

**BỘ CÔNG THƯƠNG**  
**TRƯỜNG ĐẠI HỌC CÔNG THƯƠNG TPHCM**

**Nguyễn Văn Lễ**  
**Nguyễn Thị Định**



**TÀI LIỆU HƯỚNG DẪN THỰC HÀNH**  
**CƠ SỞ DỮ LIỆU NoSQL**

**(Lưu hành nội bộ)**

**THÀNH PHỐ HỒ CHÍ MINH – NĂM 2024**

# MỤC LỤC

<b>CHƯƠNG 1: HỆ QUẢN TRỊ CSDL MONGODB .....</b>	<b>1</b>
1.1. CÀI ĐẶT MONGODDB TRÊN WINDOWS .....	1
1.1.1. Giới thiệu .....	1
1.1.2. Cài đặt MongoDB.....	2
1.1.3. Cấu hình MongoDB.....	4
1.1.4. Một số công cụ hỗ trợ thao tác trên MongoDB .....	9
1.2. CÁC MÔ HÌNH DỮ LIỆU .....	12
1.3. SAO LƯU VÀ PHỤC HỒI DỮ LIỆU .....	13
1.3.1. Export và Import collection .....	13
1.3.2. Sao lưu và phục hồi .....	14
1.4. CÁC THAO TÁC CẬP NHẬT DỮ LIỆU.....	14
1.4.1. Một số lệnh thao tác cơ bản.....	14
1.4.2. Cập nhật dữ liệu.....	14
1.5. TRUY VẤN DỮ LIỆU .....	17
1.5.1. Sử dụng lệnh find .....	17
1.5.2. Truy vấn dữ liệu với Aggregation framework.....	21
1.6. BÀI TẬP THỰC HÀNH CHƯƠNG 1 .....	26
<b>CHƯƠNG 2: HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU NEO4J.....</b>	<b>30</b>
2.1. TỔNG QUAN .....	30
2.1.1. Giới thiệu .....	30
2.1.2. Một số ứng dụng của đồ thị.....	31
2.2. NGÔN NGỮ CYPHER.....	32
2.2.1. Kiểu dữ liệu trong Cypher .....	32
2.2.2. Quy cách đặt tên .....	32
2.2.3. Sử dụng ghi chú (comment) .....	33
2.2.4. Biểu diễn các nút của đồ thị .....	33
2.2.5. Biến và nhãn nút.....	33
2.2.6. Quan hệ và biến quan hệ.....	34
2.2.7. Thuộc tính đối tượng .....	34
2.2.8. Mẫu trong Cypher.....	35
2.2.9. Thao tác trên cửa sổ lệnh.....	35
2.3. CÁC LỆNH CẬP NHẬT DỮ LIỆU .....	35

2.3.1.	Thao tác trên nút .....	35
2.3.2.	Thao tác trên mối liên kết .....	36
2.3.3.	Ví dụ tạo cơ sở dữ liệu đồ thị .....	37
2.4.	TRUY VẤN DỮ LIỆU .....	39
2.4.1.	Các lệnh truy vấn cơ bản .....	39
2.4.2.	Sử dụng mệnh đề Where .....	39
2.4.3.	Xử lý chuỗi .....	40
2.4.4.	Exists và Not Exists với pattern .....	41
2.4.5.	Một số truy vấn phức tạp.....	41
2.4.6.	Aggregation trong Cypher .....	41
2.4.7.	Kết hợp giá trị với collect .....	42
2.4.8.	Sử dụng WITH .....	42
2.4.9.	Sử dụng UNWIND .....	43
2.4.10.	Sắp xếp kết quả.....	44
2.4.11.	Loại bỏ dòng trùng .....	44
2.4.12.	Giới hạn kết quả trả về.....	44
2.5.	BÀI TẬP CHƯƠNG 2 .....	45
	<b>CHƯƠNG 3: ỨNG DỤNG KẾT NỐI CSDL NoSQL.....</b>	<b>49</b>
3.1.	ỨNG DỤNG KẾT NỐI MONGODB .....	59
3.1.1.	Ứng dụng trên C# kết nối MongoDB .....	59
3.1.2.	Ứng dụng trên Java kết nối MongoDB.....	63
3.2.	ỨNG DỤNG KẾT NỐI NEO4J .....	65
3.2.1.	Cài đặt gói chương trình driver .....	65
3.2.2.	Kết nối Neo4J .....	65
3.2.3.	Lệnh thao tác .....	66
3.2.4.	Sử dụng tham số .....	67
3.3.	BÀI TẬP CHƯƠNG 3 .....	67

# CHƯƠNG 1: HỆ QUẢN TRỊ CSDL MONGODB

## 1.1. CÀI ĐẶT MONGODB TRÊN WINDOWS

### 1.1.1. Giới thiệu


Trong những năm gần đây, với sự ra đời và phát triển mạnh mẽ của NoSQL thì MongoDB cũng đang nhận được nhiều sự chú ý trong cộng đồng công nghệ. Điểm mạnh của NoSQL nói chung và MongoDB nói riêng đó là tính linh hoạt trong việc cấu trúc dữ liệu giúp đáp ứng tốt với những thay đổi hay việc mở rộng cơ sở dữ liệu.

MongoDB là một chương trình cơ sở dữ liệu mã nguồn mở được thiết kế theo kiểu hướng đối tượng trong đó các bảng được cấu trúc một cách linh hoạt cho phép các dữ liệu lưu trên bảng không cần phải tuân theo một dạng cấu trúc nhất định nào. Chính do cấu trúc linh hoạt này nên MongoDB có thể được dùng để lưu trữ các dữ liệu có cấu trúc phức tạp, đa dạng và không cố định (hay còn gọi là Big Data).

MongoDB sử dụng thuật ngữ collection hay bộ sưu tập thay vì bảng như trong cơ sở dữ liệu quan hệ. Các bảng trong cơ sở dữ liệu quan hệ được cấu trúc với một số lượng cột (column) nhất định và các cột này cũng được định nghĩa với kiểu dữ liệu nhất định. Ngược lại MongoDB lưu document (hay tài liệu tương tự như các record trong MySQL hay SQL Server) vào các collection với định dạng JSON hay Javascript Object Notation. Do đó khi truy vấn hay cập nhật dữ liệu của document trong MongoDB chúng ta sử dụng cú pháp theo kiểu hướng đối tượng.

**MongoDB Document:** Một document trong MongoDB có thể được hiểu tương đương như một record trong MySQL hay SQL Server. Tuy nhiên sự khác biệt đó là các document trong MongoDB không sử dụng cấu trúc cố định. Ví dụ sau về một document trong MongoDB dùng để lưu dữ liệu của một người dùng trên một mạng xã hội:

```
{
  name: "sue",
  age: 26,
  status: "A",
  groups: [ "news", "sports" ]
}
```

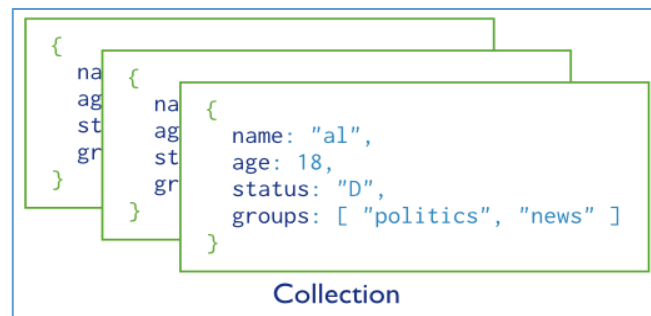


field: value  
field: value  
field: value  
field: value

Ở ví dụ trên cho thấy dữ liệu của document được lưu theo kiểu JSON với các trường như name, age, status và groups. Mặc dù document ở trên chỉ có 4 trường nhưng MongoDB vẫn cho phép chúng ta có thể thêm vào một document khác với 5 trường hoặc 3 hay 2 trường. Đây chính là tính linh động của NoSQL.

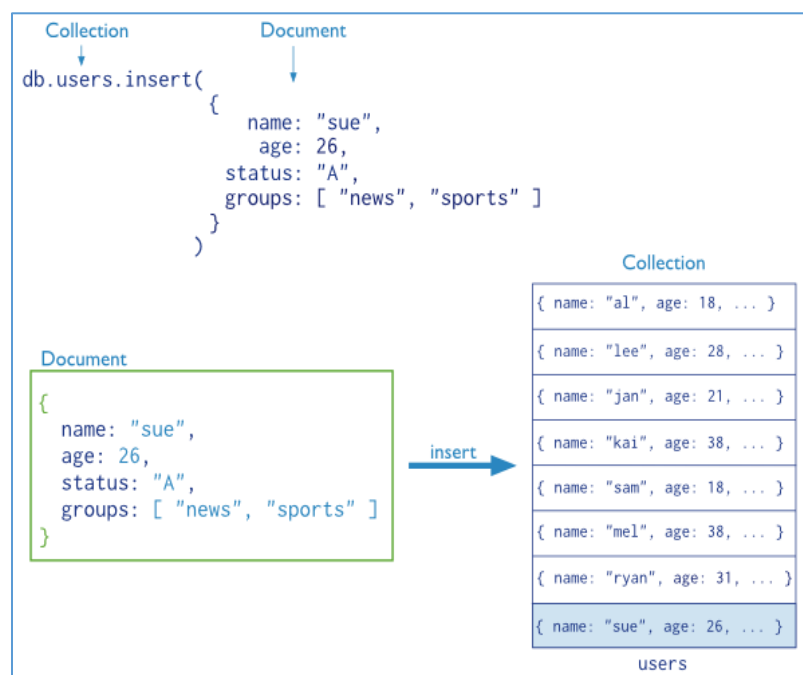
**MongoDB Collection:** Collection trong MongoDB tương đương với các bảng sử dụng trong các cơ sở dữ liệu kiểu quan hệ. Collection bao gồm tập hợp tất cả

các document riêng lẻ được lưu trên collection đó. Ví dụ sau đây minh họa về collection trong MongoDB.



Sự khác biệt cơ bản giữa collection và bảng trong MySQL hay SQL Server đó là không cần định nghĩa một cấu trúc schema cố định nào cho Collection, thay vào đó là chỉ cần định nghĩa tên cho collection và một số tùy chọn khác như index hay size của collection.

Thêm Document vào Collection: Tạo một document vào một collection có tên là users.





### 1.1.2. Cài đặt MongoDB

Hướng dẫn cài đặt MongoDB trên Windows 10 professional.

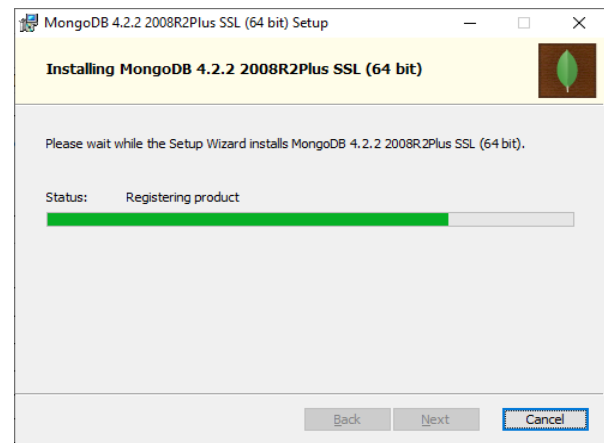
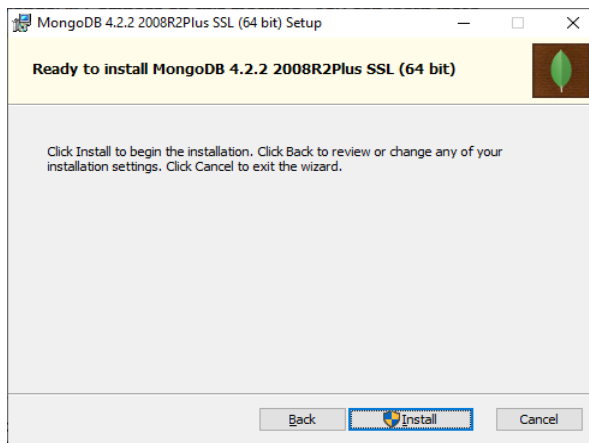
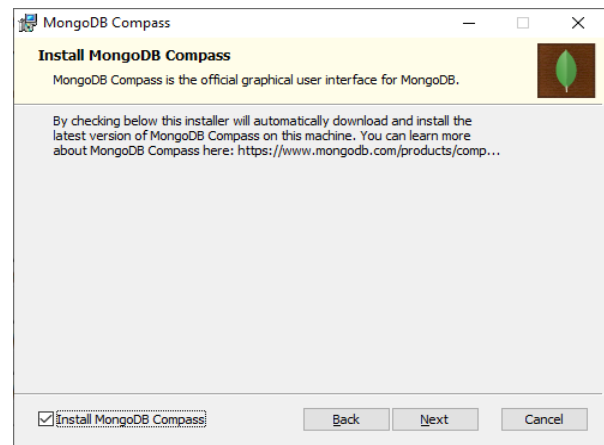
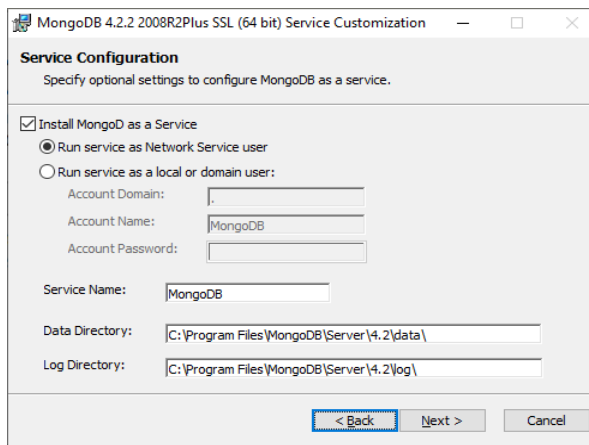
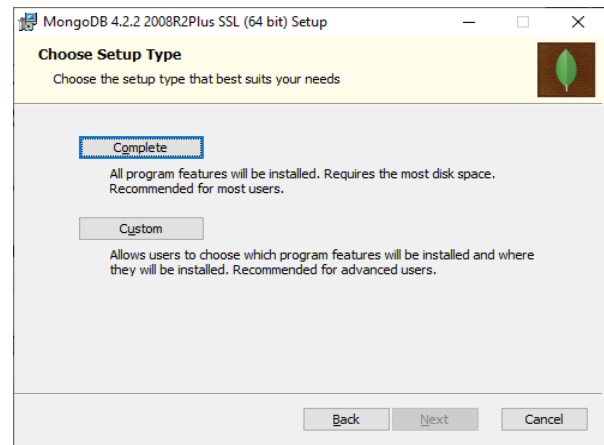
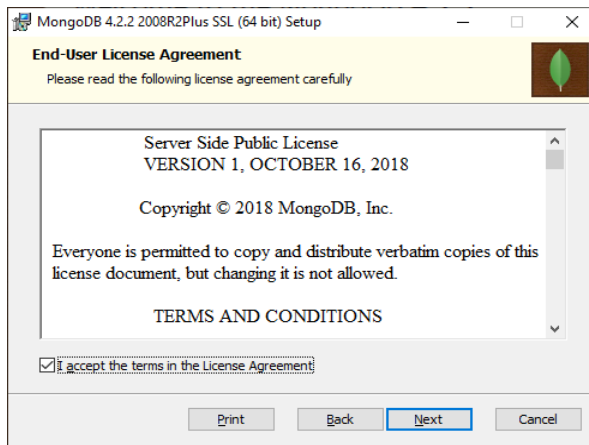
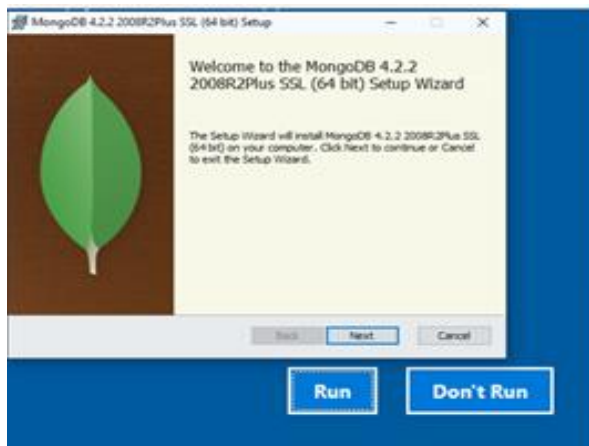
Download source cần cài đặt tại:

<https://www.mongodb.com/download-center#community>

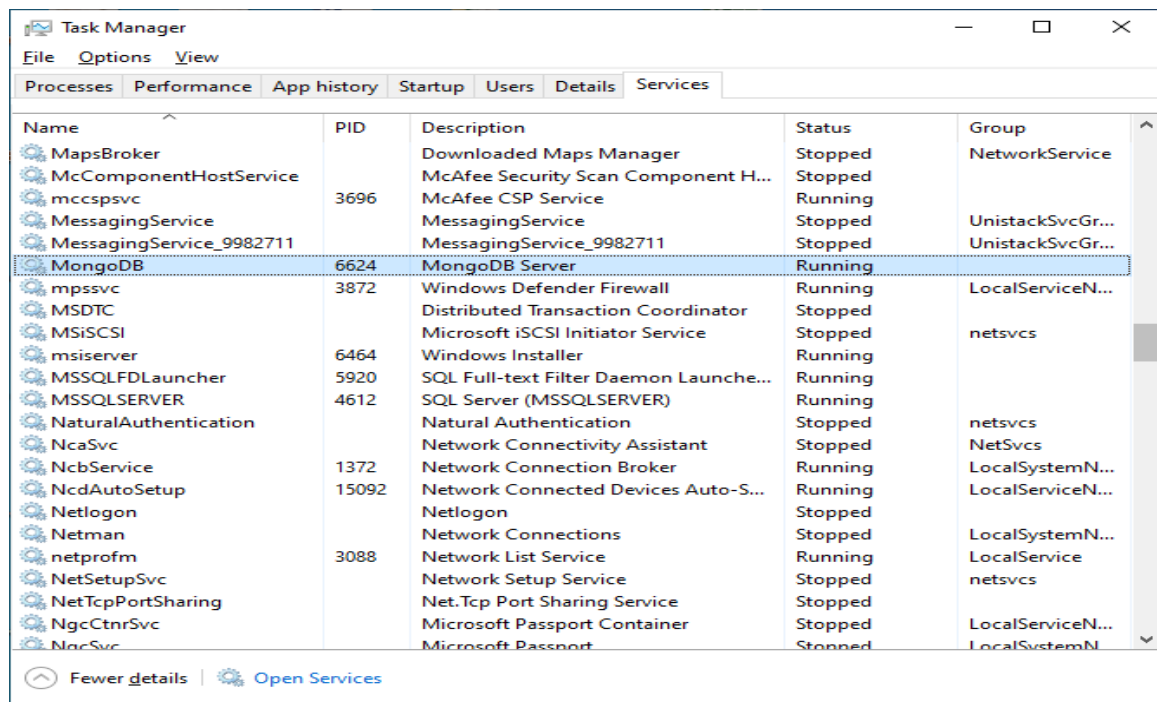
 mongodb-compass-1.20.4-win32-x64

 mongodb-win32-x86\_64-enterprise-windows-64-4.2.2-signed

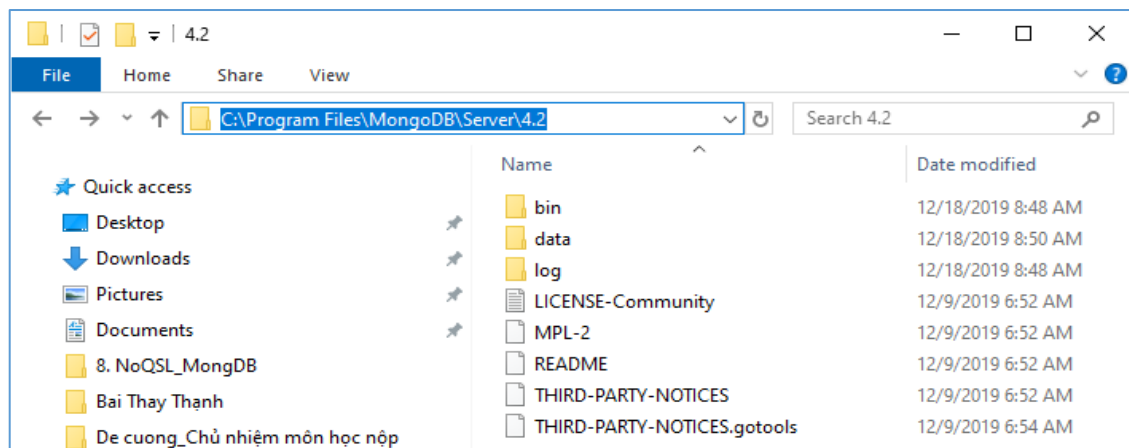
Sau khi download hai gói phần mềm trên thì tiến hành cài đặt MongoDB server enterprise theo các bước như sau:



Sau khi cài đặt xong, kiểm tra xem đã cài đặt thành công và hoạt động của MongoDB đã kích hoạt chưa, bằng cách xem:



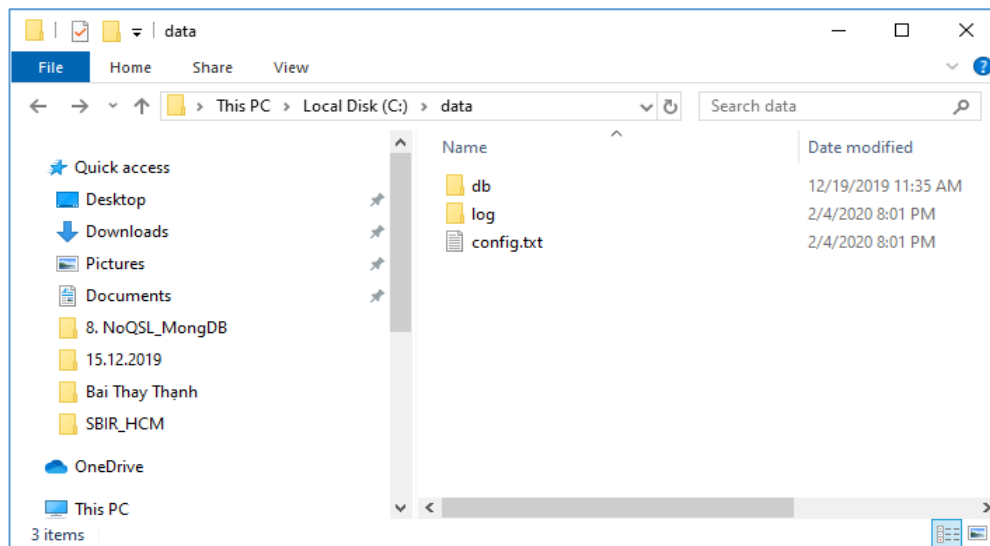
Sau khi cài đặt, xem thư mục:



Đến đây là đã cài xong MongoDB Services Enterprise.

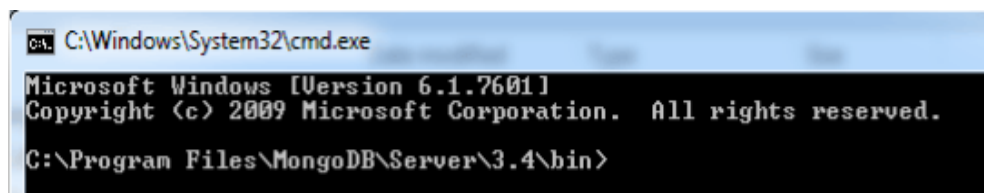
### 1.1.3. Cấu hình MongoDB

Sau khi đã cài đặt thành công MongoDB thì tạo tiếp cho mình một thư mục (ở đâu tùy ý) để lưu trữ data và log. Ở đây chọn ổ C.



