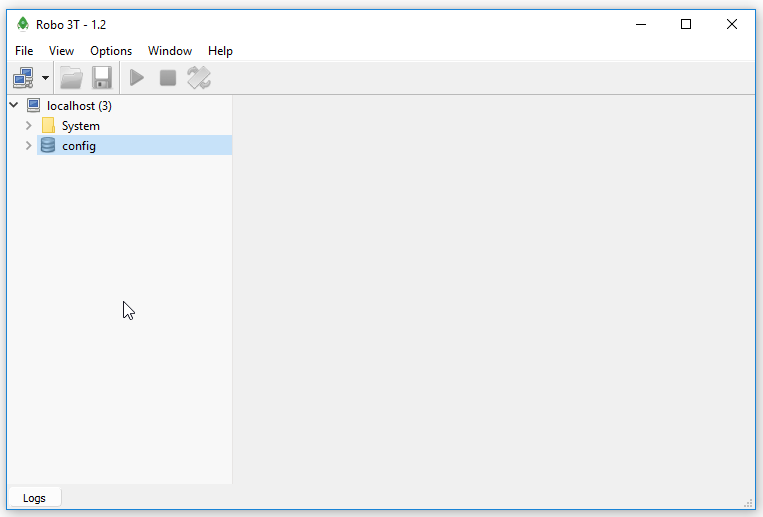
Sau khi save lại kết nối (connection) nó sẽ xuất hiện trên bảng connection cho bạn chọn.

Click vào connect để kết nối tới MongoDB.



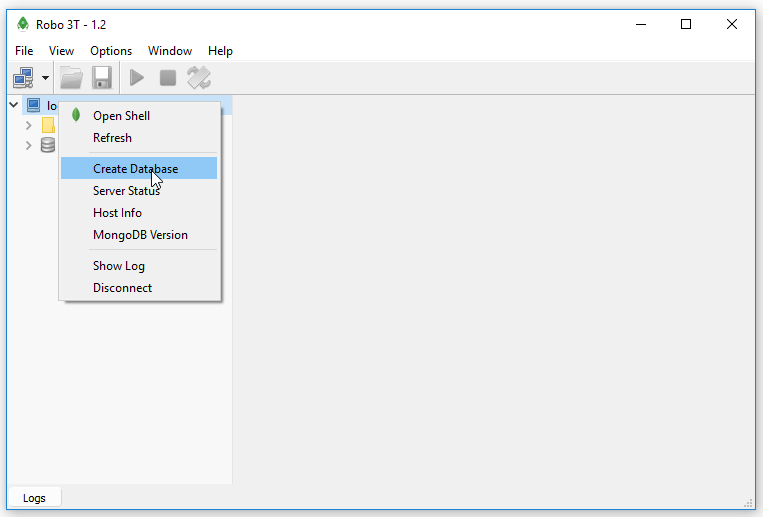
Hình 27: Kết nối đến MongoDB server

Dưới đây là giao diện RoboMongo sau khi kết nối, phần bên trái là danh sách các database.



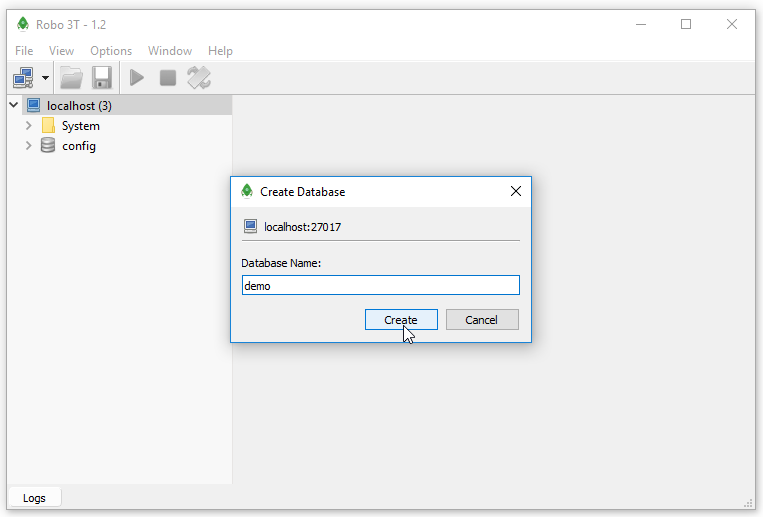
Hình 28: Giao diện Robo 3T khi kết nối lần đầu