### Khởi động MongoDB:

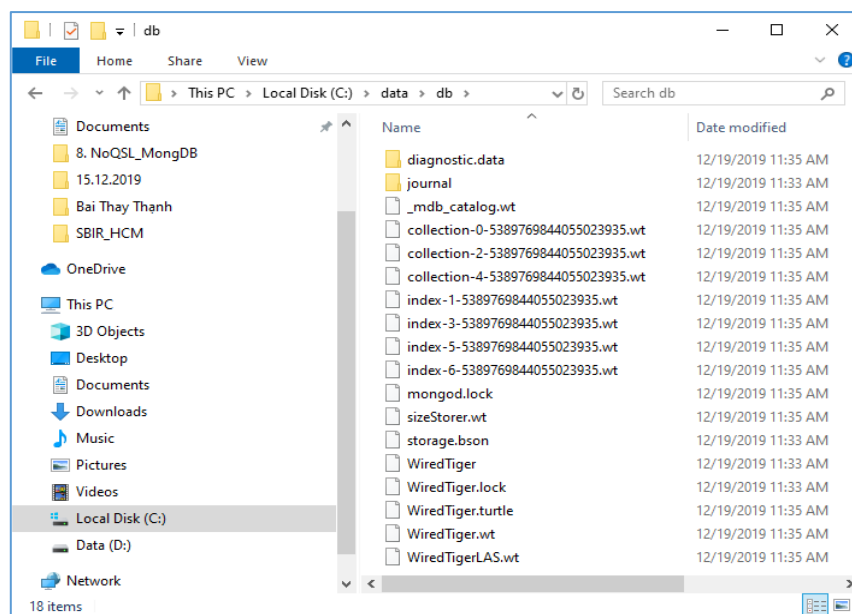
Tiếp đó để chạy MongoDB bằng cách mở cửa sổ cmd (Windows +R) và di chuyển đến thư mục cài đặt MongoDB và vào thư mục bin.



Chạy lệnh cấu hình MongoDB:

```
mongod --config "D:\data\config.txt"
```

Sau khi cấu hình, thử xem có một số file sinh ra như sau:



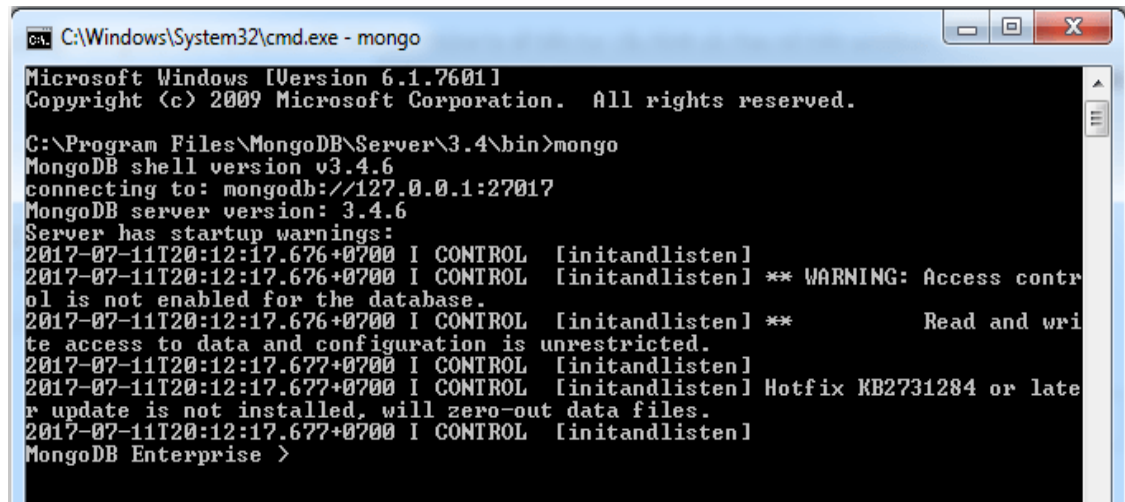


## Xem thông tin MongoDB:

Để xem thông tin của MongoDB đang chạy thì chạy tiếp 1 cửa sổ CMD nữa và chuyển đến thư mục bin của MongoDB như ở trên, thực thi lệnh:

```
mongo
```

Lúc này cửa sổ sẽ hiển thị ra các thông số của MongoDB đang chạy.

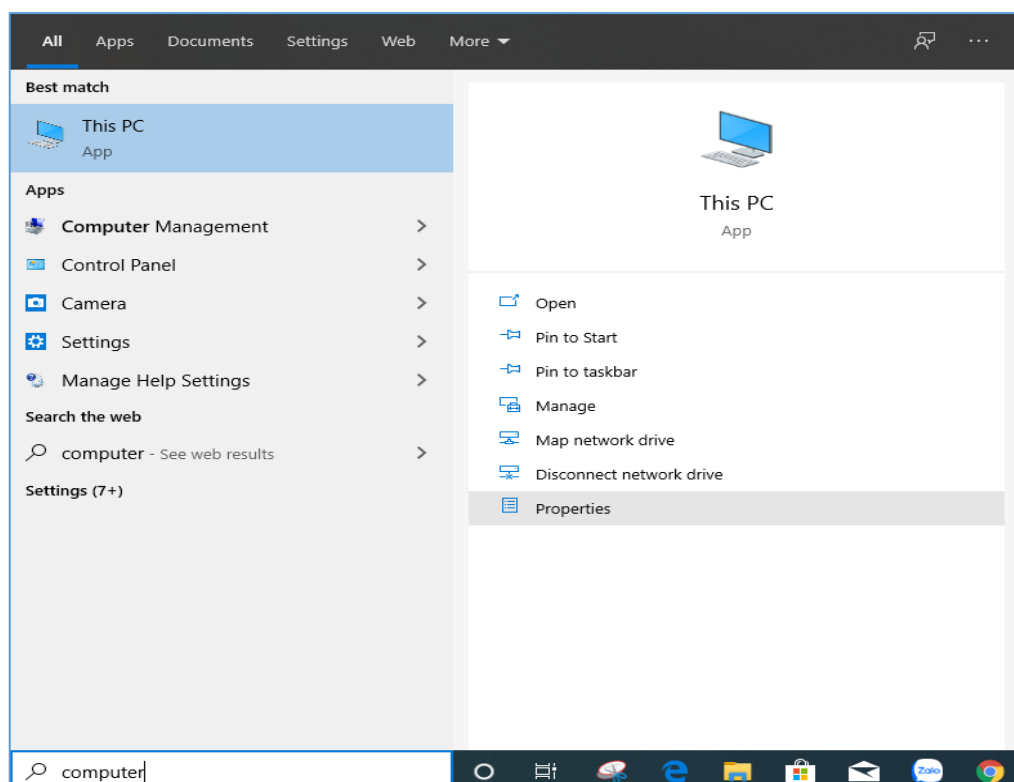


```
C:\Windows\System32\cmd.exe - mongo
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

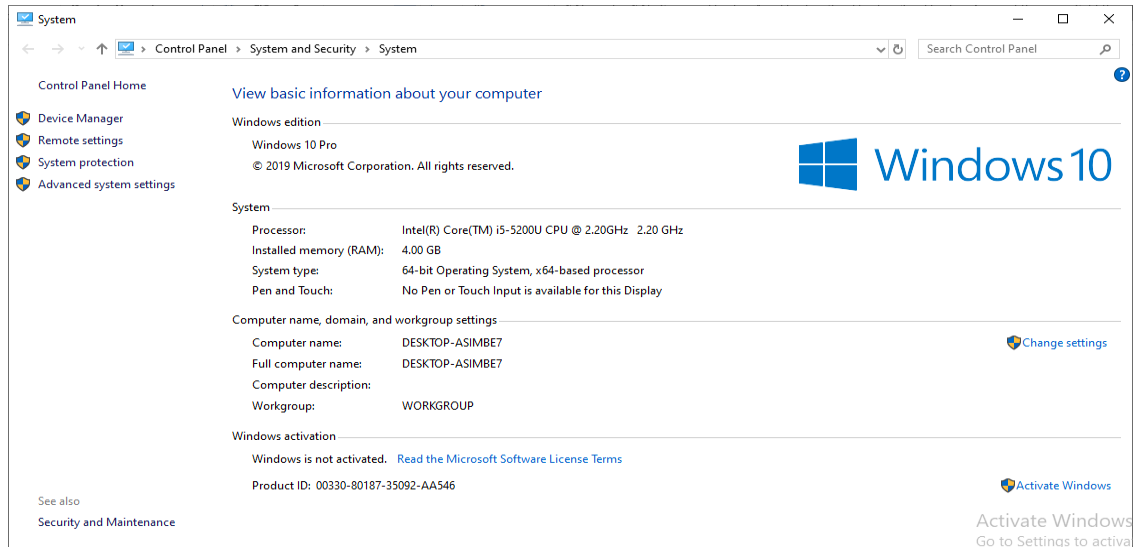
C:\Program Files\MongoDB\Server\3.4\bin>mongo
MongoDB shell version v3.4.6
connecting to: mongodb://127.0.0.1:27017
MongoDB server version: 3.4.6
Server has startup warnings:
2017-07-11T20:12:17.676+0700 I CONTROL [initandlisten]
2017-07-11T20:12:17.676+0700 I CONTROL [initandlisten] ** WARNING: Access control
ol is not enabled for the database.
2017-07-11T20:12:17.676+0700 I CONTROL [initandlisten] **          Read and wri
te access to data and configuration is unrestricted.
2017-07-11T20:12:17.677+0700 I CONTROL [initandlisten]
2017-07-11T20:12:17.677+0700 I CONTROL [initandlisten] Hotfix KB2731284 or late
r update is not installed, will zero-out data files.
2017-07-11T20:12:17.677+0700 I CONTROL [initandlisten]
MongoDB Enterprise >
```

## Cài đặt biến môi trường để thực thi MongoDB:

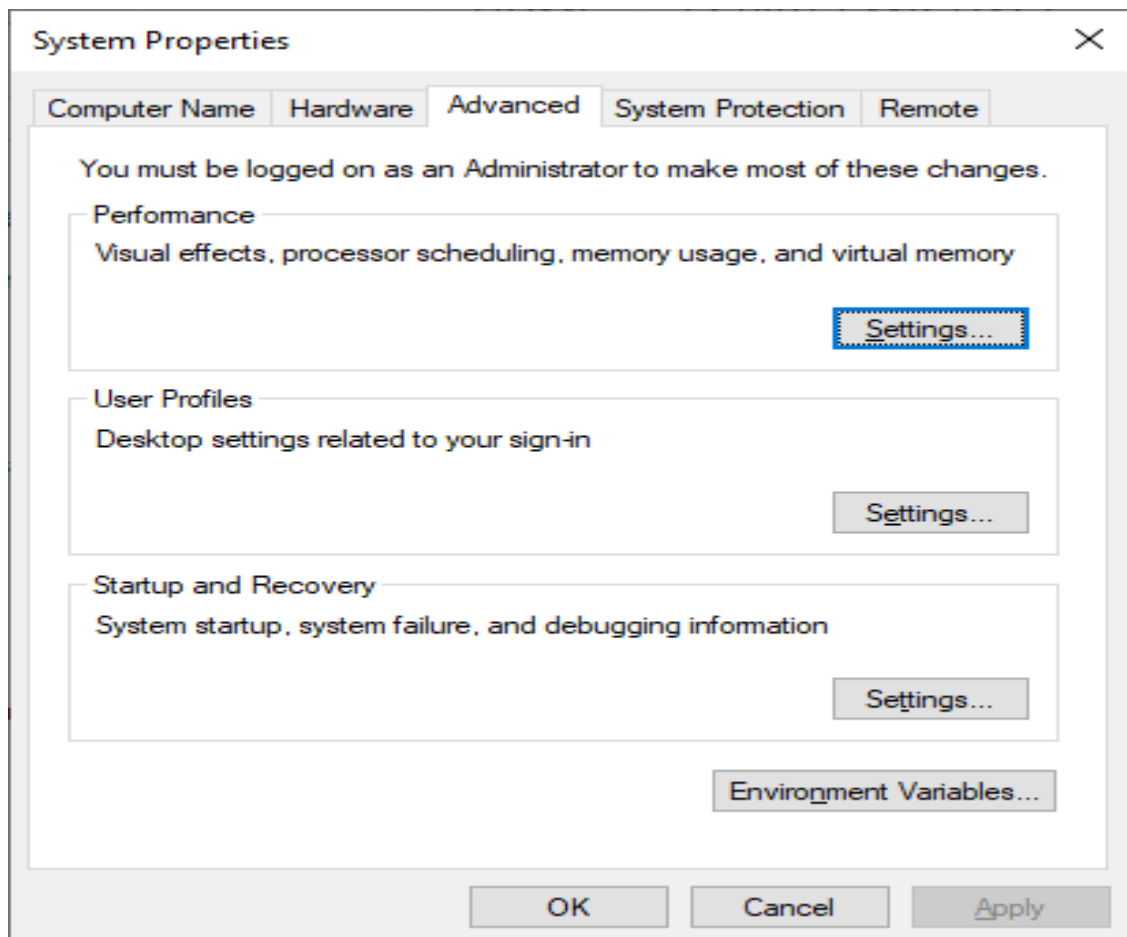
Nếu như việc sử dụng MongoDB như trên khá là bất tiện, thì có thể cài đặt biến môi trường để có thể sử dụng lệnh MongoDB ở bất kỳ thư mục nào trên windows mà không cần phải vào thư mục bin. Chọn vào biểu tượng windows (hoặc nhấn phím windows), nhấn chuột phải vào **computer** và chọn **properties**.



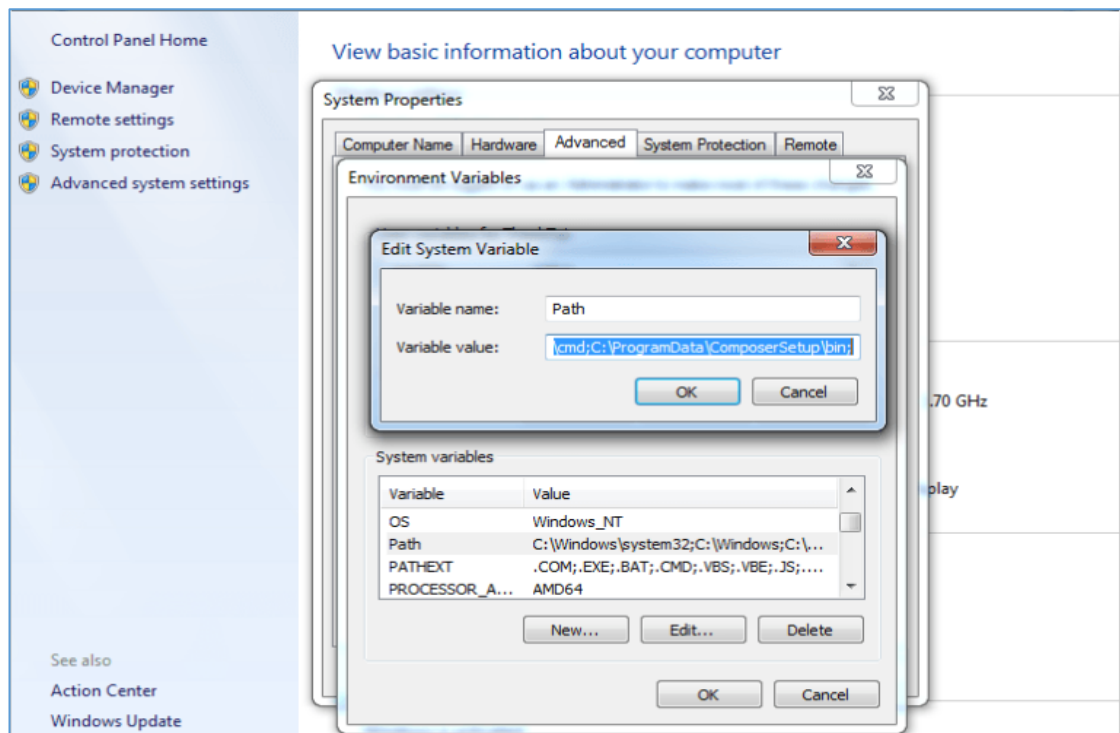
Vào Advanced system settings:



Vào tab Advanced > chọn Environment Variables:

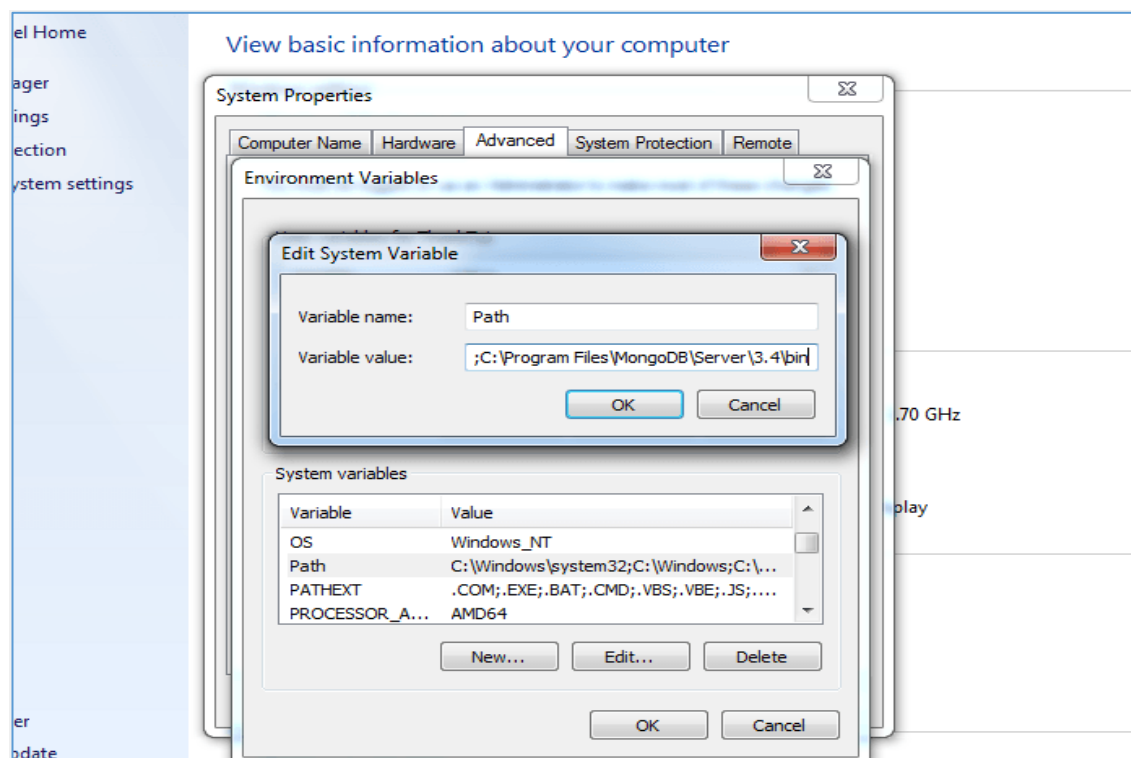


Tiếp tục có một cửa sổ mới hiện lên, ở cửa sổ **System variable** tìm đến Variable click vào **PATH**.



Ở ô Variable value, điền thêm đường dẫn tới thư mục bin vào

Chú ý: Phải có dấu ; ngăn giữa path trước đó và path sẽ thêm vào, chẳng hạn như C:\Program Files\MongoDB\Server\3.4\bin



Chọn OK để lưu lại và hoàn tất quá trình cấu hình. Sau khi đã cấu hình thành công biến môi trường, chỉ cần bật cửa sổ CMD lên ở bất kỳ đâu trên windows và gõ các lệnh của MongoDB mà không cần chuyển đến thư mục bin nữa.

#### 1.1.4. Một số công cụ hỗ trợ thao tác trên MongoDB

Hiện nay, có khá nhiều công cụ hỗ trợ giao diện thao tác thực hành MongoDB, mỗi công cụ có những ưu, nhược điểm riêng. Tùy theo mục đích và nhu cầu sử dụng của người dùng mà chúng ta có những lựa chọn công cụ phù hợp.

##### Một số công cụ thông dụng:

##### 1.1.4.1. NoQueryBooster

NoQueryBooster cho MongoDB (trước đây là MongoBooster) là một công cụ đa nền tảng tập trung phát triển cho MongoDB. Hiện nay đã có phiên bản miễn phí.

##### Tính năng, đặc điểm:

Ngôn ngữ tích hợp hiểu được tất cả các phương thức, thuộc tính, biến, từ khóa, thậm chí cả tên collection, tên trường và toán tử.

- Hỗ trợ SQL bao gồm các hàm, biểu thức, tổng hợp cho các bộ sưu tập với các đối tượng và mảng lồng nhau.
- Dịch các truy vấn MongoDB (find, aggregate hoặc SQL query) sang các ngôn ngữ đích khác nhau: MongoDB Shell, JavaScript (Node.js), Java, C # và Python.

Download: <https://nosqlbooster.com/downloads>

##### 1.1.4.2. Studio 3T

Studio 3T là một công cụ hữu ích khác để các lập trình viên làm việc với MongoDB. Công cụ này cho phép người dùng khám phá cơ sở dữ liệu cục bộ của họ hoặc làm việc với các phân đoạn và bộ bản sao.

##### Tính năng, đặc điểm:

- Tương thích hoàn toàn với các phiên bản hiện tại và phiên bản cũ của MongoDB.
- Hỗ trợ tìm kiếm kéo và thả tích hợp để tạo, tìm các truy vấn phức tạp.
- Kết nối an toàn cho các phiên bản MongoDB đơn và các bộ bản sao
- Sao chép và dán tài liệu trên các máy chủ và cơ sở dữ liệu
- Dễ dàng so sánh và đồng bộ hóa dữ liệu
- Quản lý người dùng MongoDB
- Tổng quan trực quan thời gian thực của các hoạt động máy chủ khác nhau.
- Download: <https://studio3t.com>

##### 1.1.4.3. Nucleon Database Master

Nucleon Database Master là một trong những công cụ quản trị cơ sở dữ liệu MongoDB mạnh mẽ và dễ sử dụng nhất. Nó đơn giản hóa việc quản lý, giám sát, truy vấn, chỉnh sửa, trực quan hóa NoQuery DBMS.

##### Tính năng, đặc điểm:

- Hỗ trợ Trình soạn thảo truy vấn JSON / LINQ / SQL.
- Nó cung cấp trình soạn thảo truy vấn SQL, LINQ và JSON mạnh mẽ và trực quan.
- Người dùng có thể xuất dữ liệu thành các định dạng tệp như XML, HTML, MS Office, CSV, OpenOffice, RTF, PDF, XPS, JSON, dBase và PNG.
- Cung cấp trình soạn thảo truy vấn C # Scripting động hỗ trợ Linq to MongoDB và Linq to Dataset.
- Cho phép nhập dữ liệu từ các tệp XML, CSV và SQL Script mà không giới hạn kích thước.
- Dowdload: <http://nucleonsoftware.com/doads>

#### 1.1.4.4. NoSQL Manager

Công cụ GUI MongoDB này hợp nhất sức mạnh UI và Shell 1 cách thân thiện. Cung cấp hiệu suất cao với sự hỗ trợ cho tất cả các tính năng mới nhất của MongoDB và MongoDB Enterprise. Tiết kiệm thời gian cho các nhà phát triển cơ sở dữ liệu và quản trị viên.

##### **Tính năng, đặc điểm:**

- GUI Shell MongoDB đầy đủ tính năng với tự động hoàn thành mã và tô sáng cú pháp.
- Trình chỉnh sửa đi kèm với ba chế độ xem Tree, Table và JSON
- Trình xem tài liệu dễ sử dụng
- Công cụ quản lý tệp để làm việc với GridFS
- Tùy chọn quản lý và xem đơn giản cho tất cả các loại đối tượng MongoDB
- Nhập bảng từ cơ sở dữ liệu MySQL và SQL Server
- Xuất tài liệu sang định dạng tệp CSV, XML, XLSX và JSON
- Dowdload: <https://www.mongodbmanager.com/d> Download

#### 1.1.4.5. Mongo Management Studio

Mongo Management Studio là một công cụ hiệu quả khác để quản lý MongoDB. Có thể thực thi tất cả các lệnh MongoDB thông thường mà không cần sử dụng MongoDB shell.

##### **Tính năng, đặc điểm:**

- Mongo Management Studio là nền tảng chéo, vì vậy nó chạy trên tất cả các hệ thống chính.
- Nó cung cấp hỗ trợ cho MongoDB 3.0 / 3.2 / 3.4
- Ứng dụng cho phép kết nối với cơ sở dữ liệu MongoDB từ xa bằng cách sử dụng SSH
- Với Mongo Management Studio, có thể đọc và ghi vào GridFS collections
- Nó cung cấp một tài liệu đầy đủ về tất cả các tính năng bằng cách giải thích các chủ đề liên quan đến MongoDB

- Với tính năng chỉnh sửa nội tuyến, có thể thực hiện thao tác dữ liệu nhanh chóng
- Dowdload: <http://mms.litixsoft.de>

#### 1.1.4.6. MongoJS Query Analyzer

MongoJS Query Analyzer là trình soạn thảo JavaScript MongoDB. Nó cho phép người dùng thực thi các lệnh JavaScript. Công cụ này cung cấp hỗ trợ để tự động hoàn thành và tô sáng cú pháp.

##### **Tính năng, đặc điểm:**

- Các câu lệnh và truy vấn JavaScript chạy trong giao diện dòng lệnh shell MongoDB.
- Trình phân tích truy vấn MongoJS cho phép người dùng xem kết quả theo phân cấp cây, lưới và dưới dạng kết quả văn bản.
- Tính năng Pretty Print JSON của MongoJS cho phép hiển thị kết quả JSON ở định dạng dễ đọc.
- Nó hiển thị kết quả truy vấn theo nhiều cách; như văn bản, lịch sử văn bản, lưới và lưới trực.
- Nội dung của Query Analyzer có thể được lưu theo các cách và định dạng khác nhau.
- Dowdload: <http://www.aquafold.com/aquadatastudio.html>

#### 1.1.4.7. Nosqlclient

Nosqlclient là một công cụ quản lý MongoDB hiệu quả khác. Đây là một cách dễ dàng để quản lý cơ sở dữ liệu MongoDB. Nó nhấn mạnh vào các yêu cầu của người dùng cuối và cho phép họ sử dụng toàn bộ khả năng của các trình duyệt hiện đại cũng như các ứng dụng trên máy tính.

##### **Tính năng, đặc điểm:**

- Miễn phí và là một công cụ nguồn mở
- Cập nhật, xóa và chèn dữ liệu mà không cần sử dụng truy vấn.
- RAM trực tiếp, CPU, đọc / ghi và biểu đồ hoạt động / xếp hàng
- Hỗ trợ LDAP, GSSAPI và Xs09
- Sử dụng JSON mở rộng thay cho BSON
- Tự động hoàn thành tên cho bộ sưu tập
- Khôi phục kết xuất để triển khai thử nghiệm MongoDB.
- Dowdload: <https://www.nosqlclient.com/>

#### 1.1.4.8. Cluster control

ClusterControl cung cấp bảo mật hoàn toàn tự động, duy trì tính toàn vẹn của cơ sở hạ tầng cơ sở dữ liệu. Trong công cụ MongoDB này, có thể triển khai và quản lý các công nghệ cơ sở dữ liệu nguồn mở khác nhau.

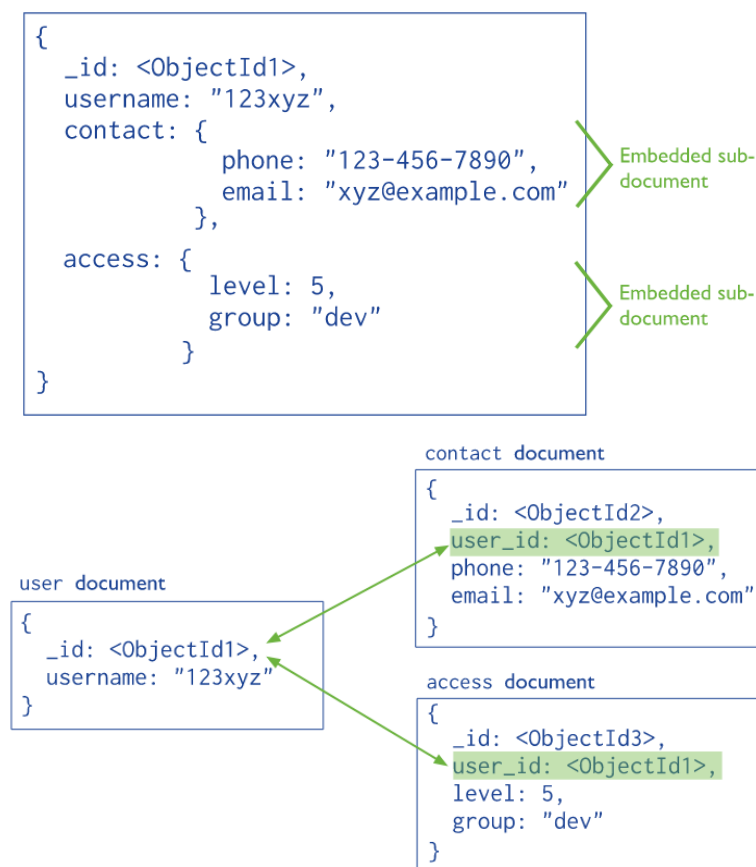
### Tính năng, đặc điểm:

- Cho phép người dùng tùy chỉnh các giải pháp cho việc triển khai MongoDB.
- Dễ dàng thêm và xóa các nút, thay đổi kích thước phiên bản và sao chép cụm sản xuất với sự trợ giúp của công cụ này.
- Nó cung cấp giao diện đơn để tự động hóa các môi trường cơ sở dữ liệu hỗn hợp MongoDB và MySQL.
- Nó cung cấp các tính năng quản lý, sửa chữa và phục hồi các nút bị hỏng, kiểm tra và tự động nâng cấp.
- Dowdload: [https://severalnines.com/product/clustercontrol/for\\_mongodb](https://severalnines.com/product/clustercontrol/for_mongodb)

## 1.2. CÁC MÔ HÌNH DỮ LIỆU

Hai mô hình dữ liệu thông dụng được sử dụng trong MongoDB là: mô hình dữ liệu nhúng (Embedded Data Models) và mô hình dữ liệu tham chiếu (References Data Models). Trong đó mô hình dữ liệu nhúng được sử dụng chủ yếu vì nó thể hiện được đặc trưng của cơ sở dữ liệu tài liệu. Trong khi mô hình dữ liệu tham chiếu có cách tổ chức dữ liệu tương tự như trong cơ sở dữ liệu quan hệ.

Ví dụ:



## 1.3. SAO LƯU VÀ PHỤC HỒI DỮ LIỆU

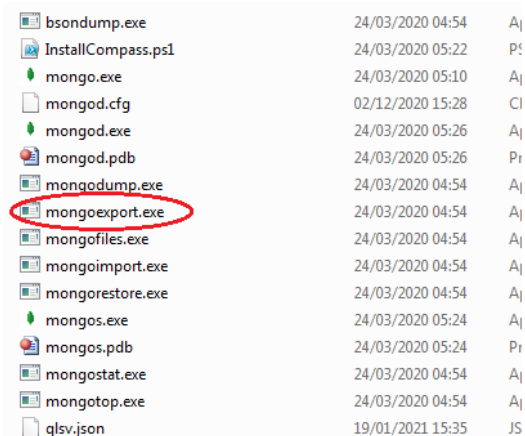
### 1.3.1. Export và Import collection

#### Export collection:

MongoDB cho phép xuất dữ liệu từng collection bằng lệnh Export ra file định dạng \*.Json hoặc \*.csv. Thực hiện export như sau:

- Mở cửa sổ *cmd.exe* tại vị trí thư mục bin có chứa các tập tin sau:

Ví dụ: C:\Program Files\MongoDB\Server\4.2\bin



bsondump.exe	24/03/2020 04:54	Aj
InstallCompass.ps1	24/03/2020 05:22	Pt
mongo.exe	24/03/2020 05:10	Aj
mongod.cfg	02/12/2020 15:28	Cl
mongod.exe	24/03/2020 05:26	Aj
mongod.pdb	24/03/2020 05:26	Pr
mongodump.exe	24/03/2020 04:54	Aj
<b>mongoexport.exe</b>	24/03/2020 04:54	Aj
mongofiles.exe	24/03/2020 04:54	Aj
mongoimport.exe	24/03/2020 04:54	Aj
mongorestore.exe	24/03/2020 04:54	Aj
mongos.exe	24/03/2020 05:24	Aj
mongos.pdb	24/03/2020 05:24	Pr
mongostat.exe	24/03/2020 04:54	Aj
mongotop.exe	24/03/2020 04:54	Aj
qlsv.json	19/01/2021 15:35	JS

- Gõ lệnh theo cú pháp:

```
mongoexport --db database_name --collection collection_name --out "file_name.json"
```

Có thể thay thế "--db" bằng "-d"; "--collection" bằng "-c"; "--out" bằng "-o"

Ví dụ: `mongoexport -d qlsv -c sv -o "D:\MONGO\EXPORT\qlsv\sv.json"`

#### Import collection:

MongoDB hỗ trợ import collection được lưu trữ trên các file có định dạng \*.Json hoặc \*.csv. Mỗi file chứa thông tin của một collection.

Thực hiện như sau:

- Mở cửa sổ lệnh cmd tương tự như khi Export
- Gõ lệnh theo cú pháp:

```
mongoimport --db database_name --collection collection_name --file "file_name.json"
```

Có thể thay thế "--db" bằng "-d"; "--collection" bằng "-c". Lưu ý "--file" không thể thay thế.



### 1.3.2. Sao lưu và phục hồi

#### Sao lưu dữ liệu:

- Mở cửa sổ lệnh cmd tương tự như khi Export
- Sử dụng cấu trúc lệnh:

```
mongodump -d <database name> -o <folder_database_backup>
```

Ví dụ: `mongodump -d qlsv -o "D:\MONGO\EXPORT\fulldb"`

#### Phục hồi dữ liệu:

- Mở cửa sổ lệnh cmd tương tự như khi sao lưu.
- Sử dụng cấu trúc lệnh:

```
mongorestore -d <database_name> <folder_database_backup_before>
```

Ví dụ: `mongorestore -d qlsv "D:\MONGO\EXPORT\fulldb\qlsv"`

## 1.4. CÁC THAO TÁC CẬP NHẬT DỮ LIỆU

### 1.4.1. Một số lệnh thao tác cơ bản

- Khởi động mongodb shell
  - > mongo
- Hiển thị danh sách database
  - > show dbs
- Hiển thị danh sách Collection
  - > show collections
- Hiển thị database hiện tại
  - > db
- Chuyển đến database chỉ định hoặc tạo database mới:
  - > use database\_name
- Xóa database hiện hành
  - > db.dropDatabase()
- Tạo mới một collection
  - > db.createCollection("collection name")
- Xóa Collection
  - > db.collection\_name.drop()

### 1.4.2. Cập nhật dữ liệu

#### 1.4.2.1. Thêm document

**Cú pháp:** `db.collection_name.insert(document)`

**Ví dụ:** Thêm một document vào collection sv

```
db.sv.insert({Masv:"sv001",Hoten:"Đỗ Thị Lan"})
```

```

db.sv.insert([ {Masv:"sv002",Hoten:"Trần Minh"}, {Masv:"sv003",Hoten:"Phạm
Tuấn"} ])
db.sv.insert(
{
    Masv:"sv004",
    Hoten:"Trần Minh Nghĩa",
    Ngaysinh: new Date("2001-04-14"),
    Lop:{Malop:"l01",Tenlop:"08DHTH"},
    Monhoc:
    [
        {Mamh:"m001",Tenmh:"Cơ sở dữ liệu",Sotc:3,Diem:7.5},
        {Mamh:"m002",Tenmh:"Toán cao cấp",Sotc:2,Diem:9},
        {Mamh:"m003",Tenmh:"Công nghệ phần mềm",Sotc:3,Diem:8.5},
    ]
})

```

#### 1.4.2.2. Xóa document

##### Cú pháp:

```
db.collection.remove({ criteria }) //Xóa Document thỏa điều kiện
```

##### Ví dụ:

```

> db.sv.remove ( {masv:"sv001"} )
> db.sv.deleteOne ( {Masv:"sv001"} )
> db.sv.deleteMany ( {Masv:"sv001"} )
> db.collection.remove({}) //Xóa tất cả các Document trong
collection
> db.collection.deleteMany({})

```

#### 1.4.2.3. Sửa document

##### – Sửa giá trị của một field:

**Cú pháp:** `db.collection.update({ criteria},{ $set:{key:"new value"} })`

##### Ví dụ:

```
> db.sv.update ( {Masv:"sv002"} , { $set: {Tuoi:24} })
```

```

Masv: 'sv002',
Hoten: 'Đỗ Thị Bình',
Tuoi: 24,

```

Sinh viên tự viết lệnh sửa: "Sửa Tên thành “Đỗ Thị Minh” và quốc tịch thành “Mỹ” của sinh viên có mã là sv002?"

##### – Sửa giá trị trong một mảng:

**Cú pháp:** `db.collection.update({ criteria},{ $set: {“array.position”:“new value”} })`

Ví dụ: sửa giá trị của mảng Ngoaingiu tại vị trí 1 có giá trị mới là “Tiếng Nhật”

```
> db.sv.update({Masv:"sv003"},{$set:{"Ngoaingiu.1":"Tiếng Nhật"}})
```

```
Phai: 'Nam',
Quoctich: 'Việt Nam',
Ngoaingiu: [ 'Tiếng Anh', 'Tiếng Nhật' ],
Lop: { Malop: '103', Tenlop: '09DHTH' },
Monhoc:
```

#### – Sửa giá trị của field trong document nhúng:

Cấu trúc: field:{subfield:subvalue,...}

Cú pháp: db.collection.update({criteria},{set:{"field.subfield":"new value"}})

Ví dụ: Sửa giá trị mã lớp và tên lớp của SV003 thành 103 và 09DHTH

```
> db.sv.update({Masv:"sv003"},{$set:{"Lop.Malop":"103","Lop.Tenlop":"09DHTH"}})
```

```
Phai: 'Nam',
Quoctich: 'Việt Nam',
Ngoaingiu: [ 'Tiếng Anh', 'Tiếng Nhật' ],
Lop: { Malop: '103', Tenlop: '09DHTH' },
Monhoc:
```

#### – Cập nhật giá trị trong mảng document con:

Cấu trúc field:[{subfield1:value1},{subfield2:value2},...]

Cú pháp: db.collection.update({criteria},{set:{"array.position.subfield":"new value"}})

Ví dụ: Sửa điểm môn học thứ nhất của sinh viên mã sv003 thành 8.5

```
> db.sv.update({Masv:"sv003"},{$set:{"Monhoc.1.Diem":8.5}})
```

```
Ngoaingiu: [ 'Tiếng Anh', 'Tiếng Nhật' ],
Lop: { Malop: '103', Tenlop: '09DHTH' },
Monhoc:
[ { Mamh: 'm005', Tenmh: 'Lịch sử đảng', Sotc: 3, Diem: 9 },
  { Mamh: 'm006', Tenmh: 'Toán rời rạc', Sotc: 3, Diem: 8.5 },
  { Mamh: 'm001', Tenmh: 'Cơ sở dữ liệu', Sotc: 3, Diem: 6 } ],
Malop: '103'
```

#### – Cập nhật sửa nhiều giá trị:

Sử dụng thuộc tính {multi:true}

Ví dụ: Sửa tên của những lớp có mã 101 thành 08DHTH2

```
> db.sv.update({"Lop.Malop":"101"},{$set:{"Lop.Tenlop":"08DHTH2"}},{multi:true})
```

– **Cập nhật thêm giá trị vào mảng:**

Cú pháp: `db.collection.update({criteria},{<toán_tử_thêm>:{key_array:"new value"}})`

**Toán tử \$push:** thêm không kiểm tra trùng

**Ví dụ:**

```
> db.sv.update({Masv:"sv001"},{$push:{Ngoaingu:"Tiếng Nga"}})
> db.sv.update({Masv:"sv003"},{$push:{Monhoc:{Mamh:"m001",Ten
  mh:"Cơ sở dữ liệu", Sotc:3, Diem:6}}})
```

**Toán tử \$addToSet:** thêm có kiểm tra trùng

**Ví dụ:**

```
> db.sv.update({Masv:"sv001"},{$addToSet: {Ngoaingu:"Tiếng
  Nga"}})
```

– **Cập nhật loại bỏ giá trị ra khỏi mảng:**

**Sử dụng toán tử \$pull:** Loại bỏ giá trị ra khỏi mảng

**Cú pháp:** `db.collection.update({criteria},{ $pull:{key_array: "value"}})`

**Ví dụ:**

Loại bỏ giá trị Tiếng Nga ra khỏi mảng ngoại ngữ của sinh viên có mã sv001

```
> db.sv.update({Masv:"sv001"},{$pull:{Ngoaingu:"Tiếng Nga"}})
```

Loại bỏ một môn học ra khỏi mảng các document môn học của sinh viên sv003

```
> db.sv.update({Masv:"sv003"},{$pull:{Monhoc:{Mamh:"m001",Ten
  mh:"Cơ sở dữ liệu", Sotc:3, Diem:6}}})
```

– **Thay thế document:**

**Cú pháp:** `db.collection_name.save({"_id":ObjectId(...), key1:value1, key2: value2})`

**Ví dụ:** Thay thế thông tin sinh viên có `_id:` "60165a20254da01228f2f015"

```
> db.sv.save({_id: ObjectId("60165a20254da01228f2f015"),
  masv: "sv018",
  hoten: "Trần Văn Minh",
  tuoi: 20,
  malop: "L44",
  ngaysinh: "2020-03-25"})
```

## 1.5. TRUY VẤN DỮ LIỆU

### 1.5.1. Sử dụng lệnh find

#### 1.5.1.1. Truy vấn không điều kiện

– Tìm tất cả các document:

```
> db.collection.find()
```

– Tìm 1 document

```
> db.collection.findOne()
```

– Hiện thị kết quả theo định dạng JSON

```
> db.collection.find().pretty()
```

#### 1.5.1.2. Truy vấn với điều kiện đơn giản

– Điều kiện bằng (Equality condition)

Sử dụng biểu thức {<field1>:<value1>,...}

Ví dụ:

Tìm các sinh viên có mã số là sv001

```
> db.sv.find({Masv:"sv001"})
```

Tìm những sinh viên phái *Nam* có quốc tịch Việt Nam:

```
> db.sv.find({Phai:"Nam",Quoctich:"Việt Nam"})
```

#### 1.5.1.3. Lọc các trường trong truy vấn

Cú pháp: db.collection.find({criteria},{field:value,...})

Trong đó: value=1 là hiển thị, value=0 là ẩn

Trường `_id` sẽ tự động hiển thị. Nếu muốn ẩn thì ghi `_id:0`

Ví dụ: Tìm những sinh viên phái Nữ, hiển thị Mã sv và Họ tên.

```
> db.sv.find({Phai:"Nữ"},{Masv:1,Hoten:1,_id:0})
```

#### 1.5.1.4. Toán tử \$in

Cú pháp: db.collection.find({field:{\$in:["value1","value2",...]}})

Ví dụ: Tìm những sinh viên có quốc tịch thuộc tập các quốc tịch gồm: Việt Nam, Singapore:

```
> db.sv.find({quoctich:{$in:["Việt Nam","Singapore"]}})
```

#### 1.5.1.5. Tìm kiếm tương đối

**Cú pháp:**

{field:/^s/}: field có giá trị bắt đầu bằng chuỗi s

{field:/s\$/}: field có giá trị kết thúc bằng chuỗi s

{field:/s/}: field có giá trị chứa chuỗi s. Lưu ý: Chuỗi s không đặt trong cặp dấu nháy “”

### Ví dụ:

- Tìm những sinh viên có họ *Trần*

```
> db.sv.find({Hoten:/^Trần/})
```

- Tìm những sinh viên có tên *Bình*

```
> db.sv.find({Hoten:/Bình$/})
```

- Tìm những sinh viên có tên lót là *Thị*

```
> db.sv.find({Hoten:/Thị/})
```

#### 1.5.1.6. Các toán tử so sánh

Cú pháp: {field:{<operator>:<value>}}

Trong đó các toán tử (operator):

\$eq: equal to (=)

\$lt: less than (<),

\$lte: less than or equal to(≤),

\$gt: greater than (>),

\$gte: greater than or equal to (≥)

\$ne: not equal (!=)

Ví dụ: Tìm những sinh viên có tuổi < 23

```
> db.sv.find({Tuoi:{$lt:23}})
```

#### 1.5.1.7. Điều kiện kết hợp

- Toán tử \$and:

Cú pháp: {\$and:[{criteria1}, {criteria2}]}

Ví dụ1: Tìm những sinh viên quốc tịch Việt Nam có tuổi >22

```
> db.sv.find({$and:[{Quoctich:"ViệtNam"},{Tuoi:{$gt:22}}]})
```

```
> db.sv.find({Quoctich:"ViệtNam",Tuoi:{$gt:22}})
```

Ví dụ 2: Tìm những sinh viên có tuổi từ 18 đến 23

```
> db.sv.find({$and:[{Tuoi:{$gte:18,$lte:22}}]})
```

- Toán tử \$or:

Cú pháp: {\$or:[{criteria1}, {criteria2}]}

Ví dụ1: Tìm những sinh viên có tuổi <22 hoặc >25

```
> db.sv.find({$or:[{Tuoi:{$lt:22}}, {Tuoi:{$gt:25}}]})
```

#### 1.5.1.8. Thao tác trên mảng và document con

##### – Điều kiện từ mảng các giá trị:

Document: {field:[value1,value2,...]}

Cú pháp: {field:value}

Ví dụ: Tìm những sinh viên có ngoại ngữ là “Tiếng Nga”

```
> db.sv.find({Ngoaingu:"Tiếng Nga"})
```

##### – Điều kiện từ document con:

Document: {field:{subfield:value}}

Cú pháp: {"field.subfield":value1}

Ví dụ: Tìm những sinh viên học lớp có mã lớp là 101

```
> db.sv.find({"Lop.Malop":"101"})
```

##### – Điều kiện từ mảng các document con:

Document: {field:[{field1:value1},...]}

Cú pháp: {field:{\$elemMatch:{field1:value1}}}

Ví dụ: Tìm những sinh viên học môn học có tên là *Cơ sở dữ liệu*

```
> db.sv.find({Monhoc:{$elemMatch:{Tenmh:"Cơ sở dữ liệu"}}})
```

##### – Lệnh limit và skip:

- Lệnh *limit(n)*: trả về n document đầu tiên trong kết quả truy vấn.
- Lệnh *skip(m)*: bỏ qua m document đầu tiên trong kết quả truy vấn

Ví dụ:

```
> db.sv.find().limit(2)//trả về 2 document đầu tiên
> db.sv.find().skip(2)//bỏ qua 2 document đầu tiên
> db.sv.find().limit(2).skip(2)//bỏ qua 2 document đầu, lấy 2
document tiếp theo.
```

#### Sắp xếp dữ liệu

Cú pháp: db.collection.find().sort({field1:value1,field2: value2,...})

Value=1: sắp tăng, value=-1: sắp giảm

Ví dụ:

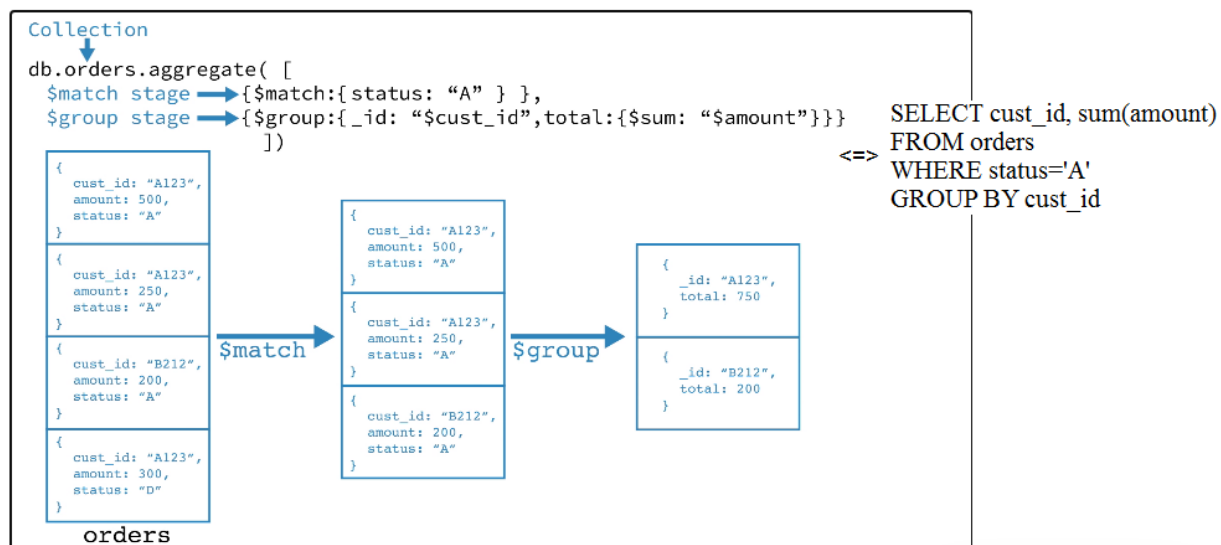
```
> db.sv.find().sort({Tuoi:1})
> db.sv.find().sort({"Lop.Malop":1,Tuoi: -1})
```

### 1.5.2. Truy vấn dữ liệu với Aggregation framework

Aggregation framework hỗ trợ các truy vấn phức tạp, có thể thực hiện các truy vấn với việc nhóm dữ liệu như trong SQL như: Group by, Count, sum,...