**Bước 4**: Tạo cơ sở dữ liệu.



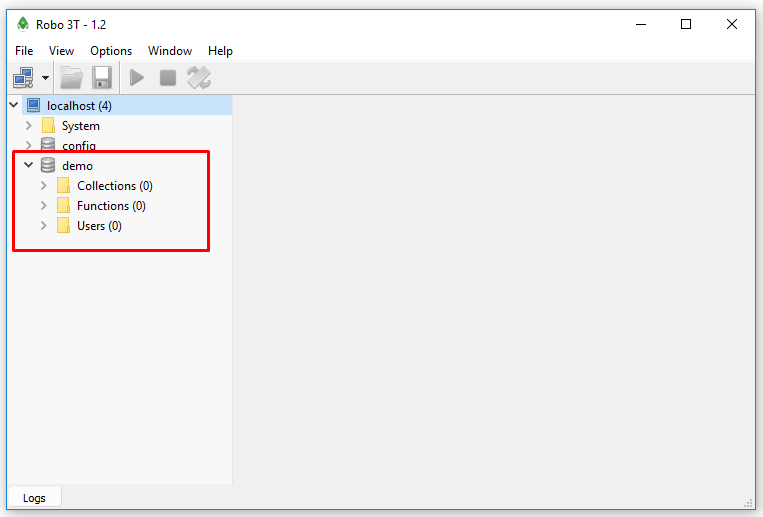
Hình 29: Tạo cơ sở dữ liệu

Đặt tên cho cơ sở dữ liệu:



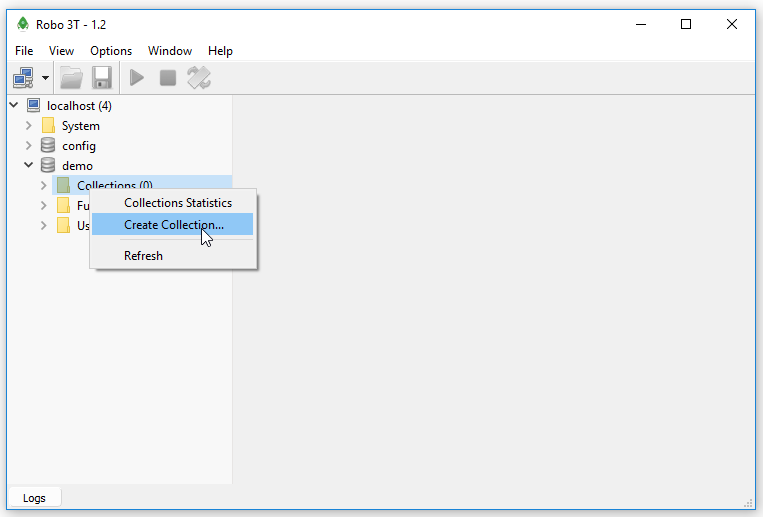
Hình 30: Đặt tên cho cơ sở dữ liệu

Kết quả:

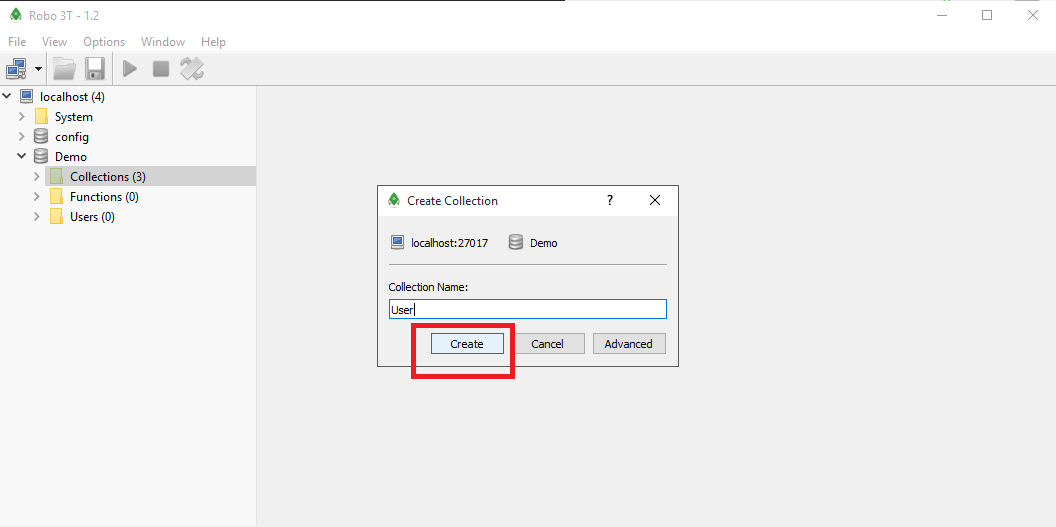


Hình 31: Tạo cơ sở dữ liệu thành công

Tạo mới Collection:

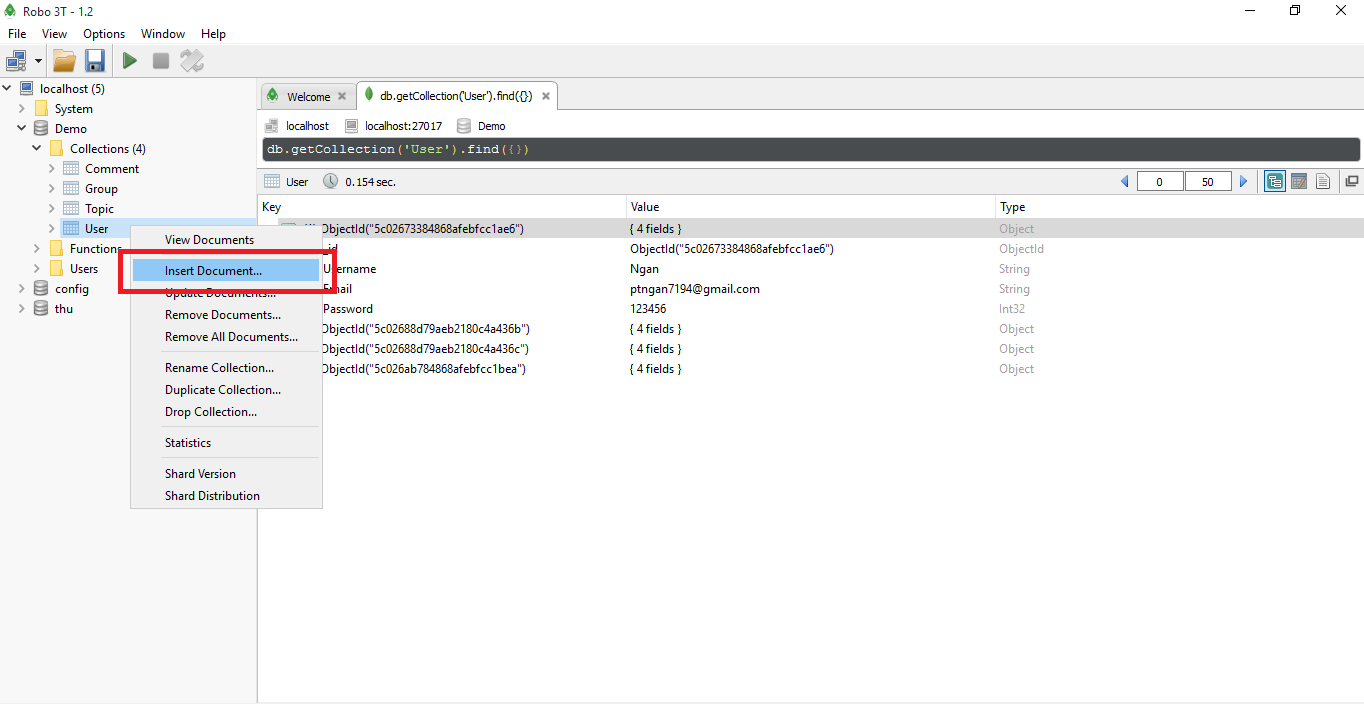


Hình 32: Tạo Collection

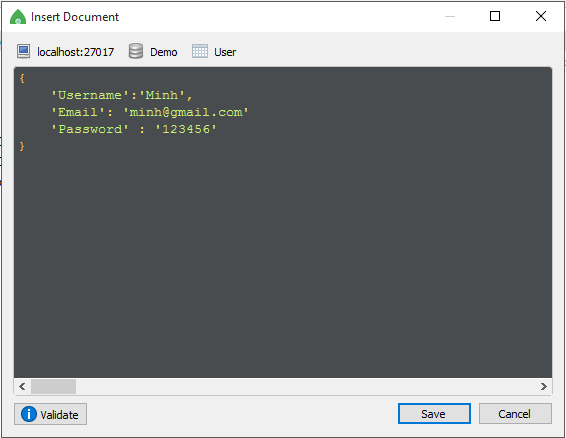


Hình 33: Đặt tên cho Collection

Insert Document vào Collection:

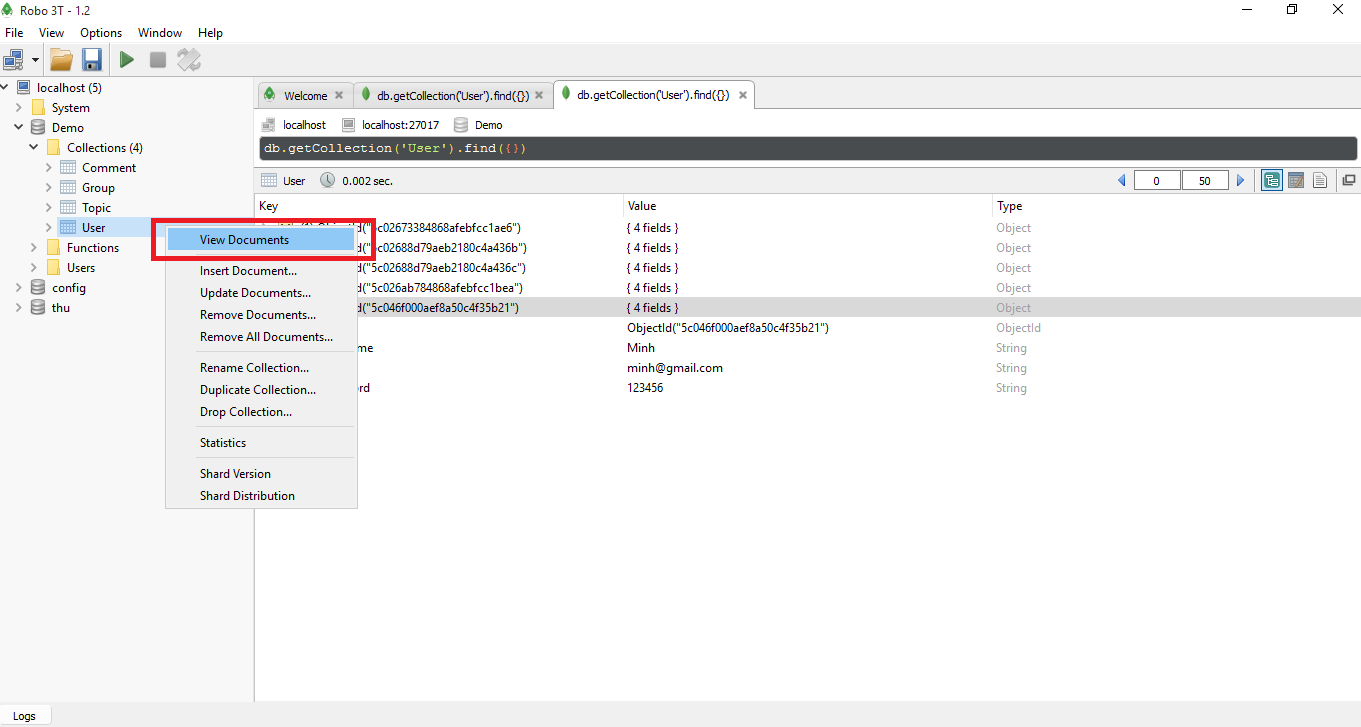


Hình 34: Insert Document

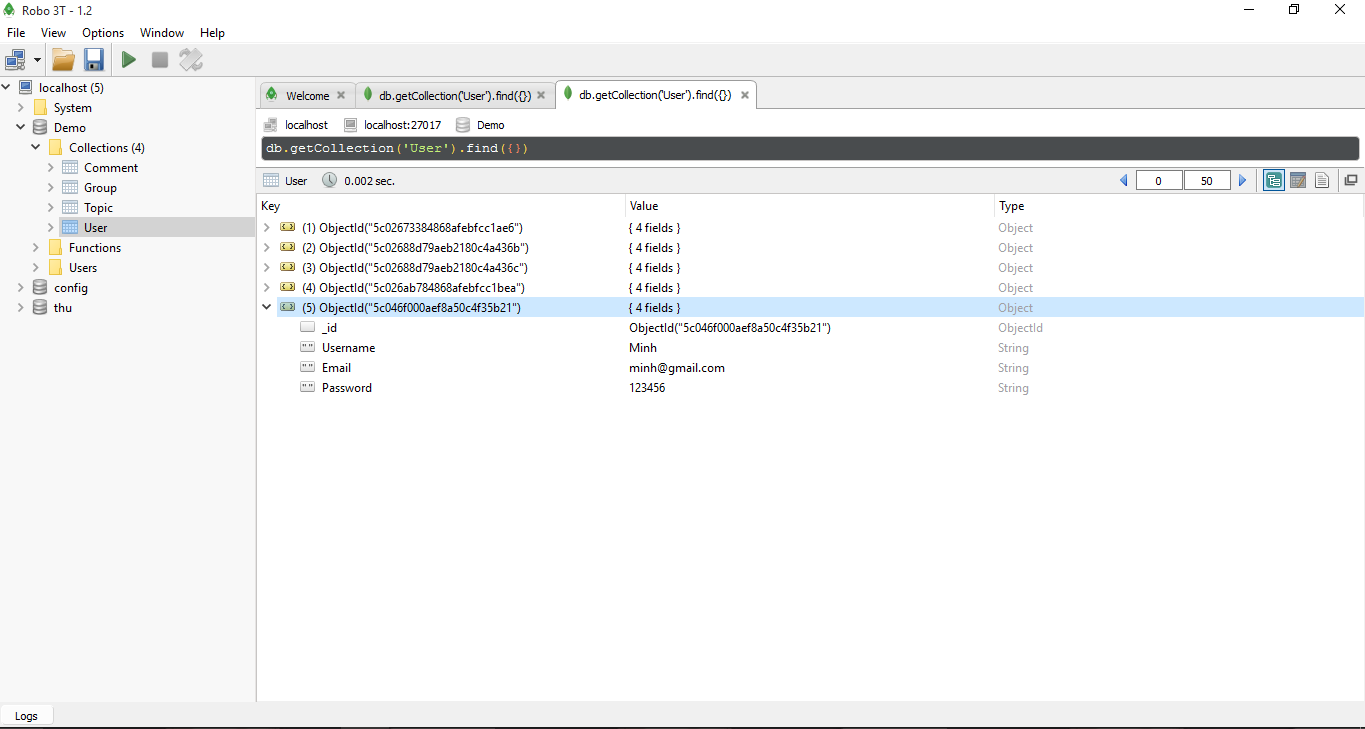


Hình 35: Giao diện thực hiện Insert Document

Kết quả:



Hình 36: Xem Document



Hình 37:Giao diện Document đã tạo thành công

**4.3.3.3. Một số giao diện**

PHẦN V. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1. Kết luận

Qua thời gian tìm hiểu và thực hiện đề tài: “Tìm hiểu về NoSQL và ứng dụng”, em đã thu được một số kết quả như sau:

- Về lý thuyết:

+ Tìm hiểu tổng quan về cơ sở dữ liệu NoSQL: đặc điểm, phân loại cùng với những ứng dụng thực tiễn của nó.

+ So sánh cơ sở dữ liệu NoSQL với cơ sở dữ liệu quan hệ và xml.

+ Tìm hiểu tổng quan một số hệ quản trị cơ sở dữ liệu dạng NoSQL: RavenDB, Hadoop, Cassandra, CouchDB.

+ Tìm hiểu về hệ quản trị cơ sở dữ liệu MongoDB và vận dụng kiến thức đã tìm hiểu được vào một ứng dụng thực tế.

- Về thực nghiệm:

Xây dựng được chương trình demo module quản lý User sử dụng hệ quản trị cơ sở dữ liệu MongoDB. Ứng dụng áp dụng được những lý thuyết cơ bản của MongoDB.

5.2. Hướng phát triển

- Tiếp tục hoàn thiện demo ứng dụng của NoSQL:

+ Giao diện: Hệ thống giao diện thân thiện, dễ sử dụng và đẹp