Cú pháp chung: `db.collection.aggregate(options)`

Ví dụ:



#### Các biểu thức sử dụng trong gom nhóm:

Biểu thức	Mô tả
\$sum	Tổng giá trị được xác định từ tất cả Document trong Collection đó
\$avg	Tính trung bình của tất cả giá trị đã cho từ tất cả Document trong Collection đó
\$min	Lấy giá trị nhỏ nhất của các giá trị từ tất cả Document trong Collection đó
\$max	Lấy giá trị lớn nhất của các giá trị từ tất cả Document trong Collection đó
\$push	Chèn giá trị vào trong một mảng trong Document kết quả
\$addToSet	Chèn giá trị tới một mảng trong Document kết quả, nhưng không tạo các bản sao
\$first	Lấy Document đầu tiên từ Source Document theo nhóm
\$last	Lấy Document cuối cùng từ Source Document theo nhóm

**Ví dụ 1:** Cho cơ sở dữ liệu quản lý nhân viên với cấu trúc collection như sau:

Collection Nhanvien

{

Manv: Chuỗi ký tự mã nhân viên,



Hoten: Chuỗi ký tự họ và tên

Phai: Phái nam hoặc nữ

Luong: Giá trị số thể hiện lương của nhân viên

Phòng ban: Chuỗi ký tự tên phòng ban

Ngoaingu: [mảng các giá trị chuỗi (ví dụ: Tiếng Anh, Tiếng Nhật,...)]

}

Viết các câu truy vấn sau:

1/ Liệt kê tên phòng và tổng lương của các nhân viên phái nam trong từng phòng.

```
> db.nv.aggregate({$group:{_id:"$Phongban", TongLuong:{$sum:"$Luong"}}})
```

2/ Liệt kê tên phòng và số nhân viên trong từng phòng

```
> db.nv.aggregate({$group:{_id:"$Phongban", TongLuong:{$sum:1}}})
```

3/ Cho biết lương trung bình theo phái

```
> db.nv.aggregate({$group:{_id:"$Phai", LuongTB:{$avg:"$Luong"}}})
```

4/ Cho biết tên phòng và lương cao nhất trong từng phòng

```
> db.nv.aggregate({$group:{_id:"$Phongban", MaxLuong:{$max:"$Luong"}}})
```

**Ví dụ 2: Sử dụng cơ sở dữ liệu quản lý sinh viên, viết các câu truy vấn sau:**

1/ Cho biết mã lớp, tên lớp và số sinh viên trong từng lớp

```
> db.sv.aggregate({$group:{_id:"$Lop", Sosv: {$sum:1}}})
```

2/ Cho biết tên lớp và số sinh viên trong từng lớp

```
> db.sv.aggregate({$group:{_id:"$Lop.Tenlop", Sosv:{$sum:1}}})
```

Lưu ý: Trường hợp lấy nhiều thuộc tính trong 1 document: `_id:{Malop:"$Lop.Malop",Tenlop:"$Lop.Tenlop"}`

3/ Cho biết Mã sinh viên, họ tên và số môn học của từng sinh viên

```
> db.sv.aggregate({$project:{_id:0,Masv:1, Hoten:1, Somh:{$size:"$Monhoc"}}})
```

*Lưu ý: Toán tử \$size không dùng trong \$group mà dùng trong \$project*

4/ Cho biết Mã sinh viên, họ tên và điểm trung bình của từng sinh viên

```
> db.sv.aggregate({$project:{_id:0,Masv:1, Hoten:1, DiemTB:{$avg:"$Monhoc.Diem"}}})
```

5/ Cho biết Mã lớp, tên lớp và số sinh viên của những lớp có số sinh viên  $\geq 2$

```
> db.sv.aggregate([{$group: {_id: "$Lop", Sosv: {
  $sum: 1}}}, {$match: {Sosv: {$gte: 2}}}]])
```

6/ Cho biết mã sv, họ tên và số môn học của những sinh viên học từ 2 môn trở lên

```
> db.sv.aggregate([{$project: {_id: 0, Masv: 1,
  Hoten: 1, Somh: {$size: "$Monhoc"}}}, {$match: {
  Somh: {$gte: 3}}}]])
```

### Sử dụng \$addToSet và \$push

Toán tử \$addToSet và \$push được dùng để đưa các giá trị vào mảng trong kết quả truy vấn khi gom nhóm với toán tử \$group

Ví dụ 1: Cho cơ sở dữ liệu với collection products như bên dưới, viết truy vấn lọc dữ liệu group theo manufacturer, đưa category vào mảng và tính giá trung bình của sản phẩm theo manufacturer.

	_id ObjectId	name String	manufacturer String	category String	price Mi
1	602cf95dac809405bc9b860c	"iPad 16GB Wifi"	"Apple"	"Tablets"	499
2	602cf95dac809405bc9b860d	"iPad 32GB Wifi"	"Apple"	"Tablets"	599
3	602cf95dac809405bc9b860e	"iPad 64GB Wifi"	"Apple"	"Tablets"	699
4	602cf95dac809405bc9b860f	"Galaxy S3"	"Samsung"	"Cell Phones"	563.99
5	602cf95dac809405bc9b8610	"Galaxy Tab 10"	"Samsung"	"Tablets"	450.99
6	602cf95dac809405bc9b8611	"Vaio"	"Sony"	"Laptops"	499
7	602cf95dac809405bc9b8612	"Macbook Air 13inch"	"Apple"	"Laptops"	499
8	602cf95dac809405bc9b8613	"Nexus 7"	"Google"	"Tablets"	199
9	602cf95dac809405bc9b8614	"Kindle Paper White"	"Amazon"	"Tablets"	129
10	602cf95dac809405bc9b8615	"Kindle Fire"	"Amazon"	"Tablets"	199

```
> db.products.aggregate({$group: { _id: "$manufacturer",
  categories: {$addToSet: "$category"},
  avg_price: {$avg: "$price"}}})
```

Kết quả:

```
{ _id: 'Apple',
  categories: [ 'Laptops', 'Tablets' ],
  avg_price: 574 },
{ _id: 'Samsung',
  categories: [ 'Cell Phones', 'Tablets' ],
  avg_price: 507.49 },
{ _id: 'Google', categories: [ 'Tablets' ], avg_price: 199 },
{ _id: 'Amazon', categories: [ 'Tablets' ], avg_price: 164 },
{ _id: 'Sony', categories: [ 'Laptops' ], avg_price: 499 }
```

Ví dụ 2: Lọc dữ liệu group theo manufacturer, đưa category vào mảng (không loại bỏ giá trị trùng), hiển thị giá cao nhất, thấp nhất của sản phẩm theo manufacturer.

```
> db.products.aggregate({$group:{
  _id:"$manufacturer",    categories:{$push:"$category"},
  max_price:{$max:"$price"},
  min_price:{$min:"$price"}}})
```

```
[ { _id: 'Google',
  categories: [ 'Tablets' ],
  max_price: 199,
  min_price: 199 },
  { _id: 'Apple',
  categories: [ 'Tablets', 'Tablets', 'Tablets', 'Laptops' ],
  max_price: 699,
  min_price: 499 },
  { _id: 'Sony',
```

## Sử dụng \$toLower, \$toUpper, \$multiply

\$toLower: hiển thị chữ in thường

\$toUpper: hiển thị chữ in hoa

\$multiply: Phép nhân

Ví dụ: Hiển thị dữ liệu với tên các manufacture phải là ký tự in hoa hay ký tự không in hoa có đơn giá nhân lên 10.

```
> db.products.aggregate([
  {$project:{
    _id:0,
    makerLower: {$toLower:"$manufacturer"},
    makerUpper: {$toUpper:"$manufacturer"},
    details: {
      category: "$category",
      price : {"$multiply":["$price",10]}
    },
    item:"$name"
  }}])
```

Kết quả:

```
[ { makerLower: 'apple',
  makerUpper: 'APPLE',
  details: { category: 'Tablets', price: 4990 },
  item: 'iPad 16GB Wifi' },
  { makerLower: 'apple',
  makerUpper: 'APPLE',
  details: { category: 'Tablets', price: 5990 },
  item: 'iPad 32GB Wifi' },
  { makerLower: 'apple',
```

## Sử dụng \$sort trong Aggregation

Sắp tăng dần: 1, giảm dần: -1

Ví dụ: sắp xếp document theo giá tăng dần

```
> db.products.aggregate({$sort:{price:1}})
```

```
[ { _id: ObjectId("602cf95dac809405bc9b8614"),
  name: 'Kindle Paper White',
  category: 'Tablets',
  manufacturer: 'Amazon',
  price: 129 },
  { _id: ObjectId("602cf95dac809405bc9b8613"),
  name: 'Nexus 7',
  category: 'Tablets',
  manufacturer: 'Google',
  price: 199 },
  { _id: ObjectId("602cf95dac809405bc9b8615")
```

### **\$skip và \$limit trong Aggregation**

\$skip để bỏ qua document, \$limit để giới hạn số document

Ví dụ: Bỏ qua 2 document đầu, lấy 3 document tiếp theo

```
> db.products.aggregate([ {$skip:2}, {$limit: 3}])
```

### **\$first và \$last trong Aggregation**

Toán tử \$first để lấy ra document đầu tiên trong nhóm, \$last lấy ra document cuối cùng trong nhóm.

Ví dụ: Xét collection sales được insert dữ liệu như sau:

```
db.sales.insert({ "_id" : 1, "item" : "abc", "price" : 10, "quantity" : 2, "date" : ISODate("2014-01-01T08:00:00Z") })
db.sales.insert({ "_id" : 2, "item" : "jkl", "price" : 20, "quantity" : 1, "date" : ISODate("2014-02-03T09:00:00Z") })
db.sales.insert({ "_id" : 3, "item" : "xyz", "price" : 5, "quantity" : 5, "date" : ISODate("2014-02-03T09:05:00Z") })
db.sales.insert({ "_id" : 4, "item" : "abc", "price" : 10, "quantity" : 10, "date" : ISODate("2014-02-15T08:00:00Z") })
db.sales.insert({ "_id" : 5, "item" : "xyz", "price" : 5, "quantity" : 10, "date" : ISODate("2014-02-15T09:05:00Z") })
db.sales.insert({ "_id" : 6, "item" : "xyz", "price" : 5, "quantity" : 5, "date" : ISODate("2014-02-15T12:05:10Z") })
db.sales.insert({ "_id" : 7, "item" : "xyz", "price" : 5, "quantity" : 10, "date" : ISODate("2014-02-15T14:12:12Z") })
```

Để lấy ra những sản phẩm được bán đầu tiên, thực hiện như sau:

```
> db.sales.aggregate({ $group:{_id: "$item", firstSalesDate:
{ $first: "$date" }}})
```

Kết quả:

```
[ { _id: 'xyz', firstSalesDate: 2014-02-03T09:05:00.000Z },
  { _id: 'jkl', firstSalesDate: 2014-02-03T09:00:00.000Z },
  { _id: 'abc', firstSalesDate: 2014-01-01T08:00:00.000Z } ]
```

## Sử dụng \$unwind

\$unwind: Được sử dụng để loại bỏ các phần tử mảng ra và đặt từng phần tử vào Document.

Ví dụ 1: Xét collection inventory có nội dung như sau:

```
{ "_id" : 1, "item" : "ABC1", "sizes" : [ "S", "M", "L" ] }
```

Để hiển thị ra các item theo từng sizes một (tách mảng sizes ra từng bản ghi riêng rẽ), sẽ dùng \$unwind

```
> db.inventory.aggregate( [ { $unwind : "$sizes" } ] )
```

Kết quả sẽ ra 3 item cho từng size như sau:

```
{ "_id" : 1, "item" : "ABC1", "sizes" : "S" }
```

```
{ "_id" : 1, "item" : "ABC1", "sizes" : "M" }
```

```
{ "_id" : 1, "item" : "ABC1", "sizes" : "L" }
```

Ví dụ 2: Sử dụng csdl sinh viên, cho biết có bao nhiêu sinh viên của từng ngoại ngữ.

```
> db.sv.aggregate([{$unwind:"$Ngoaingu"},
  {$group: {_id:"$Ngoaingu", Sosv:{$sum:1}}}] )
```

Kết quả:

```
[ { _id: 'Tiếng Nga', Sosv: 2 },
  { _id: 'Tiếng Nhật', Sosv: 1 },
  { _id: 'Tiếng Pháp', Sosv: 1 },
  { _id: 'Tiếng Anh', Sosv: 3 } ]
```

## 1.6. BÀI TẬP THỰC HÀNH CHƯƠNG 1

**Bài 1:** MongoDB Server Enterprise và MongoDB Compass đã cài đặt trên các máy. Sinh viên tiến hành thiết lập biến môi trường cho máy tính mình đang ngồi. Khởi động cửa sổ lệnh cmd hoặc Windows PowerShell thao tác kiểm tra kết quả.

**Bài 2:** Sử dụng lệnh để tạo một Database có tên QLSinhVien gồm collection sv theo cấu trúc:

```
{
  Masv: Mã số sinh viên,
  Hoten: Họ và tên sinh viên,
  Ngaysinh: ngày sinh (dữ liệu date)
  Phai: Giới tính
  Email: [Mảng các email]
  Lop: { Malop: Mã lớp, Tenlop: Tên lớp },
```

Monhoc:

```
[ {  
  Mamh: Mã môn học,  
  Tenmh: Tên môn học,  
  Sotc: Số tín chỉ,  
  Diem: Điểm thi  
  }, {...}  
]
```

**Thực hiện các yêu cầu sau:**

- a/ Viết lệnh thêm vào collection *sinhvien* trong 2 trường hợp: thêm một và nhiều document.
- b/ Viết lệnh xóa document với điều kiện mã sinh viên là “sv005”
- c/ Viết lệnh xóa những sinh viên học lớp có mã lớp là 103
- d/ Viết lệnh sửa Họ tên của sinh viên có mã sv001 thành Đỗ Nhật Lâm
- e/ Sửa tuổi thành 25, Phái thành Nữ, Họ tên thành Trần Thị Lan cho sinh viên có mã là sv003
- f/ Sửa ngoại ngữ thứ 2 của sinh viên có mã sv003 thành Tiếng Hàn
- g/ Sửa điểm của sinh viên có mã sv003 học môn thứ 1 thành 9
- h/ Viết lệnh thay thế một document với `_id` được chỉ định.

**Bài 3:** Dùng lệnh tạo cơ sở dữ liệu quản lý bán sản phẩm với collection *hoadon* có cấu trúc như sau:

```
Collection: Hóa đơn  
{  
  Mã hóa đơn: Chuỗi lưu mã số hóa đơn,  
  Ngày lập: Ngày lập hóa đơn,  
  Khách hàng: {Mã khách hàng, Tên khách hàng, Địa chỉ}  
  Sản phẩm bán: [  
    {Mã sp1, Tên sp1, số lượng, giá bán, thành tiền},  
    {Mã sp2, Tên sp2, số lượng, giá bán, thành tiền}  
    ...  
  ]  
}
```

**Thực hiện các yêu cầu sau:**

- a/ Viết lệnh thêm vào 2 document vào collection *hoadon*.
- b/ Viết lệnh xóa những hóa đơn có ngày lập “2020-03-25”
- c/ Viết lệnh xóa những hóa đơn của khách hàng có mã là “kh001”
- d/ Viết lệnh sửa ngày lập của hóa đơn có mã h001 thành “2021-02-25”

e/ Sửa thông tin khách hàng có mã là kh001 với Họ tên khách hàng thành Trần Thị Lan, địa chỉ thành TPHCM

f/ Sửa tên khách hàng có mã kh003 thành Đỗ Thanh Bình

g/ Thêm một sản phẩm vào hóa đơn có mã hd003

**Bài 4:** Thực hiện lại bài 2 và bài 3 trên giao diện MongoDB Compass

**Bài 5:** Sử dụng cơ sở dữ liệu quản lý sinh viên ở bài tập 2. Viết các câu truy vấn sau:

- 1/ Cho biết những sinh viên phái nữ có ngoại ngữ là Tiếng Anh
- 2/ Liệt kê những sinh viên phái nam trên 22 tuổi
- 3/ Liệt kê những sinh viên có họ tên bắt đầu bằng chữ T
- 4/ Liệt kê những sinh viên có tên Lan, chỉ hiển thị Mã sinh viên, Họ tên và Phái.
- 5/ Tìm những sinh viên học các ngoại ngữ thuộc gồm: Tiếng Pháp, Tiếng Nhật
- 6/ Liệt kê các sinh viên của 2 lớp có tên là 08DHTH và 09DHTH, hiển thị mã sinh viên và họ tên.
- 7/ Liệt kê những sinh viên học lớp 09DHTH có tuổi <23 hoặc >25
- 8/ Tìm những sinh viên lớp 09DHTH có ngoại ngữ Tiếng Pháp hoặc Tiếng Nhật
- 9/ Những sinh viên học môn cơ sở dữ liệu có điểm >7.5

**Bài 6:** Sử dụng cơ sở dữ liệu quản lý bán sản phẩm ở bài 3, viết các câu truy vấn sau:

- 1/ Cho biết thông tin những hóa đơn được lập ngày 12/03/2021
- 2/ Liệt kê Mã hóa đơn, ngày lập của khách hàng có mã số là kh001
- 3/ Liệt kê những hóa đơn có bán sản phẩm có mã là sp012 với số lượng >30
- 4/ Những hóa đơn nào được lập trong thời gian từ ngày 01/03/2020 đến 30/05/2021
- 5/ Liệt kê thông tin những hóa đơn không lập trong ngày 5/7/2020. Thông tin liệt kê gồm: Mã hóa đơn, ngày lập.

**Bài 7:** Sử dụng Aggregation Framework, và cơ sở dữ liệu quản lý bán sản phẩm, viết lệnh thực hiện các truy vấn sau:

- 1/ Cho biết mã khách hàng và số lượng hóa đơn của từng khách hàng
- 2/ Cho biết mã khách hàng, tên khách hàng và số lượng hóa đơn của từng khách hàng.
- 3/ Liệt kê mã hóa đơn, ngày lập và số lượng mặt hàng có trong hóa đơn tương ứng.
- 4/ Hiển thị ngày lập và số lượng hóa đơn được lập tương ứng của khách hàng Trần Minh Tuấn
- 5/ Hiển thị mã và tên những khách hàng có địa chỉ ở TPHCM
- 6/ Hiển thị thông tin những hóa đơn được lập từ ngày 12/03/2021 đến 20/03/2021
- 7/ Hiển thị mã hóa đơn, ngày lập của khách hàng tên là Trần Minh Tuấn, sắp xếp tăng dần theo ngày lập.
- 8/ Liệt kê mã và tên những khách hàng có trên 2 hóa đơn.
- 9/ Cho biết Mã sản phẩm và giá bán trung bình của từng sản phẩm (HD: Dùng unwind)

10/ Hiển thị Mã khách hàng, tên khách hàng, và danh sách các mã hóa đơn của khách hàng tương ứng được đưa vào mảng (HD: sử dụng \$AddToSet)

11/ Hiển thị ngày lập và danh sách các mã hóa đơn (đưa vào mảng) của từng ngày lập tương ứng.

12/ Hiển thị mã sản phẩm, tên sản phẩm và tổng số lượng bán ra tương ứng.

13/ Liệt kê mã hóa đơn, tên khách hàng và trị giá của hóa đơn tương ứng (biết rằng trị giá hóa đơn = sum(số lượng \* đơn giá)

14/ Liệt kê mã những hóa đơn được lập vào tháng 7 (HD: sử dụng toán tử \$month)

15/ Liệt kê mã những hóa đơn được lập vào năm 2020 (HD: sử dụng toán tử \$year)

16/ Liệt kê mã những hóa đơn được lập vào tháng 7 năm 2020

**Bài 8:** Sử dụng Aggregation Framework, và cơ sở dữ liệu quản lý sinh viên, viết lệnh thực hiện truy vấn sau:

1/ Hiển thị Mã sinh viên, họ tên và năm sinh của sinh viên.

2/ Hiển thị Mã sinh viên, tên lớp và tuổi của sinh viên. Biết rằng tuổi được tính dựa vào ngày sinh (HD: sử dụng new Date(), để lấy ngày hiện tại, \$year để lấy năm hiện tại, toán tử \$subtract để thực hiện phép trừ năm hiện tại cho năm sinh)

3/ Cho biết Mã môn học, tên môn học và số sinh viên học tương ứng, sắp xếp tăng dần theo mã môn học.

4/ Liệt kê mã lớp, tên lớp và danh sách mã sinh viên đưa vào mảng tương ứng.

5/ Liệt kê mã môn học, tên môn học và danh sách sinh viên đưa vào mảng tương ứng.

6/ Hiển thị mã và tên những môn học có số lượng sinh viên  $\geq 2$

**Bài 9:** Dựa vào cơ sở dữ liệu quản lý sinh viên, viết lệnh thực hiện các thao tác sau:

1/ Hiển thị tất cả các index hiện có trong cơ sở dữ liệu.

2/ Tạo index cho cột Mã sinh viên với hướng chỉ mục tăng.

3/ Xóa index ở câu 2, tạo lại index cho cột Mã sinh viên hướng giảm, thiết lập thuộc tính duy nhất cho chỉ mục.

4/ Tạo index kết hợp cho 2 cột mã sinh viên và họ tên với hướng chỉ mục giảm, thiết lập thuộc tính chỉ mục thưa.

5/ Xóa hết các chỉ mục đã tạo. Tạo lại cùng lúc 2 chỉ mục cho Mã sinh viên và chỉ mục cho Họ tên. có đặt tên cho từng chỉ mục.

6/ Xóa hết các chỉ mục vừa tạo, thực hiện truy vấn bất kỳ có liên quan đến họ tên, địa chỉ và tên lớp, dùng lệnh explain("executionStats") để ghi lại thông tin: thời gian, số lần quét index, số lần quét document. Lần lượt tạo các chỉ mục trên các trường Họ tên, địa chỉ và tên lớp, mỗi khi tạo thêm chỉ mục mới thì ghi lại các thông tin như trên và so sánh các trường hợp với nhau.

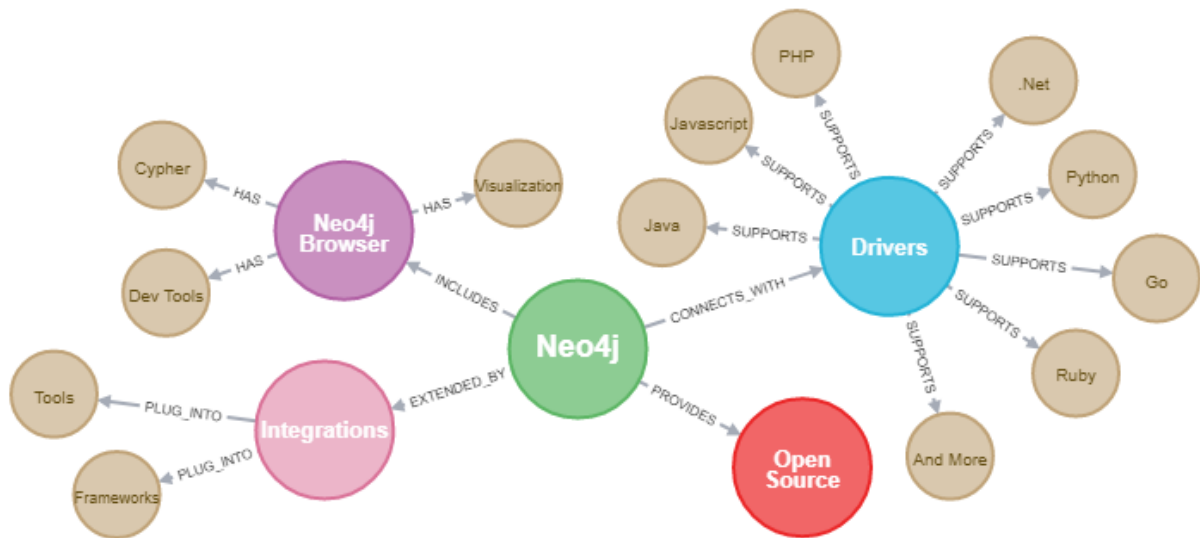


## CHƯƠNG 2: HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU NEO4J

### 2.1. TỔNG QUAN

#### 2.1.1. Giới thiệu

Cơ sở dữ liệu đồ thị là cấu trúc được thiết kế để lưu trữ và xử lý dữ liệu theo mô hình đồ thị. Các hệ quản trị cơ sở dữ liệu đồ thị có cơ chế xử lý dữ liệu hiệu quả với nhiều mối liên kết phức tạp trên đồ thị mà các mô hình dữ liệu khác không thể đáp ứng.



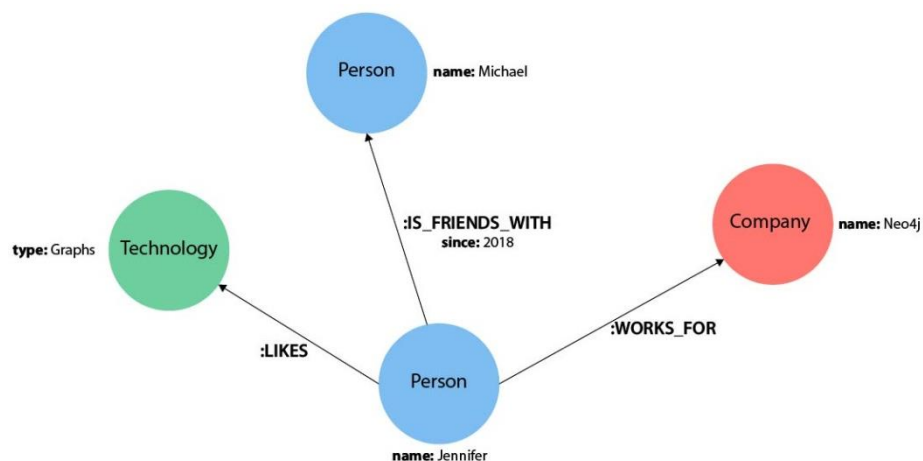
Neo4J là một hệ quản trị CSDL đồ thị được phát triển bởi công ty Neo4J, sử dụng ngôn ngữ Cypher để tạo đồ thị, cập nhật dữ liệu và truy vấn. Cung cấp các driver để kết nối với các ứng dụng lập trình trên nhiều ngôn ngữ khác nhau: Java, .Net, Python,...

Neo4J có 2 phiên bản:

- Neo4J Community (mã nguồn mở)
- Neo4J Enterprise (thương mại)

Ví dụ cơ sở dữ liệu đồ thị được mô tả sau:

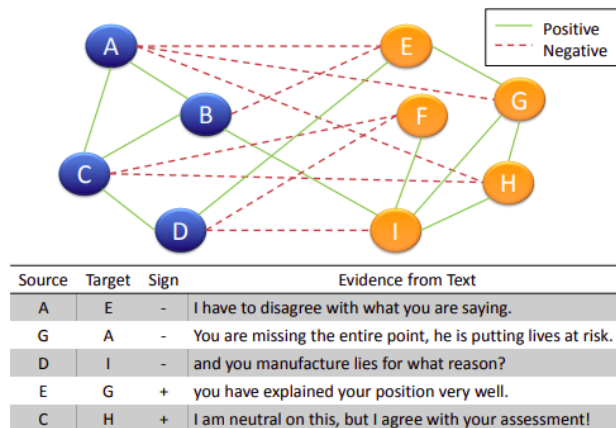
*Jennifer thích Graphs. Jennifer là bạn của Michael. Jennifer làm việc cho Neo4j.*



## 2.1.2. Một số ứng dụng của đồ thị

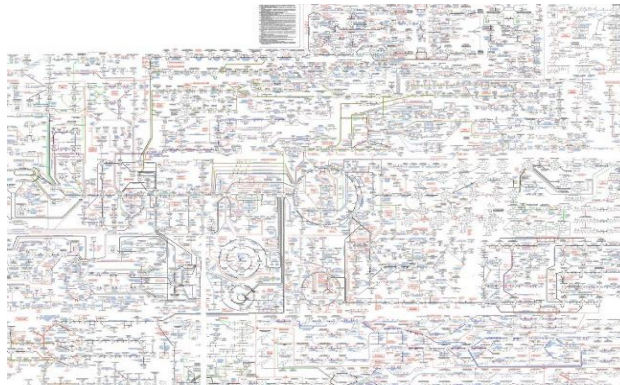
### 2.1.2.1. Khoa học xã hội

Để tìm những người Nổi tiếng X trong mạng xã hội, chúng ta có thể thống kê số lượng Follow của X và đưa ra những người có số lượng Follow cao nhất. Nói theo cách toán học thì ta tìm bậc của đỉnh hay số lượng cạnh nối với đỉnh đó (X) và tìm ra những đỉnh có bậc cao nhất

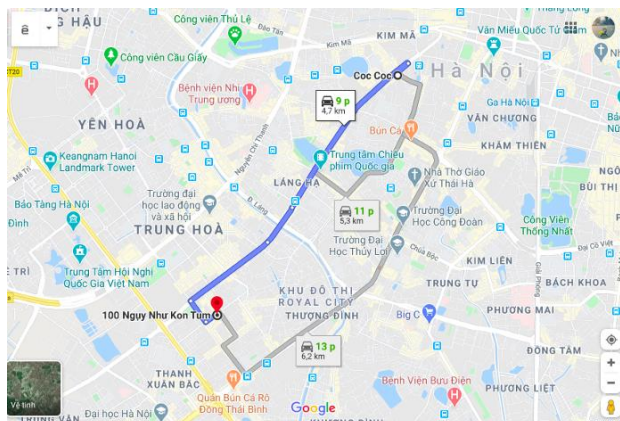


### 2.1.2.2. Nghiên cứu sinh học

Các thành phần sinh học(protein, phân tử, gen) và các tương tác của chúng cũng tạo nên một đồ thị sinh học. Dựa vào đó người ta có thể tìm hiểu được quá trình trao đổi chất trong cơ thể, sự tương tác giữa các bộ phận khác nhau trên cơ thể.

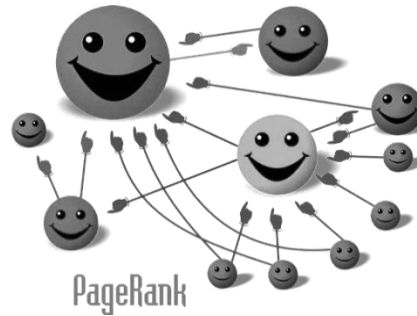


### 2.1.2.3. Bài toán tìm đường đi



#### 2.1.2.4. WebSearch

Pagerank là thuật toán phân tích các liên kết được dùng trong Google Search để xếp hạng các trang web.



## 2.2. NGÔN NGỮ CYPHER

### 2.2.1. Kiểu dữ liệu trong Cypher

Property Type	Structural types	Composite types
<ul style="list-style-type: none"><li>• Number: Integer, Float</li><li>• String</li><li>• Boolean</li></ul>	<b>Node:</b> <ul style="list-style-type: none"><li>• Id</li><li>• Label(s)</li><li>• Map (of properties)</li></ul>	<b>List</b> , a heterogeneous, ordered collection of values, each of which has any property, structural or composite type.
Temporal: <ul style="list-style-type: none"><li>• Date</li><li>• Time</li><li>• LocalTime</li><li>• DateTime</li><li>• LocalDateTime</li><li>• Duration</li></ul>	<b>Relationship:</b> <ul style="list-style-type: none"><li>• Id</li><li>• Type</li><li>• Map (of properties)</li><li>• Id of the start node</li><li>• Id of the end node</li></ul>	<b>Map</b> , a heterogeneous, unordered collection of (Key, Value) pairs. Key is a String Value has any property, structural or composite type
Point	<b>Path</b> , an alternating sequence of nodes and relationships	

### 2.2.2. Quy cách đặt tên

Đặt tên các đối tượng theo quy cách sau:

- Tên đối tượng (nút, quan hệ, biến,...) bắt đầu bằng ký tự alphabet.
- Được chứa dấu gạch dưới “\_”
- Không được bắt đầu bằng số.
- Không được chứa các ký tự đặt biệt như \$, @, ngoại trừ ký tự \$ đứng đầu là tham số
- Có phân biệt chữ hoa, chữ thường

### 2.2.3. Sử dụng ghi chú (comment)

Sử dụng ký hiệu `//` để thiết lập ghi chú:

Ví dụ:

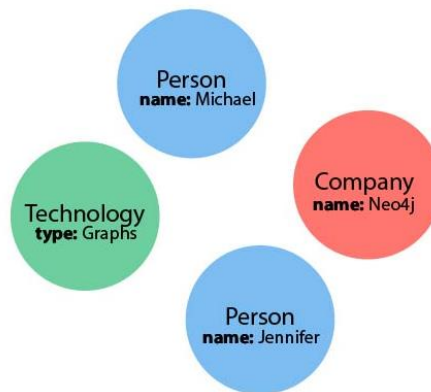
```
//data stored with this direction
CREATE (p:Person)-[:LIKES]->(t:Technology)

//query relationship backwards will not return results
MATCH (p:Person)<-[:LIKES]-(t:Technology)

//better to query with undirected relationship unless sure of direction
MATCH (p:Person)-[:LIKES]-(t:Technology)
```

### 2.2.4. Biểu diễn các nút của đồ thị

Mỗi nút được biểu diễn đồ họa như hình dưới. Ký hiệu (node). Trong hình có 4 nút có tên là: Jennifer, Michael, Graphs và Neo4j



### 2.2.5. Biến và nhãn nút

#### Biến nút (Node Variables)

Sử dụng biến nút để tham chiếu đến một nút có trong đồ thị.

Biểu diễn: (tên\_biến). Nút không có tên biến là nút ẩn danh.

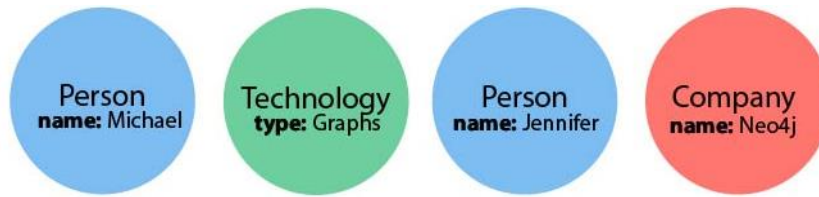
Ví dụ:

- ( ): nút ẩn danh
- (p): biến có tên là p
- (person): biến có tên là person
- (t): biến có tên là t
- (thing): biến có tên là thing

#### Nhãn nút (Node Labels)

Những nút có cùng loại được gán cùng một nhãn. Trong hình dưới, các nhãn nút là: Person, Technology và Company. Tương quan với SQL, mỗi nút tương tự như một

dòng, mỗi nhãn nút tương tự như một bảng. Nếu không chỉ định nhãn thì truy vấn sẽ duyệt qua hết tất cả các nút trong đồ thị.



## 2.2.6. Quan hệ và biến quan hệ

### Mối quan hệ

Mối quan hệ biểu diễn sự quan hệ giữa 2 nút trên đồ thị. Có thể biểu diễn mối quan hệ có hướng hoặc vô hướng. Mối quan hệ có hướng thể hiện quan hệ đơn từ đối tượng thứ nhất đến đối tượng thứ 2 dựa theo chiều mũi tên. Trường hợp mối quan hệ vô hướng thể hiện quan hệ 2 chiều. Tên mối quan hệ thường là động từ.

Có 2 loại quan hệ:

- Có hướng:

`$ -[:RELATIONSHIP]-> hoặc <-[:RELATIONSHIP]-`

- Vô hướng:

`$ -[:RELATIONSHIP]-`

Trong đó: RELATIONSHIP là tên mối quan hệ.

### Ví dụ:

`$ (p:Person)-[:LIKES]->(t:Technology)`

`$ (p:Person)<-[:LIKES]-(t:Technology)`

`$ (p:Person)-[:LIKES]-(t:Technology)`

### Biến mối quan hệ

Biến mối quan hệ được dùng để tham chiếu đến một mối quan hệ có trong đồ thị.

Ký hiệu `-[variable]->`, `<-[variable]-`

hoặc `-[variable]-`

Có thể chỉ định kiểu quan hệ như:

`-[variable:TYPE]->`

Ví dụ:

`$ -[r]-> hoặc -[r:LIKES]->`

## 2.2.7. Thuộc tính đối tượng

Thuộc tính đối tượng trong cơ sở dữ liệu Neo4J mô tả tính chất của đối tượng. Các đối tượng có thể định nghĩa thuộc tính gồm: nút và mối quan hệ.

Biểu diễn thuộc tính: {name:value}

Ví dụ:

– Thuộc tính nút:

```
$ (p:Person {name: 'Jennifer'})
```

– Thuộc tính mối quan hệ:

```
$ -[rel:IS_FRIENDS_WITH {since: 2018}]->
```

### 2.2.8. Mẫu trong Cypher

Một mẫu (pattern) được tạo thành từ các nút và các mối quan hệ. Một mẫu phức tạp có thể được tách thành các mẫu nhỏ đơn giản hơn.

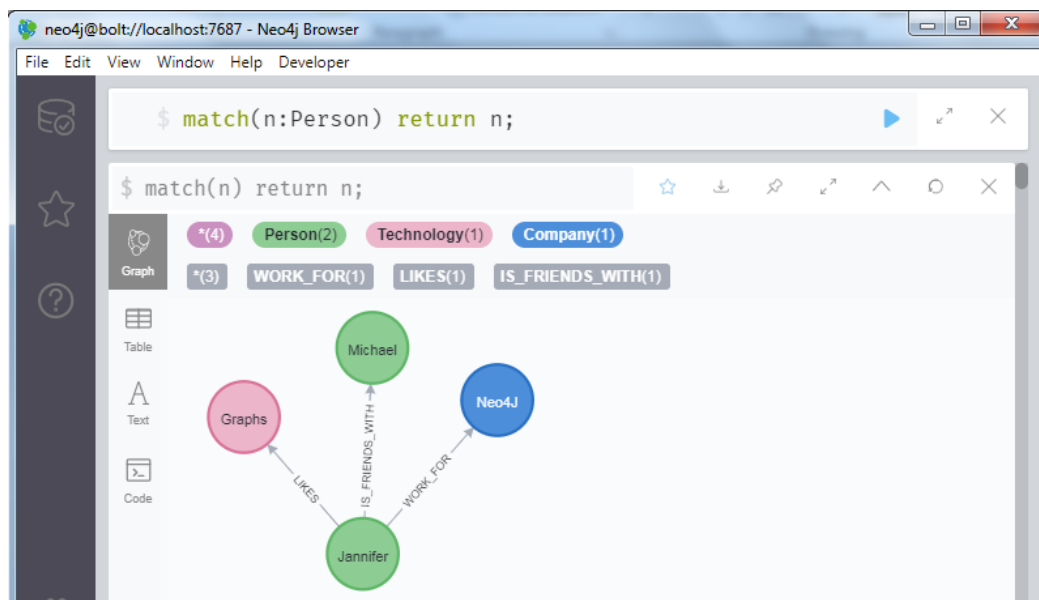
Ví dụ:

```
$ (p:Person {name: "Jennifer"})-[rel:LIKES]->(g:Technology  
{type: "Graphs"})
```

### 2.2.9. Thao tác trên cửa sổ lệnh

Khởi động Neo4J Desktop, thực hiện Start cơ sở dữ liệu cần thao tác, sau đó mở công cụ Neo4J Browser từ Neo4J Desktop.

Gõ lệnh thao tác ở khung nhập phía trên. Xuống dòng lệnh bằng phím Shift+Enter. Thực thi bằng phím Ctrl+Enter hoặc click vào nút thực thi màu xanh bên phải.



## 2.3. CÁC LỆNH CẬP NHẬT DỮ LIỆU

### 2.3.1. Thao tác trên nút

Thêm nút:

– Thêm nút mới không thuộc tính, không nhãn:

\$ Create (a)

– Thêm cùng lúc 2 nút:

\$ Create (b), (c)

– Thêm nút với nhãn Person:

\$ create (p:Person)

– Thêm nút với 2 nhãn là Person và VietNam, hiển thị nút vừa tạo bằng lệnh return

\$ create (p:Person:VietNam) return p;

– Thêm nút với nhãn Person và 2 thuộc tính name và title, hiển thị nút p vừa tạo

\$ create(p:Person {name:'Andy', title:'Developer'}) return p;

### **Xóa nút:**

– Xóa nút thuộc nhãn Person có tên là Andy:

\$ MATCH (p:Person{name:'Andy'}) DELETE p;

– Xóa nút thuộc nhãn Person có tên là Andy và bất kỳ liên kết nối với nút.

\$ MATCH (p:Person {name: 'Andy'}) DETACH DELETE n;

– Xóa tất cả các nút và mối liên kết:

\$ MATCH (n) DETACH DELETE n;

### **Thêm/sửa thuộc tính nút:**

– Thêm/sửa thuộc tính surname = 'Taylor' cho nút có name là Andy:

```
$ MATCH (n {name: 'Andy'})  
  SET n.surname = 'Taylor'  
  RETURN n.name, n.surname;
```

– Thêm/sửa thuộc tính birthdate cho nút có name là Jennifer

```
$ MATCH (p:Person {name: 'Jennifer'})  
  SET p.birthdate = date('1980-01-01')  
  RETURN p
```

### **Xóa thuộc tính nút:**

– Xóa thuộc tính surname

```
$ MATCH (n {name: 'Andy'})  
  REMOVE n.surname  
  RETURN n
```

### **2.3.2. Thao tác trên mối liên kết**

– Thêm liên kết:

```
$ MATCH (a:Person), (b:Person)
  WHERE a.name = 'Andy' AND b.name = 'Michael'
  CREATE (a)-[r:KNOWS{since:2018}]->(b)
  RETURN a,b
```

– Xóa liên kết:

```
$ MATCH (n {name: 'Andy'})-[r:KNOWS]->()
  DELETE r
```

– Thêm/sửa thuộc tính của liên kết:

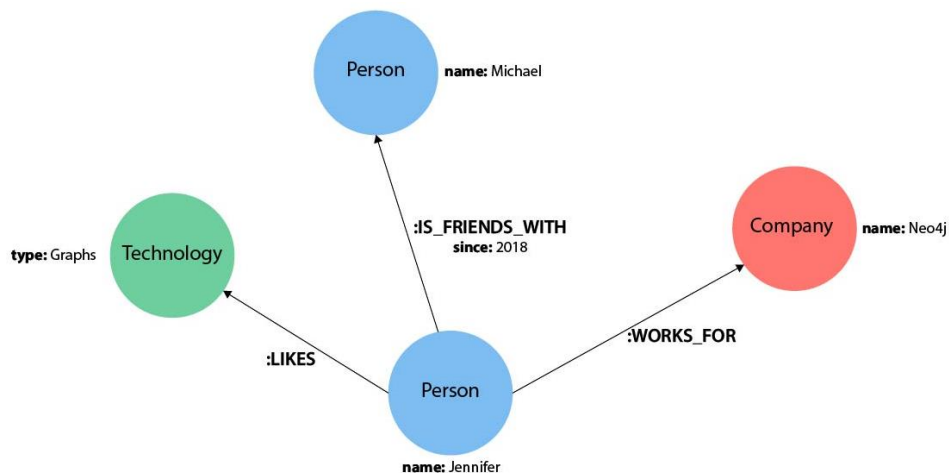
```
$ MATCH (n{name: 'Andy'})-[r:KNOWS]->(m{name: 'Michael'})
  set r.since=2018
  return n,m
```

– Xóa thuộc tính của liên kết:

```
$ MATCH (n{name: 'Andy'})-[r:KNOWS]->(m{name: 'Michael'})
  REMOVE r.since
  return n,m
```

### 2.3.3. Ví dụ tạo cơ sở dữ liệu đồ thị

Tạo cơ sở dữ liệu đồ thị như sau:



**Tạo các nút:**

– Tạo node tên là Jennifer có nhãn Person

```
$ Create (p:Person{id:1,name: 'Jennifer'})
  return p;
```

– Thiết lập id là khóa:

```
$ create constraint on (p:Person) assert p.id is unique
```

– Tạo node tên là Michael có nhãn Person

```
$ Create (p:Person{id:2,name: 'Michael'})
  return p;
```



- Tạo node type là Graphs, có nhãn Technology

```
$ Create (p:Technology{id:1,type: 'Graphs'}) return p;
```

- Thiết lập id là khóa cho nhãn Technology:

```
$ create constraint on (p:Technology) assert p.id is unique
```

- Tạo node tên là Neo4J có nhãn Company

```
$ Create (p:Company{id:1,name: 'Neo4J'}) return p;
```

- Thiết lập id là khóa cho nhãn Company

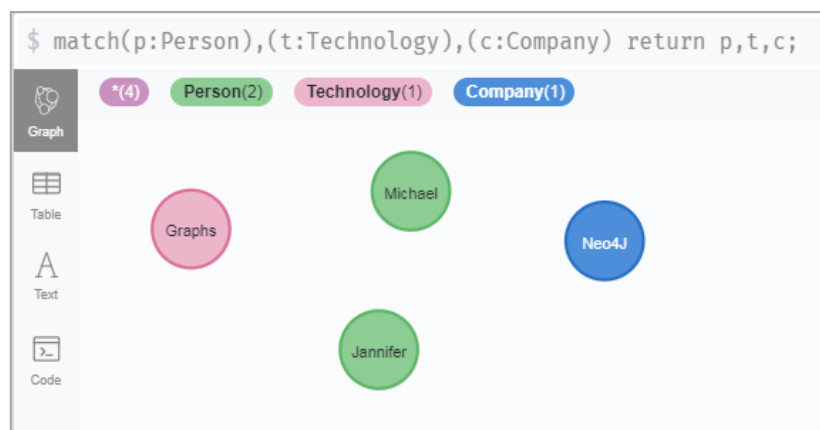
```
$ create constraint on (p:Company) assert p.id is unique
```

- Hiển thị các nút đã tạo

```
$ match (p:Person), (t:Technology), (c:Company) return p,t,c;
```

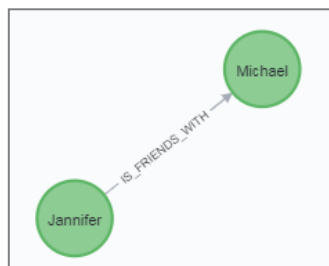
- Hoặc

```
$ match (n) return n;
```



- Tạo liên kết giữa 2 nút Jennifer và Michael có thuộc tính *since:2018*

```
$ MATCH (a:Person), (b:Person)
WHERE a.name='Jennifer' and b.name='Michael'
CREATE (a)-[r:IS_FRIENDS_WITH{since:2018}]->(b)
RETURN a,b;
```



- Tạo liên kết giữa 2 nút Jennifer và Graphs không có thuộc tính.

```
$ match (a:Person), (b:Technology)
where a.name='Jennifer' and b.type='Graphs'
```

```
create (a)-[r:LIKES]->(b)
return a,b;
```

- Tạo liên kết giữa 2 nút Jennifer và Neo4J không thuộc tính

```
$ match(a:Person), (b:Company)
  where a.name='Jennifer' and b.name='Neo4J'
  create (a)-[r:WORK_FOR]->(b)
  return a,b;
```

- Xem các nút và liên kết

```
$ match(a:Person), (b:Company), (c:Technology) return a,b,c;
```

## 2.4. TRUY VẤN DỮ LIỆU

### 2.4.1. Các lệnh truy vấn cơ bản

- Hiển thị tất cả các nút

```
$ MATCH(p) RETURN p;
```

- Hiển thị các nút thuộc nhãn Person

```
$ MATCH(p:Person) RETURN p;
```

- Hiển thị nút thuộc nhãn Person có tên là Jennifer

```
$ MATCH(j:Person{name:'Jennifer'}) RETURN j
```

- Hiển thị tên công ty mà người có tên Jennifer làm việc.

```
$ MATCH(:Person{name:'Jennifer'})-[:WORK_FOR]-> (c:Company)
  RETURN c;
```

✎ Có thể dùng RETURN c.name để chỉ trả về tên công ty

- Hiển thị label của nút

```
$ MATCH (p{name:"Jennifer"}) return labels(p);
```

- Hiển thị tất cả các label có trong CSDL

```
$ CALL db.labels();
```

- Hiển thị type của mỗi liên kết

```
$ MATCH (p)-[r]->(q) return type(r);
```

### 2.4.2. Sử dụng mệnh đề Where

- Hiển thị nút thuộc nhãn Person có tên là Jennifer

```
$ MATCH(j:Person{name:'Jennifer'}) RETURN j
```

Trương đương khi sử dụng WHERE

```
$ MATCH(p:Person)
```

```
WHERE p.name='Jennifer'
RETURN p;
```

- **Hiển thị tên công ty mà người có tên Jennifer làm việc.**

```
$ MATCH (p:Person) , (c:Company)
WHERE (p)-[:WORK_FOR]->(c)
RETURN c;
```

- **Hiển thị những người không có tên là Jennifer**

```
$ MATCH (j:Person)
WHERE NOT j.name = 'Jennifer'
RETURN j;
```

- **Hiển thị những người có tuổi từ 20 đến 30**

```
$ MATCH (p:Person)
WHERE 20<=p.age<=30
RETURN p;
```

- **Hiển thị tên những người có thuộc tính birthdate**

```
$ MATCH (p:Person)
WHERE exists(p.birthdate)
RETURN p.name
```