TÀI LIỆU THAM KHẢO

**Tài liệu tiếng anh**

[1]. Kristina Chodorow and Michael Dirolf, September 2010, *MongoDB The Definitive Guide,* Printed in the United States of America, Published by O’Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

[2]. Kristina Chodorow, April 2011, *50 Tips and Tricks for MongoDB Deverlopers,* Printed in the United States of America, Published by O’Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

[3]. Rubayeet Islam, November 2011, PHP and MongoDB Web Deverlopment,

Published by Packt Publishing Ltd., Livery Place, 35 Livery Street, Birmingham B3 2PB, UK.

**Tài liệu Internet**

[4]. List of NoSQL databases: http://nosql-database.org/

[1]. DB-Engines Ranking: https://db-engines.com/en/ranking

[3]. Mongo Manual: <https://docs.mongodb.com/>

[4]. Cơ bản về MongoDB: <https://vietjack.com/mongodb/index.jsp>

[4]. Series: MongoDB: <https://stackjava.com/series/mongodb>

[5]. MongoDB Tutorial: <https://www.tutorialspoint.com/mongodb/index.htm>

Ngày 03 tháng 12 năm 2018

|  |  |
| --- | --- |
| **GIẢNG VIÊN HƯỚNG DẪN**  *(Ký và ghi rõ họ tên)* | **SINH VIÊN**  *(Ký và ghi rõ họ tên)* |