### 2.4.3. Xử lý chuỗi

- **Điều kiện thuộc tính bắt đầu bằng ký tự 'M'**

```
$ MATCH (p:Person)
WHERE p.name STARTS WITH 'M'
RETURN p.name;
```

- **Thuộc tính có chứa ký tự 'a'**

```
$ MATCH (p:Person)
WHERE p.name CONTAINS 'a'
RETURN p.name;
```

- **Thuộc tính kết thúc với ký tự 'n'**

```
$ MATCH (p:Person)
WHERE p.name ENDS WITH 'n'
RETURN p.name;
```

- **Thuộc tính có 2 ký tự đầu là Je**

```
$ MATCH (p:Person)
WHERE p.name =~ 'Je.*'
RETURN p.name;
```

- **Sử dụng từ khóa IN**

```
$ MATCH (p:Person)
  WHERE p.yearsExp IN [1, 5, 6]
  RETURN p.name, p.yearsExp;
```

#### 2.4.4. Exists và Not Exists với pattern

- Tìm những người là bạn của những người làm việc cho công ty có tên là Neo4J

```
$ MATCH (p)-[:IS_FRIENDS_WITH]-(q)
  WHERE EXISTS (
    (p)-[:WORK_FOR]->(:Company{name:'Neo4J'}))
  return q;
```

- Tìm những người bạn của Jennifer mà không làm việc cho công ty nào.

```
$ MATCH (p:Person)-[r:IS_FRIENDS_WITH]->(friend:Person)
  WHERE p.name = 'Jennifer'
  AND NOT exists((friend)-[:WORKS_FOR]->(:Company))
  RETURN friend.name
```

#### 2.4.5. Một số truy vấn phức tạp

- Tìm những người thích Graphs ngoài Jennifer

```
$ MATCH (j:Person {name: 'Jennifer'})-[:LIKES]-
  (graph:Technology {type: 'Graphs'})-[:LIKES]-(p:Person)
  RETURN p.name;
```

- Tìm những người thích Graphs ngoài Jennifer và Jennifer cũng là bạn với những người đó.

```
$ MATCH (j:Person {name: 'Jennifer'})-[:LIKES]->(:Technology
  {type: 'Graphs'})<-[:LIKES]-(p:Person),
  (j)-[:IS_FRIENDS_WITH]-(p)
  RETURN p.name;
```

#### 2.4.6. Aggregation trong Cypher

##### Các hàm thống kê:

- Đếm số lượng người có email

```
$ MATCH (p:Person)
  RETURN count(p.email);
```

Nếu dùng count(\*) sẽ đếm hết các đối tượng kể cả có thuộc tính email bằng null

- Cho biết độ tuổi trung bình

```
$ MATCH (p:Person)
  RETURN avg(p.age);
```

Sử dụng tương tự cho các hàm sum, min, max

## 2.4.7. Kết hợp giá trị với collect

Hàm collect sẽ đưa các kết quả vào danh sách

- Hiện thị tên của mỗi người và danh sách bạn bè của người đó

```
$ MATCH (p:Person)-[:IS_FRIENDS_WITH]->(friend:Person)
RETURN p.name, collect(friend.name) AS friend
```

p.name	friend
"Sally"	["John", "Jennifer"]
"Dan"	["Ann"]
"John"	["Sally", "Jennifer"]
"Diana"	["Joe"]
"Jennifer"	["Sally", "John", "Ann", "Mark", "Melissa"]
"Ann"	["Dan", "Jennifer"]
"Mark"	["Joe", "Jennifer"]
"Joe"	["Diana", "Mark"]

- Hiện thị tên của mỗi người và số lượng bạn bè của người đó.

```
$ MATCH (p:Person)-[:IS_FRIENDS_WITH]->(friend:Person)
RETURN p.name, size(collect(friend.name)) AS
numberOfFriends
```

p.name	numberOfFriends
"Jennifer"	2

- Hiện thị tên của mỗi người, danh sách bạn bè tương ứng sao cho mỗi người trong danh sách này có số lượng bạn bè >1

```
$ MATCH (p:Person)-[:IS_FRIENDS_WITH]->(friend:Person)
WHERE size((friend)-[:IS_FRIENDS_WITH]-(:Person)) > 1
RETURN p.name, collect(friend.name) AS friends,
size((friend)-[:IS_FRIENDS_WITH]-(:Person)) AS numberOfFoFs
```

p.name	friends	numberOfFoFs
"Joe"	["Mark"]	2
"Jennifer"	["Mark", "John", "Sally", "Ann"]	2
"John"	["Sally"]	2

## 2.4.8. Sử dụng WITH

WITH được dùng để chuyển giá trị từ phần này đến phần khác trong câu truy vấn.

```
$ MATCH (a:Person)-[r:LIKES]-(t:Technology)
WITH a.name AS name, collect(t.type) AS technologies
```

```
RETURN name, technologies
```

- Sử dụng WITH viết lại truy vấn: Hiển thị tên của mỗi người, danh sách bạn bè tương ứng sao cho mỗi người trong danh sách này có số lượng bạn bè >1

```
$ MATCH (p:Person)-[:IS_FRIENDS_WITH]->(friend:Person)
  WITH p, collect(friend.name) AS friendsList, size((friend)-
    [:IS_FRIENDS_WITH]-(:Person)) AS numberOfFoFs
  WHERE numberOfFoFs > 1 RETURN p.name, friendsList,
    numberOfFoFs
```

- WITH còn được sử dụng để thiết lập giá trị tham số trước câu truy vấn.

```
$ WITH 2 AS experienceMin, 6 AS experienceMax
  MATCH (p:Person)
  WHERE experienceMin <=p.yrsExperience <= experienceMax
  RETURN p
```

#### 2.4.9. Sử dụng UNWIND

UNWIND được dùng để tách một mảng thành các giá trị riêng lẻ để xử lý.

**Ví dụ:**

- Với danh sách kỹ năng cho trước, cần tìm những người có kỹ năng tương ứng trong danh sách.

```
$ WITH ['Graphs','Query Languages'] AS techRequirements
  UNWIND techRequirements AS technology
  MATCH (p:Person)-[r:LIKES]-(t:Technology {type:
    technology})
  RETURN t.type, collect(p.name) AS potentialCandidates
```

\$ WITH ["Graphs","Query Languages"] as techRequirements U...				
			t.type	potentialCandidates
			"Graphs"	["Diana", "Mark", "Melissa", "Jennifer"]
			"Query Languages"	["Diana", "Melissa", "Joe"]

- Với danh sách số năm kinh nghiệm, tìm các ứng viên có số năm kinh nghiệm thuộc danh sách.

```
$ WITH [4, 5, 6, 7] AS experienceRange
  UNWIND experienceRange AS number
  MATCH (p:Person) WHERE p.yearsExp = number
  RETURN p.name, p.yearsExp
```

\$ WITH [4, 5, 6, 7] as experienceRange UNWIND experienceR...			📄
<div>Table</div> <div>A</div> <div>Text</div> <div>&lt;/&gt;</div>	p.name	p.yearsExp	
	"Sally"	4	
	"Jennifer"	5	
	"John"	5	
	"Dan"	6	

#### 2.4.10. Sắp xếp kết quả

ORDER BY được dùng để sắp xếp kết quả truy vấn. Mặc định sắp tăng dần, Sử dụng DESC để sắp giảm dần.

**Ví dụ:**

- Sử dụng truy vấn trước: Với danh sách số năm kinh nghiệm, tìm các ứng viên có số năm kinh nghiệm thuộc danh sách.

```
$ WITH [4, 5, 6, 7] AS experienceRange
  UNWIND experienceRange AS number
  MATCH (p:Person)
  WHERE p.yearsExp = number
  RETURN p.name, p.yearsExp
  ORDER BY p.yearsExp DESC
```

#### 2.4.11. Loại bỏ dòng trùng

Distinct được dùng để loại bỏ dòng kết quả trùng trong truy vấn

Ví dụ: Tìm những người có tài khoản email hoặc thích graphs hoặc thích query languages

```
$ MATCH (user:Person)
  WHERE user.email IS NOT null
  WITH user
  MATCH (user)-[:LIKES]-(t:Technology)
  WHERE t.type IN ['Graphs','Query Languages']
  RETURN DISTINCT user.name
```

#### 2.4.12. Giới hạn kết quả trả về

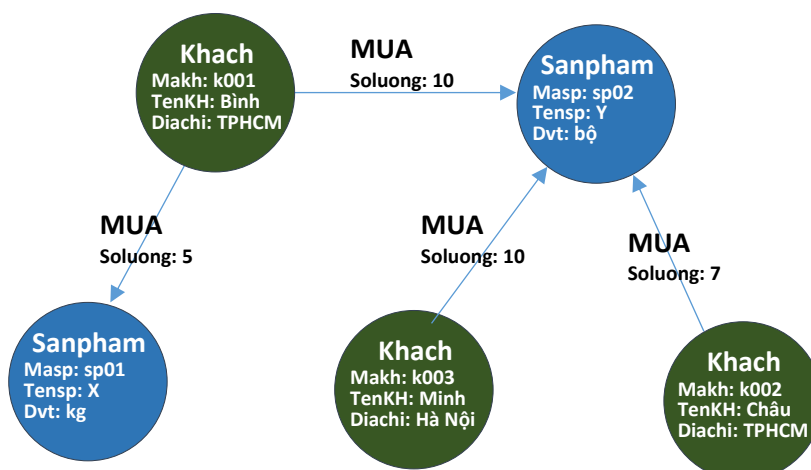
Limit được sử dụng để giới hạn kết quả trả về của câu truy vấn

Ví dụ: Tìm 3 người đầu tiên có nhiều bạn nhất

```
$ MATCH (p:Person)-[r:IS_FRIENDS_WITH]-(other:Person)
  RETURN p.name, count(other.name) AS numberOfFriends
  ORDER BY numberOfFriends DESC
  LIMIT 3
```

## 2.5. BÀI TẬP CHƯƠNG 2

**Bài 1: Cho cơ sở dữ liệu đồ thị như sau:**

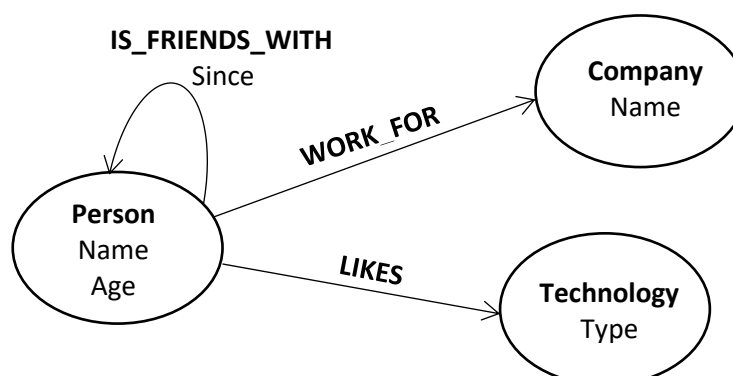


**1/ Viết lệnh tạo các nút và mối liên kết cho cơ sở dữ liệu trên.**

**2/ Viết các câu truy vấn sau:**

- Liệt mã và tên những khách hàng có địa chỉ ở TPHCM.
- Liệt kê những sản phẩm có đơn vị tính là kg.
- Hiển thị những khách hàng có mua sản phẩm Y.
- Cho biết mã và tên những khách hàng mua sản phẩm X với số lượng  $\geq 10$
- Mã và tên những khách hàng nào mua sản phẩm X và cũng mua sản phẩm Y.
- Những khách hàng nào mua sản phẩm Y mà không mua sản phẩm X.
- Liệt kê mã khách hàng, tên khách hàng và số các mặt hàng mà khách hàng đã mua.
- Liệt kê mã khách hàng, tên khách hàng và danh sách các mặt hàng mà khách hàng đã mua.
- Cho biết tên sản phẩm và tổng số lượng bán ra cho những khách hàng ở TPHCM.
- Liệt kê mã sản phẩm, tên sản phẩm và danh sách các khách hàng mua tương ứng.

**Bài 2: Cho mô hình dữ liệu đồ thị như sau:**



**1/ Hãy tạo cơ sở dữ liệu theo mô hình trên dựa vào mô tả như sau:**

**Tạo các nút:**

- Các nút thuộc Person: Lan: 26 tuổi, Tuấn: 32 tuổi, Châu: 24, Minh: 21, Nam: 45 tuổi



- Các nút thuộc nhãn Company: C2Net, Tech2A, MPA
- Các nút thuộc nhãn Technology: Java, C#, Web, NoSQL

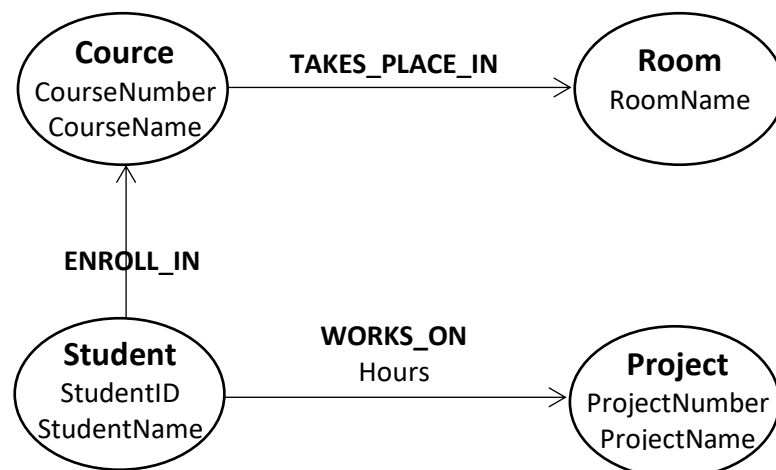
### Tạo các mối liên kết:

- Lan là bạn của Tuấn từ năm 2015, thích các công nghệ như C#, Java và làm việc tại công ty Tech2A
- Tuấn là bạn của Minh từ năm 2012, bạn của Châu từ năm 2017, thích Web và NoSQL.
- Châu là bạn của Lan từ năm 2016, bạn của Nam và Minh năm 2018, thích Web và NoSQL, làm việc tại công ty Tech2A.
- Nam làm việc tại công ty MPA, thích Java.
- Minh là bạn của Nam, làm việc tại công ty C2Net

### 2/ Viết các câu truy vấn sau:

- Liệt kê những công nghệ là người tên Lan thích.
- Ai thích công nghệ NoSQL, liệt kê tên và tuổi của những người đó.
- Cho biết tên những người làm việc tại công ty Tech2A
- Liệt kê những người bạn của Nam và những người bạn đó làm tại công ty C2Net
- Cho biết số người bạn của Minh.
- Cho biết số người bạn của Minh và những người bạn đó cũng là bạn của Lan
- Hiển thị tên và số người bạn của từng người.
- Hiển thị tên và số người bạn của từng người với số người bạn từ 2 trở lên.
- Công nghệ nào có ít nhất 2 người thích.
- Hiển thị tên công nghệ và danh sách những người thích tương ứng.
- Hiển thị tên, tuổi từng người và danh sách các công nghệ mà người đó thích, sắp xếp tăng dần theo tuổi.
- Cho biết tên những công ty mà những người bạn của Tuấn làm ở đó.
- Với danh sách các công nghệ [Web, Java], hãy cho biết tên những người thích các công nghệ này.

### Bài 3: Cho mô hình dữ liệu đồ thị như sau:



1/ Sao chép và thực thi tập lệnh tạo các nút có dữ liệu như sau:

```
CREATE (s1: Student {StudentID: "1", StudentName: "Ana"})
CREATE (s2: Student {StudentID: "2", StudentName: "Peter"})
CREATE (s3: Student {StudentID: "3", StudentName: "John"})
CREATE (s4: Student {StudentID: "4", StudentName: "Stine"})
CREATE (s5: Student {StudentID: "5", StudentName: "Michael"})
CREATE (c1: Course {CourseNumber: "1", CourseName: "Databases"})
CREATE (c2: Course {CourseNumber: "2", CourseName: "Programming"})
CREATE (c3: Course {CourseNumber: "3", CourseName: "Graphics"})
CREATE (p1: Project {ProjectNumber: "34", ProjectName: "eCommerce database"})
CREATE (p2: Project {ProjectNumber: "24", ProjectName: "eCommerce website"})
CREATE (p3: Project {ProjectNumber: "13", ProjectName: "User interface"})
CREATE (p4: Project {ProjectNumber: "26", ProjectName: "Reporting"})
CREATE (r1: Room {RoomName: "Pascal"})
CREATE (r2: Room {RoomName: "Seminar C"})
CREATE (r3: Room {RoomName: "Alpha"})
CREATE (r4: Room {RoomName: "Beta"})
```

2/ Sao chép và thực thi lệnh tạo các liên kết theo mô tả sau:

```
CREATE (c1) -[:TAKES_PLACE_IN] -> (r1)
CREATE (c1) -[:TAKES_PLACE_IN] -> (r3)
CREATE (c1) -[:TAKES_PLACE_IN] -> (r4)
CREATE (c2) -[:TAKES_PLACE_IN] -> (r2)
CREATE (s1) -[:ENROLLED_IN] -> (c1)
CREATE (s2) -[:ENROLLED_IN] -> (c1)
CREATE (s3) -[:ENROLLED_IN] -> (c2)
CREATE (s4) -[:ENROLLED_IN] -> (c1)
CREATE (s1) -[:WORKS_ON {hours: 1}] -> (p1)
CREATE (s1) -[:WORKS_ON {hours: 2}] -> (p2)
CREATE (s2) -[:WORKS_ON {hours: 3}] -> (p1)
CREATE (s2) -[:WORKS_ON {hours: 4}] -> (p2)
CREATE (s2) -[:WORKS_ON {hours: 1}] -> (p3)
CREATE (s2) -[:WORKS_ON {hours: 1}] -> (p4)
CREATE (s3) -[:WORKS_ON {hours: 1}] -> (p1)
CREATE (s3) -[:WORKS_ON {hours: 2}] -> (p2)
CREATE (s3) -[:WORKS_ON {hours: 3}] -> (p4)
```

3/ Viết lệnh truy vấn cho các yêu cầu sau:

- a/ Hiển thị danh sách gồm mã khóa học, tên khóa học.
- b/ Hiển thị số giờ mà sinh viên Ana làm đề án có tên *eCommerce database*
- c/ Cho biết mã và tên những sinh viên đăng ký khóa học tên là Databases
- d/ Cho biết mã và tên những khóa học có từ 2 sinh viên
- e/ Liệt kê những khóa học được đăng ký bởi sinh viên có tên là John
- f/ Mã và tên những khóa học nào diễn ra tại phòng có tên là Seminar C
- g/ Những khóa học nào diễn ra tại ít nhất 2 phòng. Thông tin liệt kê gồm: mã khóa học, tên khóa học.
- h/ Liệt kê mã, tên những sinh viên đăng ký khóa học diễn ra tại phòng Seminar C
- i/ Với những phòng mà khóa học (CourseNumber=1) diễn ra. Cho biết tên khóa học và tên phòng tương ứng.
- j/ Với sinh viên có StudentID là 1, cho biết tên sinh viên, tên dự án và số giờ mà sinh viên làm việc.

k/ Với dự án có ProjectID là 24, cho biết tên dự án, tên sinh viên tham gia và số giờ mà sinh viên làm việc trên dự án đó.

l/ Liệt kê mã đề án, tên đề án và danh sách những nhân viên tham gia đề án đó.

m/ Với những sinh viên làm việc trên 2 đề án, cho biết tên sinh viên và số lượng đề án mà sinh viên đó tham gia, sắp xếp giảm dần theo số lượng đề án.

n/ Với danh sách dự án cho trước [User interface, Reporting], liệt kê những sinh viên tham gia tương ứng.

#### **Bài 4: Mô hình chăm sóc khách hàng.**

Mô tả: Một công ty cần xây dựng hệ thống chăm sóc khách hàng theo mô hình đồ thị với các thông tin: Hệ thống có một người là tổng quản lý sẽ phụ trách chung cho toàn hệ thống. Mỗi khu vực (Tỉnh/Thành) có một quản lý khu vực và nhiều nhân viên chăm sóc khách hàng trong khu vực đó. Mỗi nhân viên được phân công chăm sóc một số khách hàng, nhân viên cần nắm rõ thông tin về khách hàng như những sản phẩm mà khách hàng đã mua tại công ty cũng như sở thích và nhu cầu của khách hàng để có những chiến lược tư vấn phù hợp.

1/ Hãy thiết kế mô hình đồ thị cho hệ thống được mô tả trên.

2/ Tạo cơ sở dữ liệu dựa vào mô hình trên.

## CHƯƠNG 3: HỆ QUẢN TRỊ CASSANDRA

### 3.1. TỔNG QUAN

#### 3.1.1. Giới thiệu

Cassandra là một hệ quản trị cơ sở dữ liệu phân tán, không quan hệ (NoSQL), được thiết kế để xử lý khối lượng dữ liệu lớn với khả năng mở rộng linh hoạt và tính sẵn sàng cao. Cassandra hỗ trợ xử lý trên mô hình Wide Column, được phát triển bởi Facebook và hiện đang được duy trì bởi Apache Software Foundation.

#### 3.1.2. Đặc điểm chính

**Khả năng mở rộng theo chiều ngang (Horizontal Scalability):** Cassandra cho phép mở rộng hệ thống bằng cách thêm nhiều nút vào cluster mà không cần phải thay đổi cấu trúc dữ liệu hay dừng hoạt động của hệ thống. Điều này làm cho Cassandra trở thành lựa chọn lý tưởng cho các ứng dụng cần xử lý lượng dữ liệu rất lớn và yêu cầu hiệu suất cao.

**Tính sẵn sàng cao (High Availability):** Với kiến trúc phân tán và khả năng sao chép dữ liệu giữa các nút, Cassandra đảm bảo tính sẵn sàng và độ tin cậy ngay cả khi một hoặc nhiều nút gặp sự cố. Mỗi phần dữ liệu được sao chép nhiều lần (theo replication factor), giúp hệ thống tiếp tục hoạt động mà không bị gián đoạn.

**Đọc và ghi nhanh (Fast Reads and Writes):** Cassandra hỗ trợ việc ghi dữ liệu nhanh chóng bằng cách sử dụng cấu trúc dữ liệu dựa trên bảng phân phối (Partitioned tables) và lưu trữ dữ liệu theo định dạng "commit log". Điều này giúp giảm thiểu độ trễ và tăng hiệu suất xử lý các thao tác ghi và đọc.

**Khả năng tùy chỉnh chính sách phân phối dữ liệu (Flexible Data Distribution):** Cassandra cung cấp các chiến lược phân phối dữ liệu linh hoạt như SimpleStrategy và NetworkTopologyStrategy, cho phép người dùng tùy chỉnh cách dữ liệu được phân phối và sao chép giữa các datacenter.

**Hỗ trợ truy vấn mạnh mẽ (Powerful Query Capabilities):** Cassandra sử dụng Cassandra Query Language (CQL), một ngôn ngữ truy vấn tương tự như SQL, giúp người dùng dễ dàng thao tác với dữ liệu. CQL hỗ trợ các truy vấn phức tạp và khả năng tìm kiếm hiệu quả trong hệ thống phân tán.

**Tính bền vững cao (High Durability):** Dữ liệu trong Cassandra được ghi vào commit log và sau đó được lưu trữ trong bộ nhớ (memtable) và đĩa. Điều này đảm bảo rằng dữ liệu không bị mất ngay cả khi có sự cố hệ thống hoặc lỗi phần cứng.

Với những ưu điểm trên, Cassandra là lựa chọn hàng đầu cho các ứng dụng cần khả năng mở rộng cao, tính sẵn sàng liên tục và khả năng xử lý dữ liệu lớn, chẳng hạn như các nền tảng mạng xã hội, các dịch vụ streaming dữ liệu và các ứng dụng phân tích dữ liệu thời gian thực.

## 3.2. CƠ SỞ DỮ LIỆU CASSANDRA

Cơ sở dữ liệu trong Cassandra được tổ chức theo một cấu trúc phân tán và linh hoạt, dựa trên mô hình dữ liệu Wide column. Các thành phần chính trong cơ sở dữ liệu bao gồm Keyspace, Table, Row và các dạng khóa. Dưới đây là các thành phần trong cơ sở dữ liệu Cassandra:

### 3.2.1. Không gian khóa

Không gian khóa (Keyspace): là đơn vị chính để tổ chức dữ liệu trong Cassandra, tương đương với cơ sở dữ liệu trong các hệ quản trị cơ sở dữ liệu quan hệ.

#### Tạo Keyspace:

Cấu trúc:

```
CREATE KEYSPACE keyspace_name
WITH replication = {
    'class': 'SimpleStrategy',
    'replication_factor': replication_factor
};
```

Trong đó:

- *Keyspace\_name*: Tên của keyspace bạn muốn tạo.
- *SimpleStrategy* hoặc *NetworkTopologyStrategy*: Với giá trị class SimpleStrategy thường được sử dụng cho môi trường phát triển hoặc môi trường đơn giản, trong khi NetworkTopologyStrategy thích hợp hơn cho môi trường dữ liệu lớn với cấu hình nhiều nút.
- *Replication\_factor*: Số lượng bản sao của dữ liệu muốn lưu trữ. Ví dụ, nếu replication\_factor là 3, dữ liệu sẽ được sao chép đến 3 nút khác nhau.

Ví dụ: Tạo một keyspace có tên qlsv:

```
CREATE KEYSPACE qlsv
WITH replication = {
    'class': 'SimpleStrategy',
    'replication_factor': 3
};
```

#### Xóa Keyspace:

DROP KEYSPACE keyspace\_name;

Ví dụ: DROP KEYSPACE banhang

### 3.2.2. Kiểu dữ liệu trong Cassandra

Kiểu dữ liệu	Mô tả
ascii	Biểu diễn cho một chuỗi ký tự ASCII
bigint	Đại diện cho số nguyên có dấu dài 64-bit
blob	Lưu trữ các byte tùy ý
boolean	Giá trị true hoặc false
counter	Đại diện một số nguyên dài 64-bit, giá trị của cột này chỉ có hai hoạt động trên cột này, tăng và giảm.
date	Lưu trữ giá trị ngày, không có giờ
decimal	Giá trị thập phân
double	Lưu trữ một giá trị dấu chấm động dài 64-bit.
float	Lưu trữ một giá trị dấu chấm động dài 32-bit.
inet	Biểu diễn cho một chuỗi địa chỉ IP trong định dạng của IPv4 hoặc IPv6.
int	Lưu trữ số nguyên có dấu dài 32-bit
smallint	Biểu diễn số nguyên 16-bit
text	Biểu diễn chuỗi UTF8
time	Chuỗi thời gian có dạng 01:02:03.123
timestamp	Lưu trữ dữ liệu ngày và giờ với độ chính xác mili giây
timeuuid	Lưu trữ chuỗi UUID phiên bản 1
tinyint	Số nguyên 1 byte 8 bit
uuid	Lưu trữ chuỗi UUID chuẩn
varchar	Lưu trữ chuỗi UTF8 tương tự kiểu Text
varint	Số nguyên với độ chính xác tùy ý

### 3.2.3. Bảng

Mỗi keyspace chứa một hoặc nhiều bảng (Table). Bảng trong Cassandra lưu trữ dữ liệu dưới dạng hàng và cột. Trên bảng có các thành phần như khóa chính, khóa phân vùng, khóa thứ cấp, chỉ mục giúp tối ưu hóa việc phân tán dữ liệu và hiệu suất thực thi trên cơ sở dữ liệu lớn.

**Khóa chính:** Khóa chính là một tập hợp các cột được sử dụng để xác định duy nhất một dòng trong bảng. Khóa chính bao gồm hai phần:

- **Khóa phân vùng (Partition Key):** Xác định cách dữ liệu được phân phối trên các nút trong cluster.
- **Khóa thứ cấp (Clustering Columns):** Xác định cách dữ liệu trong cùng một phân vùng được sắp xếp.

**Lệnh tạo bảng:**

```
CREATE TABLE table_name
(
    column1 datatype1,
    column2 datatype2,
    ...
    PRIMARY KEY(partition_key, clustering_column1,
                clustering_column2,...)
);
```

Ví dụ:

```
CREATE TABLE ketqua
(
    masv text,
    mamh text,
    lanthi int,
    diemthi float,
    primary key(masv,mamh,lanthi)
);
```

**Lệnh xóa bảng:**

```
Drop table <tên bảng>
```

Ví dụ: Drop table ketqua

**Thêm cột vào bảng:**

```
ALTER TABLE <table_name>
ADD <column_name> <data_type>;
```

Ví dụ: ALTER TABLE lop

```
ADD siso int
```

**Thêm dữ liệu vào bảng:**

```
INSERT INTO <table_name> (<column1>, <column2>, <column3>, ...)
VALUES (<value1>, <value2>, <value3>, ...);
```

Ví dụ:

```
INSERT INTO ketqua (masv, mamh, lanthi, diemthi)
VALUES ('sv001', 'mh001', 1, 8)
```

Để thêm nhiều dòng dữ liệu cùng lúc, sử dụng BEGIN BATCH....APPLY BATCH

Ví dụ:

```
BEGIN BATCH
    INSERT INTO ketqua (masv, mamh, lanthi, diemthi)
    VALUES ('sv001', 'mh002', 1, 9);
    INSERT INTO ketqua (masv, mamh, lanthi, diemthi)
    VALUES ('sv002', 'mh001', 1, 8.5);
APPLY BATCH
```

Ví dụ: Tạo bảng user sử dụng kiểu dữ liệu UUID cho cột userid và kiểu dữ liệu TIMESTAMP cho cột datecreate. Thực hiện chèn dữ liệu vào bảng.

```
CREATE TABLE user
(
    userid UUID PRIMARY KEY,
    username varchar,
    fullname varchar,
    datecreate timestamp
)
```

Thêm dữ liệu vào bảng user:

```
BEGIN BATCH
INSERT INTO user(userid,username,fullname,datecreate)
VALUES(uuid(),'vanminh','Trần Văn Minh',toTimestamp(now()));
INSERT INTO user(userid,username,fullname,datecreate)
VALUES(uuid(),'ngoc','Đỗ Thị Ngọc',toTimestamp('2024-05-24'));
APPLY BATCH
```

**Xóa dữ liệu trong bảng:**

```
DELETE FROM <table_name>
WHERE <primary_key_column> = <primary_key_value>;
```

Ví dụ:

```
DELETE FROM monhoc WHERE mamh='mh001'
```

**Sửa dữ liệu trong bảng:**

```
UPDATE <table_name>
SET <column1> = <value1>, <column2> = <value2>, ...
```



WHERE <primary\_key\_column> = <primary\_key\_value>;

Ví dụ:

```
UPDATE monhoc  
SET ngaytao=toTimestamp(now()), sotc=2  
WHERE mamh='mh001'
```

**Tạo chỉ mục trên bảng:**

CREATE INDEX <tên index> ON <table\_name> (<column\_name>);

Ví dụ: CREATE INDEX idx\_tenmh ON monhoc(tenmh)

**Xóa chỉ mục:**

DROP INDEX <tên index>

Ví dụ: DROP INDEX idx\_tenmh

### 3.3. TRUY VẤN DỮ LIỆU

Trong Cassandra, việc truy vấn dữ liệu thường được thực hiện dựa trên các khóa phân vùng và khóa thứ cấp để tối ưu hóa hiệu suất truy vấn. Cách dữ liệu được sắp xếp trong từng phân vùng giúp các truy vấn theo cột cụ thể trở nên nhanh chóng và hiệu quả, nhờ vào việc Cassandra sử dụng cấu trúc dữ liệu như SSTables và Bloom Filters để tối ưu hóa việc truy xuất.

Cú pháp:

```
SELECT *|<column1>, <column2>, ...  
FROM <table_name>  
WHERE <primary_key_column> = <primary_key_value>;
```

Ví dụ 1: Truy vấn trên bảng ketqua với điều kiện từ khóa phân vùng là masv và khóa thứ cấp mamh:

```
SELECT * FROM ketqua  
WHERE masv='sv001' AND mamh='mh001';
```

Ví dụ 2: Truy vấn trên bảng monhoc với điều kiện trên thuộc tính sotc không phải là khóa phân vùng thì phải sử dụng ALLOW FILTERING.

```
SELECT * FROM monhoc WHERE mamh='mh001'  
SELECT * from monhoc WHERE sotc=3 ALLOW FILTERING;
```

**Các hàm thống kê:**

COUNT: Đếm số lượng hàng trong tập hợp dữ liệu.

SUM: Tính tổng của các giá trị số trong một cột.

AVG: Tính giá trị trung bình của các giá trị số trong một cột.

MIN: Tìm giá trị nhỏ nhất trong một cột.

MAX: Tìm giá trị lớn nhất trong một cột.

Ví dụ:

```
SELECT count(*) as somh
FROM ketqua
WHERE masv='sv001'
```

### 3.4. BÀI TẬP THỰC HÀNH

**Bài 1:** Thao tác trên cửa sổ lệnh Cassandra Shell:

a/ Khởi động Cassandra, sử dụng Cassandra Shell thực hiện các lệnh như sau:

```
cqlsh> help
cqlsh> describe keyspaces;
cqlsh> describe tables;
cqlsh> describe <keyspaces name>.<table name>;
cqlsh> CREATE KEYSPACE qlnhanvien WITH REPLICATION =
{
    'class': 'SimpleStrategy',
    'replication_factor': '3'
};
```

Nếu thấy xuất hiện cảnh báo: “*Warnings : Your replication factor 3 for keyspace banhang is higher than the number of nodes 1*”. Thực hiện xóa keyspace và tạo lại cho phù hợp.

b/ Trong keyspace qlnhanvien, tạo một bảng nhanvien có cấu trúc như sau:

Tên cột	Kiểu dữ liệu	Ghi chú
manv	text	Khóa chính
hoten	text	
ngaysinh	date	
phai	text	
luong	int	

HD: cqlsh> create table qlnhanvien.nhanvien (  
...)

c/ Nhập dữ liệu vào bảng nhanvien 3 dòng phù hợp.

HD: `cqlsh>insert into qlnhanvien.nhanvien(manv,hoten,...) values(...)`

## Bài 2: Thao tác trên công cụ hỗ trợ giao diện xử lý cơ sở dữ liệu Cassandra

a/ Tạo keyspace có tên qlsinhvien với các tham số:

```
'class': 'SimpleStrategy',  
'replication_factor': '3'
```

b/ Trên keyspace qlsinhvien, tạo các bảng dữ liệu theo cấu trúc như sau:

sinhvien

Tên cột	Mô tả	Kiểu dữ liệu	Thuộc tính khóa
masv	Mã sinh viên	text	Khóa chính
hoten	Họ tên	text	
ngaysinh	Ngày sinh	date	
phai	Phái	text	
dienthoai	Điện thoại	text	
lop	Lớp học	text	

dangky

Tên cột	Mô tả	Kiểu dữ liệu	Thuộc tính khóa
masv	Mã sinh viên	text	x
mamh	Mã môn học	text	x
tenmh	Tên môn học	date	
hocky	Học kỳ	text	x
ngaydangky	Timestamp	text	

c/ Sử dụng lệnh Insert into để nhập dữ liệu vào mỗi bảng ít nhất 5 dòng dữ liệu. Có thể dùng khối BEGIN BATCH...APPLY BATCH để nhập nhiều dòng cùng lúc.

d/ Cập nhật dữ liệu với yêu cầu như sau:

- Sửa họ tên và điện thoại của sinh viên có mã SV002 thành Đỗ Văn Thanh và 0984453226
- Xóa sinh viên có mã SV005
- Sửa ngày đăng ký của sinh viên SV001 đăng ký môn MH003 thành 2024-08-15

e/ Viết lệnh truy vấn sau:

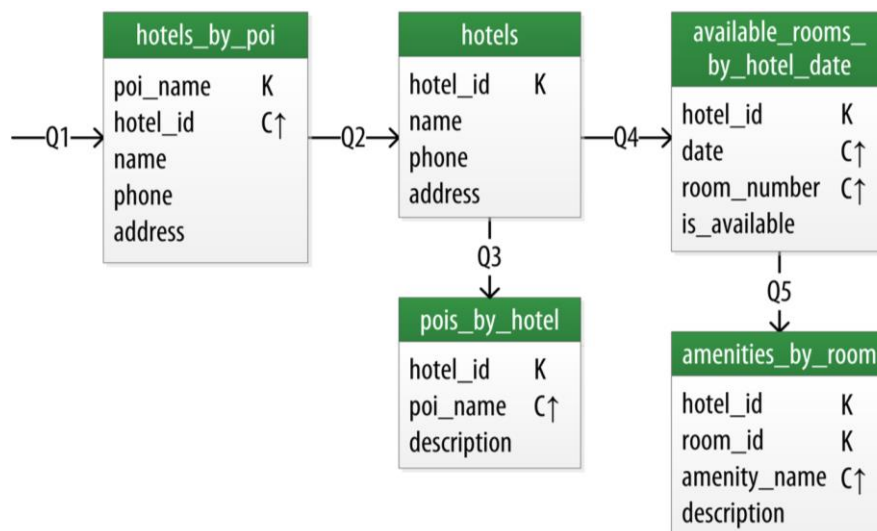
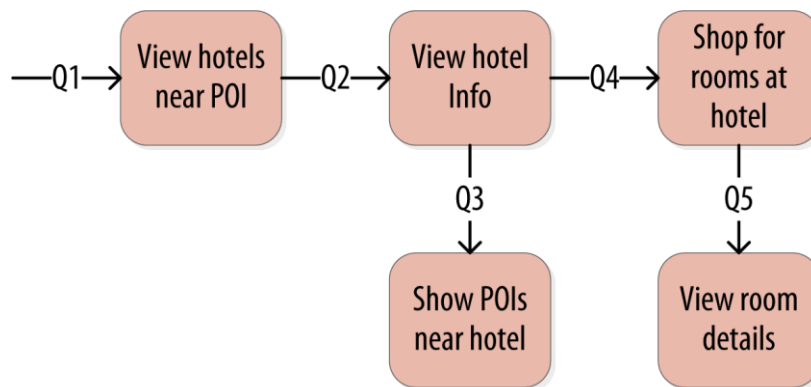
- Liệt kê các thông tin của sinh viên có mã là SV001
- Liệt kê mã sv, họ tên, ngày sinh, phái của những sinh viên thuộc lớp 15DHTH
- Hiện thị danh sách môn học: mã môn, tên môn mà sinh viên SV001 đã đăng ký.

### Bài 3: Xây dựng cơ sở dữ liệu Cassandra theo mô tả như sau:

Cần xây dựng cơ sở dữ liệu Cassandra lưu trữ thông tin về các khách sạn, khách lưu trú tại khách sạn, danh sách các phòng trong mỗi khách sạn, giá cả và tình trạng phòng trống, danh sách các địa điểm quan tâm gần khách sạn để khách có thể ghé thăm trong thời gian lưu trú.

Các truy vấn của ứng dụng được xác định như sau:

- Q1: Tìm khách sạn gần một điểm tham quan nhất định.
- Q2: Tìm thông tin về một khách sạn nhất định, tên và vị trí của khách sạn đó.
- Q3: Tìm điểm tham quan gần một khách sạn nhất định.
- Q4: Tìm phòng trống trong một khoảng thời gian nhất định.
- Q5: Tìm giá và tiện nghi cho một phòng.



- Hãy viết lệnh tạo các bảng với cấu trúc được xác định trên.
- Nhập dữ liệu vào mỗi bảng ít nhất 5 dòng dữ liệu.
- Viết các lệnh truy vấn đáp ứng yêu cầu thiết kế trên.

**Bài 4:** Hãy thiết kế mô hình dữ liệu cho việc đăng ký khám chữa bệnh tại bệnh viện với các yêu cầu như sau:

a/ Thiết kế các truy vấn (4-5 câu)

b/ Thiết kế luồng dữ liệu cho các truy vấn

c/ Thiết kế cấu trúc bảng đáp ứng yêu cầu truy vấn

d/ Viết lệnh tạo bảng và tạo các truy vấn đáp ứng yêu cầu của ứng dụng.

## CHƯƠNG 4: ỨNG DỤNG KẾT NỐI CSDL NoSQL

Việc ứng dụng cơ sở dữ liệu MongoDB, Neo4J và Cassandra cũng tương tự như các hệ quản trị khác, cần xây dựng các giao diện giao tiếp người dùng bằng một ngôn ngữ lập trình cụ thể và kết nối đến MongoDB, Neo4J, Cassandra. Các hệ quản trị này hỗ trợ rất nhiều ngôn ngữ lập trình ứng dụng như: C#, Java, Python, PHP,... Mỗi ngôn ngữ được xây dựng gói chương trình gọi là driver tương ứng để kết nối thao tác trên cơ sở dữ liệu.

### 4.1. ỨNG DỤNG KẾT NỐI MONGODB

#### 4.1.1. Ứng dụng trên C# kết nối MongoDB

##### Cài đặt thư viện

Để kết nối ứng dụng viết bằng C# với MongoDB, trước tiên phải cài đặt gói chương trình: *mongo-csharp-driver*.

Có 2 cách cài đặt:

- Cài đặt online bằng *Nuget Package Manager*:

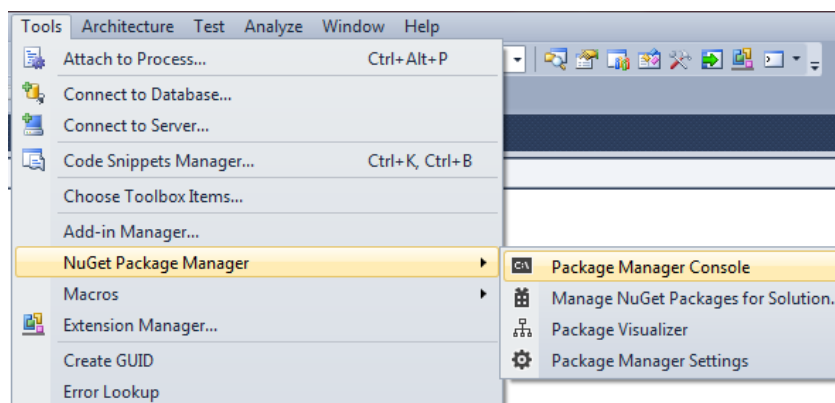
Link cài đặt: <https://www.nuget.org/packages/mongocsharpdriver>

- Hoặc cài đặt bằng cách add References 2 tập tin .dll (nếu đã có file .DLL):

1. MongoDB.Bson.dll
2. MongoDB.Driver.dll

##### Cài đặt bằng Nuget:

**Bước 1:** Trên project hiện hành, vào menu Tool > NuGet Package Manager > Package Manager Console



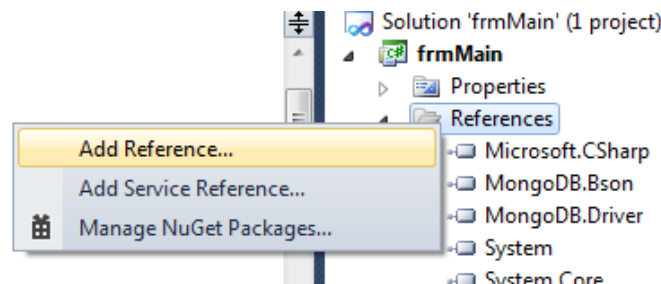
**Bước 2:** Trên cửa sổ Package Manager Console, gõ vào lệnh cài đặt, nhấn Enter

```
Package Manager Console
Package source: All [v] [g] Default project: frmMain

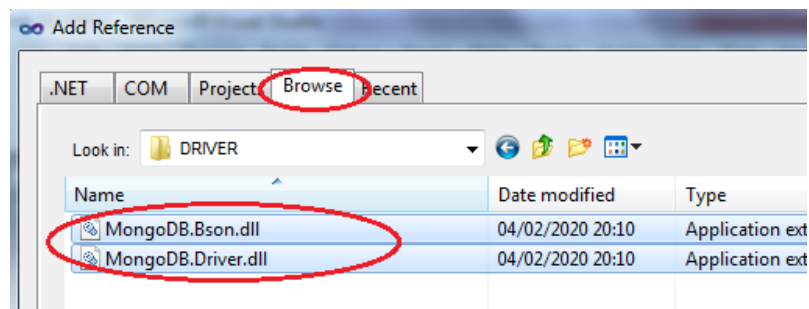
PM> NuGet\Install-Package mongocsharpdriver -Version 2.28.0
```

## Cài đặt bằng file .DLL:

**Bước 1:** Mở rộng danh mục project > Click phải vào References > chọn Add References



**Bước 2:** Chọn Tab Browse > chọn 2 tập tin .dll > OK



Sau khi cài đặt *mongo-csharp-driver*, ngoài các namespace mặc định, sử dụng thêm các namespace sau để kết nối mongodb:

```
6 using System.Linq;
7 using System.Text;
8 using System.Windows.Forms;
9 using MongoDB.Driver;
10 using MongoDB.Bson;
11
```

## Kết nối database

– Sử dụng chuỗi `connectionString` kết nối cơ sở dữ liệu MongoDB:

```
var connectionString="mongodb://localhost:27017";
var client = new MongoClient(connectionString);
```

– Lấy đối tượng database:

```
var db = client.GetDatabase("mydb");
```

– Lấy đối tượng collection:

```
var collection=db.GetCollection<BsonDocument>(collectionName);
```

collectionName: Tên của collection trong database. Ví dụ: NHANVIEN

## Thao tác trên collection

### – Duyệt các Document:

```
var list = collection.Find(new BsonDocument()).ToList();
foreach (var document in list)
{
    string value = document.GetElement(fieldname).Value.ToString();
}
```

**Trong đó:** fieldname: là tên của thuộc tính cần lấy giá trị, kết quả lưu vào biến value.

### – Thêm document vào collection:

#### ○ Thêm với các trường đơn:

```
var document = new BsonDocument()
    .Add(field1, valueOfField1)
    .Add(field2, valueOfField2)
    ...
    .Add(fieldN, valueOfFieldN);
collection.InsertOne(document);
```

**Trong đó:**

- Field1, field2,...là tên của các thuộc tính.
- valueOfField1, valueOfField2,...là các giá trị thuộc tính tương ứng.

#### ○ Thêm document con (nhúng):

### – Khai báo parentDocument và childDocument:

```
var parentDocument = new BsonDocument();
var childDocument = new BsonDocument();
//Thêm các element vào childDocument
childDocument.AddRange(new BsonElement[] {
    new BsonElement("childfield1", "value1"),
    new BsonElement("childfield2", "value2"),
    ...
});
//Thêm childDocument vào parentDocument
parentDocument.Add("fieldname", childDocument)
```

#### ○ Thêm mảng các giá trị:

### – Khai báo 2 đối tượng kiểu BsonDocument và BsonArray

```
var parentDocument = new BsonDocument();
var bsonArray=new BsonArray(arrayValue);
//Với arrayValue là mảng các giá trị cần truyền vào
//Thêm bsonArray vào parentDocument
parentDocument.Add("fieldname", bsonArray)
```

#### ○ Thêm mảng các document con(nhúng)

### – Khai báo một đối tượng kiểu BsonDocument và một mảng BsonDocument

```
var parentDocument = new BsonDocument();
BsonDocument[] arrayChildDocument = new BsonDocument[size];
```



```

//Với size là kích thước mảng cho trước
//Gán giá trị cho từng childDocument trong mảng
for(int i=0;i<size;i++)
{
    arrayChildDocument[i]=new BsonDocument();
    arrayChildDocument[i].AddRange(
        new BsonElement[]{new BsonElement("childfield1", "value1"),
            new BsonElement("childfield2", "value2"),
            ...
        });
}
//Khai báo đối tượng bsonArray để chứa mảng các document con
var bsonArray= new BsonArray(arrayChildDocument);
//Thêm bsonArray vào parentDocument
parentDocument.Add("fieldname",bsonArray)

```

– **Xóa document trong collection:**

```

var filter = Builders<BsonDocument>.Filter.Eq(fieldName, valueOfField);
collection.DeleteOne(filter);

```

– **Sửa document trong collection:**

```

var filter = Builders<BsonDocument>.Filter.Eq(fieldName, FieldValue);
var update = Builders<BsonDocument>.Update.Set(fieldName1, FieldValue1)
    .Set(fieldName2, FieldValue2)
    ...
collection.UpdateOne(filter, update);

```

– **Sửa giá trị của document nhúng**

```

var update = Builders<BsonDocument>.Update.Set(fieldName.SubField,
SubFieldValue)
    //Các lệnh khác tương tự như sửa một giá trị cụ thể

```

– **Sửa bằng cách Thêm/bớt giá trị mảng**

```

var update = Builders<BsonDocument>.Update.AddToSet("fieldArrayName",
valueOfArray);
//Trường hợp loại bỏ giá trị ra khỏi mảng thì dùng Update.Pull(...)

```

– **Truy vấn và lấy dữ liệu:**

```

var filter = Builders<BsonDocument>.Filter.<Eq/Lt/...>(fieldname, value);
var list = collection.Find(filter).ToList();
foreach (var document in list)
{
    string value = document.GetElement(fieldname).Value.ToString();
}

```

– **Lấy dữ liệu từ field con:**

```

var collection = db.GetCollection<BsonDocument>("collectionName");
var filter = Builders<BsonDocument>.Filter.Eq(fieldName, value);
foreach (var item in collection.Find(filter).ToList())
{
    //show embedded document
}

```

```

        var bson=(BsonDocument)item.GetElement("fieldName").Value;
        //fieldName là tên field của document nhúng.
        //Ví dụ: Phongban:{Maph:"p1",Tenph:"Kinh doanh"}
        //thì fieldName là Phongban.
        //Lấy giá trị của subfield để hiển thị lên form
        <control1>.Text=bson.GetElement("subfieldName1").Value.ToString();
        <control2>.Text=bson.GetElement("subfieldName2").Value.ToString();
        ...

        //show array
var bsonArray = (BsonArray)item.GetElement("fieldName").Value;
    foreach (var b in bsonArray)
    {
        //Lấy từng giá trị trong mảng gán cho đối tượng trên form
        <control>.Text=b.ToString();
    }

    //show array document
var bsonArray = (BsonArray)item.GetElement("fieldName").Value;
//fieldName là tên field của mảng document nhúng.
    foreach (var document in bsonArray)
    {
        var doc = (BsonDocument)document;
        //Lấy từng giá trị của các subfield trong từng document con
        string value1 = doc.GetElement("subfieldName1").Value.ToString(),
        string value2 = doc.GetElement("subfieldName2").Value.ToString();
    }
}

```

#### 4.1.2. Ứng dụng trên Java kết nối MongoDB

##### Cài đặt các gói thư viện

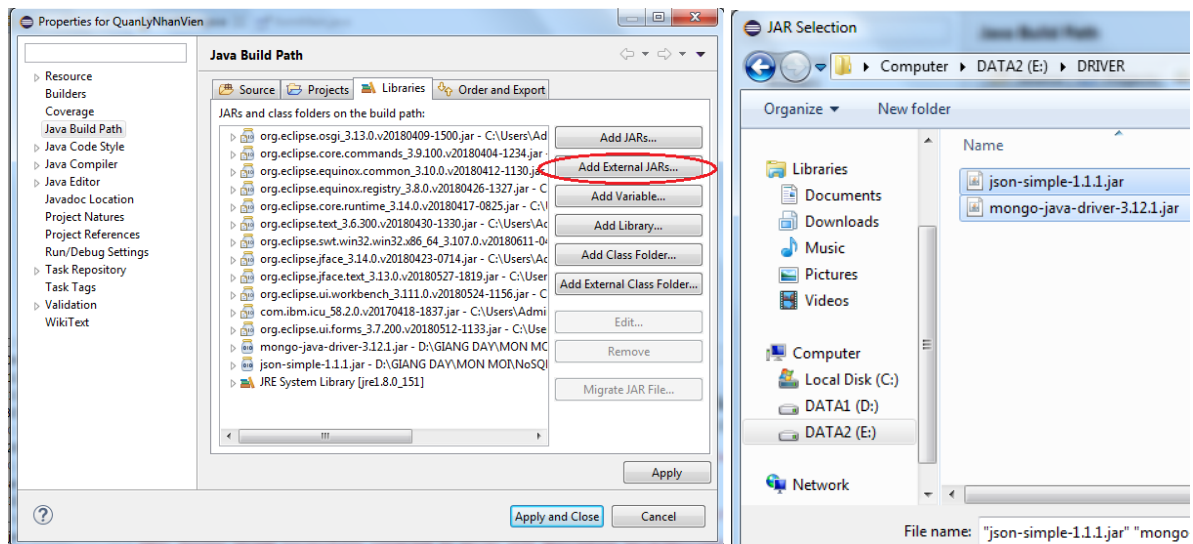
Có nhiều thư viện hỗ trợ ứng dụng java kết nối cơ sở dữ liệu MongoDB. Trong tài liệu này trình bày sử dụng bộ thư viện *Mongodb Java Driver* kết hợp *Json-simple*

Link download:

1. Mongodb Java Driver:  
[https://jar-download.com/?search\\_box=mongo-java-driver](https://jar-download.com/?search_box=mongo-java-driver)
2. Json-simple:  
<https://jar-download.com/artifacts/com.googlecode.json-simple/json-simple/1.1.1/source-code>

##### Import MongoDB Java Driver:

Click phải vào project > chọn Build Path > chọn Configure Build Path > click nút lệnh Add External Jars > chọn các file cần import.



Sau khi cài đặt *mongoDB Java driver*, ngoài các package mặc định, import thêm các package sau để kết nối mongodb:

```

17 import org.eclipse.swt.widgets.Item;
18
19 import org.json.simple.*;
20 import org.json.simple.parser.JSONParser;
21 import org.json.simple.parser.ParseException;
22
23 import com.mongodb.MongoClient;
24 import com.mongodb.client.MongoCollection;
25 import com.mongodb.client.MongoDatabase;
26 import static com.mongodb.client.model.Filters.*;
27

```

#### – Kết nối MongoDB:

```
MongoClient mongoClient = new MongoClient("localhost", 27017);
```

#### – Lấy đối tượng database:

```
MongoDatabase database = mongoClient.getDatabase(dbName);
```

#### – Lấy đối tượng collection:

```
MongoCollection<Document> collection = database.getCollection(collectName);
```

#### – Duyệt các Document:

```

for (Document doc : collection.find()){
    JSONParser parser = new JSONParser();
    JSONObject json = (JSONObject) parser.parse(doc.toJson());
    String value = json.get(fieldName).toString();
}

```

**Trong đó:** *fieldName* là tên của thuộc tính cần lấy giá trị, kết quả lưu vào biến *value*.

#### – Thêm document vào collection:

```
Document doc=new Document(field1, valueOfField1)
    .append(field2, valueOfField2)
    .append(field3, valueOfField3)
    ...
    .append(fieldN, valueOfFieldN)
collection.insertOne(doc);
```

**Trong đó:**

- Field1, field2,...là tên của các thuộc tính.
- valueOfField1, valueOfField2,...là các giá trị thuộc tính tương ứng.

– **Xóa document trong collection:**

```
collection.deleteOne(eq(FieldName,Value));
```

**Trong đó:**

- FieldName: là tên của các thuộc tính cần thiết lập điều kiện xóa.
- Value: là giá trị của thuộc tính làm điều kiện xóa.
- eq: (equal) so sánh bằng. Ngoài ra còn có các ký hiệu khác như: ne (not equal), gt (greater than), lt (less than), gte (greater than or equal), lte (less than or equal)

– **Sửa document trong collection:**

```
collection.updateOne(eq(FieldName,oldValue), new Document("$set", new
Document(FieldName,newValue)));
```

**Trong đó:** FieldName là tên thuộc tính của document cần sửa thông tin. oldValue là giá trị cũ của thuộc tính. newValue là giá trị mới cần sửa.

## 4.2. ỨNG DỤNG KẾT NỐI NEO4J

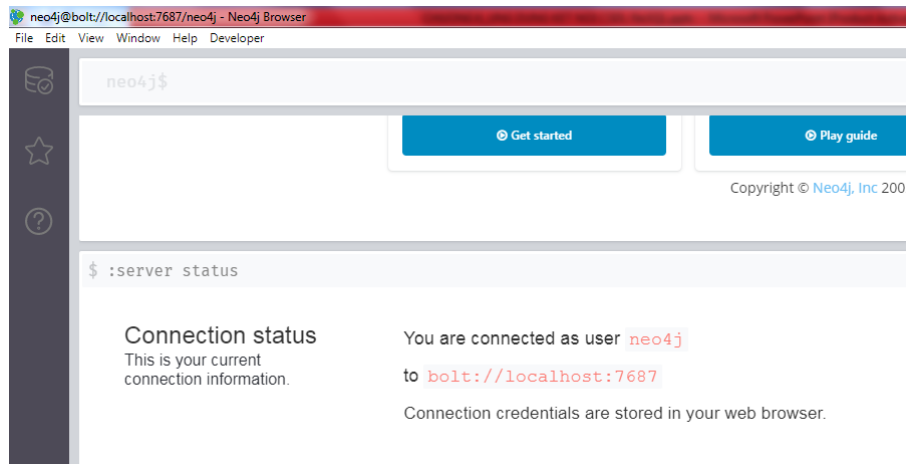
Neo4J hỗ trợ nhiều ngôn ngữ lập trình ứng dụng kết nối cơ sở dữ liệu đồ thị. Tài liệu này chỉ trình bày ứng dụng Java kết nối Neo4J. Các ngôn ngữ khác thực hiện tương tự.

### 4.2.1. Cài đặt gói chương trình driver

- Download gói chương trình: *neo4j-java-driver*
- Link download: <https://jar-download.com/artifact-search/neo4j-java-driver>
- Giải nén và import vào project

### 4.2.2. Kết nối Neo4J

- Khởi động Neo4J Desktop > start database
- Khởi động Neo4J Browser để lấy thông tin user và uri hoặc gõ lệnh :server status;



– Import driver

```
import org.neo4j.driver.*;
```

– Kết nối Neo4J:

```
driver=GraphDatabase.driver(uri,AuthTokens.basic(user,password))
```

Ví dụ:

```
driver = GraphDatabase.driver("bolt://localhost:7687", AuthTokens.basic("neo4j", "123"));
```

### 4.2.3. Lệnh thao tác

– Cấu trúc lệnh thao tác:

```
try (Session session = driver.session())
{
    //lệnh xử lý
}
```

– Xử lý kết quả truy vấn:

```
try (Session session = driver.session())
{
    Result result=session.run("Lệnh truy vấn...As name");
    while(result.hasNext())
    {
        Record record=result.next();
        System.out.Print(record.get("name").asString());
    }
}
```

Ví dụ: Truyền vào tên một người, trả về tên của những người bạn tương ứng.

```
private void printPeople(String nodeName)
{
    try (Session session = driver.session())
    {
        Result result=session.run("MATCH (p:Person {name:\""+
            nodeName +"\"})-[:IS_FRIENDS_WITH]->(q) return q.name
            AS name");
        while (result.hasNext())
        {
```

```

        Record record = result.next();
        System.out.println(record.get("name").asString());
    }
}

```

#### 4.2.4. Sử dụng tham số

- Khai báo cấu trúc HashMap để lưu các tham số và giá trị tương ứng

```

Map<String, Object> params = new HashMap<>();
params.put(attribute, value);

```

- Ví dụ: truyền vào tham số tên một người, trả về tên của những người bạn tương ứng.

```

private void printPeople(String name_value)
{
    try (Session session = driver.session())
    {
        Map<String, Object> params = new HashMap<>();
        params.put("name_parameter", name_value);
        Result result = session.run("MATCH (p:Person{name:$name_parameter}), (q:Person) WHERE (p)-[:IS_FRIENDS_WITH]->(q) return q.name AS name", params);
        while (result.hasNext())
        {
            Record record = result.next();
            System.out.println(record.get("name").asString());
        }
    }
}

```

- Ví dụ 2: Tạo nút có 2 tham số truyền vào là name và age:

```

private void addPerson(String name, int age)
{
    try (Session session = driver.session())
    {
        Map<String, Object> params = new HashMap<>();
        params.put("name", name);
        params.put("age", age);
        session.writeTransaction(tx -> tx.run("CREATE(a:Person {name: $name, age: $age})", params));
    }
}

```

### 4.3. ỨNG DỤNG KẾT NỐI CASSANDRA

#### Cài đặt thư viện:

Để kết nối ứng dụng viết bằng C# với Cassandra, trước tiên phải cài đặt gói chương trình: *CassandraCSharpDriver*. Có thể cài đặt online bằng *Nuget Package Manager* theo link: <https://www.nuget.org/packages/CassandraCSharpDriver/>

#### Kết nối database:

```
using Cassandra;

var cluster = Cluster.Builder()
    .AddContactPoint("Địa chỉ ip của cassandra")
    .Build();

var session = cluster.Connect("tên database");
```

Ví dụ: Địa chỉ IP của Cassandra là "127.0.0.1", tên database là "qlnhanvien"

### Thêm dữ liệu vào database:

```
var cql = session.Prepare("insert into tên_bảng(danh sách thuộc tính)
values(?,?,?,...)");
var bindvalue = cql.Bind("danh sách giá trị");
session.Execute(bindvalue);
```

### Sửa dữ liệu:

```
var cql = session.Prepare("update tên_bảng set tên_cột1=?,tên_cột2=?,...
where tên_cột3=?");
var bindvalue = cql.Bind("danh sách giá trị");
session.Execute(bindvalue);
```

### Xóa dữ liệu:

```
var cql = session.Prepare("delete from tên_bảng where tên_cột=?");
var bindvalue = cql.Bind("giá trị");
session.Execute(bindvalue);
```

### Truy vấn:

- Truy vấn không điều kiện:

```
rowSet = session.Execute("select * from tên_bảng");
```

- Truy vấn có điều kiện:

```
var cql = session.Prepare("select * from tên_bảng where tên_cột =?");
var bindvalue = cql.Bind("giá trị");
rowSet = session.Execute(bindvalue);
```

### Lấy giá trị từ truy vấn:

```
foreach (var row in rowSet)
{
    biến1 = row["tên_cột1"],
    biến1 = row["tên_cột2"],
    ...
}
```

#### 4.4. BÀI TẬP CHƯƠNG 3

**Bài 1:** Tạo ứng dụng Window form C# với MongoDB có các yêu cầu sau:

	MANV	HOTEN	TUOI
▶ 1	NV001	Trần Văn Minh	28
2	NV002	Đỗ Thị Bình	26
3	NV003	Tạ Thị Bình	32
4	NV004	Nguyễn Trí Tường	40

1/ Tạo một cơ sở dữ liệu MongoDB gồm collection NhanVien có cấu trúc như bên dưới. Thêm ít nhất 3 document vào collection này.

```
{
  MANV: Mã nhân viên,
  HOTEN: Họ tên nhân viên,
  TUOI: tuổi, số nguyên,
  NGOAINGU: [ mảng các ngoại ngữ ],
  PHONGBAN: { MAPH: mã phòng, TENPH: tên phòng },
  DEAN:
    [ { MADA: mã đề án, TENDA: tên đề án },
      ...
    ]
}
```

2/ Tạo form giao diện với các đối tượng điều khiển và chức năng như hình trên. Trong đó Mã nhân viên, Họ tên, Tuổi, Phòng ban, Ngoại ngữ là các Textbox; danh sách đề án sử dụng đối tượng Listbox.

3/ Cài đặt chức năng load dữ liệu từ database lên form

- Thiết lập kết nối database
- Load dữ liệu lên form sau khi khởi động

4/ Cài đặt các chức năng nhập một document từ form.



- Khi click vào nút Nhập mới sẽ xóa trống các textbox: Mã nhân viên, Họ tên, Tuổi, Phòng ban, Ngoại ngữ và listbox Đề án.
- Khi sau khi nhập đầy đủ thông tin, click nút Thêm sẽ thực hiện thêm document vào cơ sở dữ liệu.

5/ Cài đặt chức năng xóa: khi click nút xóa sẽ hiển thị thông báo xác nhận xóa, sau đó xóa đúng document đang chọn trên form.

6/ Cài đặt chức năng sửa:

- Khi click vào nút Sửa sẽ cập nhật tất cả các thông tin sửa trên form xuống database.
- Khi click vào nút sửa bên phải các Textbox: Tuổi, phòng ban, ngoại ngữ và listbox đề án sẽ sửa thông tin tương ứng vào database.

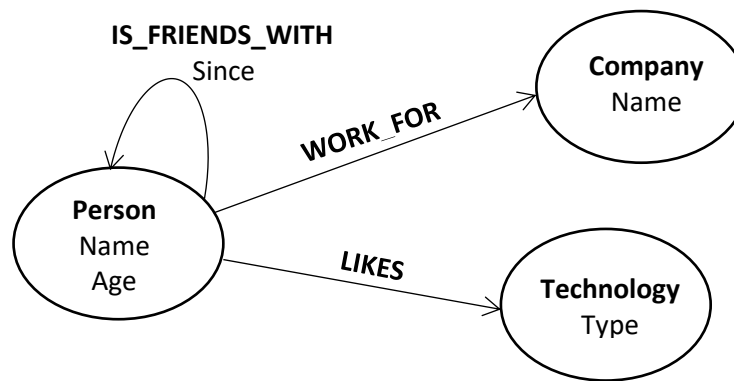
7/ Cài đặt sự kiện xử lý khi click vào mỗi dòng trong datagridview sẽ hiển thị thông tin tương ứng trên các textbox và listbox.

**Bài 2:** Tạo ứng dụng Java với Neo4J có các yêu cầu sau:

The screenshot shows a Java SWT Application window titled "SWT Application". It contains a form with three main sections for creating and managing data in a Neo4J database:

- Top Section:**
  - Label: "Nhân nút" (Create Node)
  - Dropdown menu: "Person" (with a downward arrow)
  - Text input field: "Thuộc tính" (Property)
  - Button: "Tạo nút" (Create Node)
- Middle Section:**
  - Label: "Chọn nút" (Select Node)
  - Two dropdown menus: "Lan" and "Bình" (both with downward arrows)
  - Text input field: "Liên kết" (Relationship)
  - Dropdown menu: "IS\_FRIENDS\_WITH" (with a downward arrow)
  - Text input field: "Thuộc tính" (Property)
  - Button: "Tạo liên kết" (Create Relationship)
- Bottom Section:**
  - Label: "Chọn nút, liên kết" (Select Node, Relationship)
  - Two dropdown menus: "Lan" and "IS\_FRIENDS\_WITH" (both with downward arrows)
  - Text input field: "Thuộc tính" (Property)
  - Button: "Tìm kiếm" (Search)

1/ Tạo cơ sở dữ liệu đồ thị theo mô hình bên dưới. Thêm vào cơ sở dữ liệu 3 nút thuộc nhãn Person, 2 nút thuộc nhãn Company và 1 nút thuộc nhãn Technology với các liên kết và thuộc tính phù hợp.



2/ Tạo form giao diện ứng dụng java SWT trên Eclipse hoặc tương đương với các điều khiển tương ứng: Text, Combo, Button như hình trên.

3/ Cài đặt chức năng load dữ liệu từ database lên form

- Thiết lập kết nối database
- Load dữ liệu lên form sau khi khởi động. Trong đó:
  - Combo Nhận nút hiển thị tất cả các nhân nút có trong cơ sở dữ liệu. Text Tính: hiển thị các thuộc tính của nhân tương ứng, mỗi thuộc tính cách nhau bởi dấu phẩy.
  - Combo Chọn nút hiển thị danh sách tên nút ứng với nhân ở phần trên.
  - Combo Liên kết: Hiển thị danh sách tất cả các nhân liên kết. Text Thuộc tính: hiển thị thuộc tính của liên kết tương ứng. Nếu có nhiều thuộc tính thì cách nhau bởi dấu phẩy.

4/ Cài đặt chức năng tạo nút: click Tạo nút sẽ tạo một nút mới với nhân và các thuộc tính tương ứng lưu vào database.

5/ Cài đặt chức năng tạo liên kết: chọn 2 nút cần tạo mối liên kết, chọn liên kết cần tạo và nhập các thuộc tính. Click nút tạo liên kết sẽ tạo liên kết giữa 2 nút.

6/ Tìm kiếm: chọn nút và liên kết, click vào Tìm kiếm sẽ hiển thị thông tin những nút với liên kết vừa chọn.

**Bài 3:** Tạo ứng dụng Window form C# với Cassandra có các yêu cầu sau:

1/ Tạo một cơ sở dữ liệu Cassandra gồm hai bảng *nhanvien* và *phancong* có cấu trúc như sau:

Bảng *nhanvien*

Tên cột	Kiểu dữ liệu	Khóa
manv	text	Khóa chính
hoten	text	
ngaysinh	text	
phai	text	

dienthoai	text	
email	text	
maphong	text	

Bảng *phancong*

Tên cột	Kiểu dữ liệu	Khóa
manv	text	Khóa chính
mada	text	
tenda	text	
ngaypc	timestamp	
sogio	int	

2/ Tạo form giao diện với các đối tượng điều khiển label, textbox, button, datagridview và thiết lập vị trí như hình sau.

	Mã NV	Họ tên	Ngày sinh	Phái	Điện thoại	Email	Phòng
1	NV001	Đỗ Thị Lan	1990-04-23	Nữ	0914045226	landt@gmail.com	P01
2	NV002	Võ Minh Trung	1982-08-15	Nam	0904591163	trungvm@gmail.com	P01
3	NV003	Lê Thị Mỹ Ngọc	1992-05-14	Nữ	0376578895	ngoc@gmail.com	P02
4	NV004	Trần Văn Bình	1992-03-24	Nam	0982556227	binh@gmail.com	P02
5	NV005	Đỗ Thị Lại	1983-05-26	Nữ	0902234888	thanhdtd@gmail.com	P03
6	NV006	Trương Hữu Chí	1977-05-18	Nam	0964435675	chi@gmail.com	P03

3/ Cài đặt chức năng load dữ liệu từ database Cassandra lên form

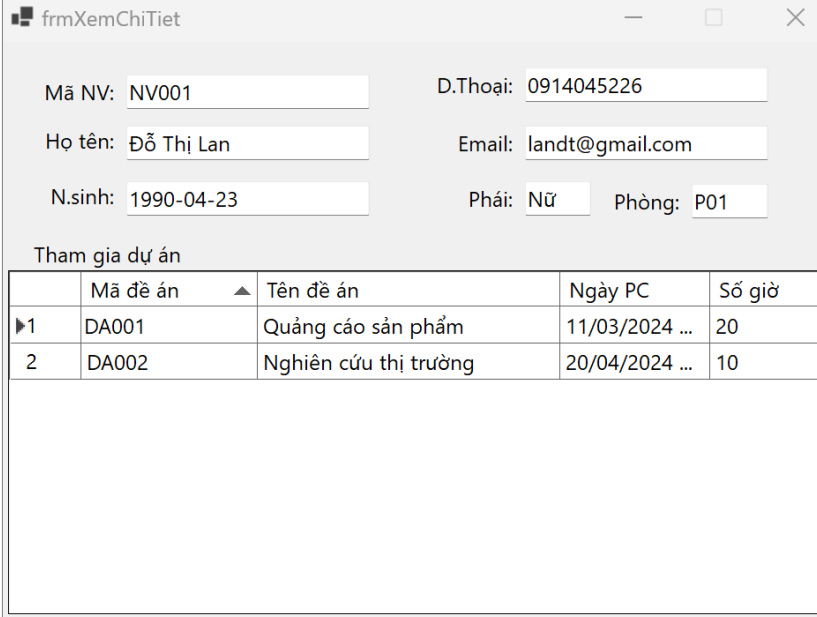
- Thiết lập kết nối database
- Load dữ liệu lên form sau khi khởi động

4/ Cài đặt các chức năng thêm, xóa, sửa, tìm kiếm dữ liệu nhân viên.

- Thêm: nhận đầy đủ thông tin vào các textbox bên trái, nhấn nút Thêm sẽ thêm thông tin nhân viên vào database đồng thời hiển thị thông tin nhân viên mới trên form. Hiển thị thông báo thêm thành công.

- Sửa: Chỉ sửa các thông tin khác ngoài Mã nhân viên. Khi nhấn nút sửa sẽ lưu các thông tin sửa của nhân viên được chọn xuống database. Hiện thị thông báo sửa thành công.
- Xóa: Chọn nhân viên cần xóa trên datagridview, click nút Xóa sẽ hiển thị cảnh báo hỏi người dùng có chắc muốn xóa hay không. Nếu chọn Yes thì sẽ xóa thông tin nhân viên được chọn.
- Tìm kiếm: Nhập mã nhân viên cần tìm vào ô tìm kiếm. Click nút Tìm kiếm sẽ tìm nhân viên thỏa điều kiện và load lên datagridview.

5/ Cài đặt chức năng Xem chi tiết: khi click nút Xem chi tiết sẽ hiển thị thông tin các đề án mà nhân viên được chọn có tham gia như sau:



The screenshot shows a Windows form titled "frmXemChiTiet". It contains several text boxes for employee information: "Mã NV:" (NV001), "D.Thoại:" (0914045226), "Họ tên:" (Đỗ Thị Lan), "Email:" (landt@gmail.com), "N.sinh:" (1990-04-23), "Phái:" (Nữ), and "Phòng:" (P01). Below these is a section titled "Tham gia dự án" containing a table with 5 columns: an index, "Mã đề án", "Tên đề án", "Ngày PC", and "Số giờ". The table lists two projects: DA001 (Quảng cáo sản phẩm) and DA002 (Nghiên cứu thị trường).

	Mã đề án	Tên đề án	Ngày PC	Số giờ
1	DA001	Quảng cáo sản phẩm	11/03/2024 ...	20
2	DA002	Nghiên cứu thị trường	20/04/2024 ...	10