

TRƯỜNG ĐẠI HỌC SƯ PHẠM THÀNH PHỐ HỒ CHÍ MINH
Khoa Công nghệ thông tin



BÁO CÁO ĐỀ TÀI CUỐI KÌ TÌM HIỂU MONGODB

Môn: Cơ sở dữ liệu nâng cao

GV hướng dẫn: ThS. Lương Trần Hy Hiến

Nhóm 5

Nguyễn Văn An	43.01.104.002
Võ Thế Duy	43.01.104.032
Nguyễn Minh Pháp	43.01.104.124
Châu Bảo Thịnh	43.01.104.169
Trần Minh Trường	43.01.104.192
Nguyễn Doãn Tứ	43.01.104.196

Mục lục

1. MongoDB.	2
1.1 Giới thiệu MongoDB.....	2
1.2 Ưu nhược điểm của MongoDB.	3
1.3 Khi nào thì nên sử dụng MongoDB?	4
2. Cài đặt và cấu hình MongoDB.....	5
2.1 Tải và cài đặt.....	5
2.2 Cấu hình MongoDB.	9
3. Truy vấn dữ liệu trong MongoDB.	13
3.1 Database MongoDB	14
3.2 MongoDB - Create/Drop Collection	16
3.3 Một số toán tử truy vấn so sánh trong MongoDB	18
3.4 MongoDB - Insert	19
3.5 MongoDB - Find.....	21
3.6 MongoDB - Update.....	24
3.7 MongoDB - Remove	26
3.8 Embedded/Nested Documents	27
3.8.1 Embedded Documents là gì?	27
3.8.2 Truy vấn Embedded Documents.....	28
4. Truy vấn dữ liệu trên C# .Net.....	31
4.1 Các gói NuGet cần thiết.....	31
4.2 Tạo kết nối đến server.....	31
4.3 Các câu truy vấn cơ bản	31

1. MongoDB.

1.1 Giới thiệu MongoDB.

MongoDB là một mã nguồn mở và là một tập tài liệu dùng cơ chế NoSQL để truy vấn, nó được viết bởi ngôn ngữ C++. Chính vì được viết bởi C++ nên nó có khả năng tính toán với tốc độ cao chứ không giống như các hệ quản trị CSDL hiện nay.

Mỗi một table (bảng dữ liệu) trong SQL sử dụng thì trong MongoDB gọi là collection (tập hợp). Một bản ghi của MongoDB được lưu trữ dưới dạng document (tài liệu), nó được ghi xuống với cấu trúc field (trường) và value (giá trị). Nó giống như là một đối tượng JSON có dạng như sau:

```
{ name : "Joe", age : 30, interests : 'football' }
```

```
{ name : "Kate", age : 25 }
```

Nhìn vào tập data, bạn có thể biết được cấu trúc (schema) của collection này. Nhưng do không có một cấu trúc ràng buộc định nghĩa sẵn từ trước nên document trong collection có thể rất tùy ý và nếu chẳng may application lưu thêm một document không có liên quan về thông tin như:

```
{ name : "Joe", age : 30, interests : 'football' }
```

```
{ name : "Kate", age : 25 }
```

```
{ phone : "iphone", ios : 6 }
```

thì mongodb cũng chấp nhận. Tuy vậy, nhìn ở khía cạnh bất khả kê hơn thì không có ràng buộc lại đem đến cho mongodb khả năng linh động uyển chuyển trong cấu trúc của nó. Agile development đang là xu hướng hiện nay. Với các lợi thế linh động uyển chuyển, mongodb rất thích hợp với các nhu cầu thay đổi requirement thường xuyên.

☞ Hiệu suất cao: Hỗ trợ nhúng dữ liệu dạng mô hình dữ liệu giúp giảm thiểu hoạt động của server, Truy vấn dữ liệu sử dụng chỉ mục giúp tối ưu tốc độ truy vấn.

☞ Dễ dàng tăng tính mở rộng: chế sharding tự động (tự động phân vùng dữ liệu trên máy chủ, động bộ hóa dữ liệu tốt).

*Sự tương quan các thuật ngữ được sử dụng trong MongoDB và SQL

Operation	In SQL	In MongoDB
Create	Insert	Insert
Read	Select	Find
Update	Update	Update
Delete	Delete	Remove
Table	Table	Collection
Row	Row	Document

1.2 Ưu nhược điểm của MongoDB.

*Ưu điểm

Open Source:

- MongoDB là phần mềm mã nguồn mở miễn phí, có cộng đồng phát triển rất lớn

Hiệu năng cao:

- Tốc độ truy vấn (find, update, insert, delete) của MongoDB nhanh hơn hẳn so với các hệ quản trị cơ sở dữ liệu quan hệ (RDBMS).
- Thử nghiệm cho thấy tốc độ insert, tốc độ insert của MongoDB có thể nhanh tới gấp 100 lần so với MySQL.

Tại sao MongoDB có hiệu năng cao như thế? có các lý do sau:

- MongoDB lưu dữ liệu dạng JSON, khi bạn insert nhiều đối tượng thì nó sẽ là insert một mảng JSON gần như với trường hợp insert 1 đối tượng
- Dữ liệu trong MongoDB không có sự ràng buộc lẫn nhau như trong RDBMS, khi insert, xóa hay update nó không cần phải mất thời gian kiểm tra xem có thỏa mãn các bảng liên quan như trong RDBMS.
- Dữ liệu trong MongoDB được đánh chỉ mục (đánh index) nên khi truy vấn nó sẽ tìm rất nhanh.
- Khi thực hiện insert, find... MongoDB sẽ khóa các thao tác khác lại, ví dụ khi nó thực hiện find(), trong quá trình find mà có thêm thao tác insert, update thì nó sẽ dừng hết lại để chờ find() xong đã.

Dữ liệu linh hoạt:

- MongoDB là document database, dữ liệu lưu dưới dạng JSON, không bị bó buộc về số lượng field, kiểu dữ liệu... bạn có thể insert thoải mái dữ liệu mà mình muốn.

Là Rich Query Language:

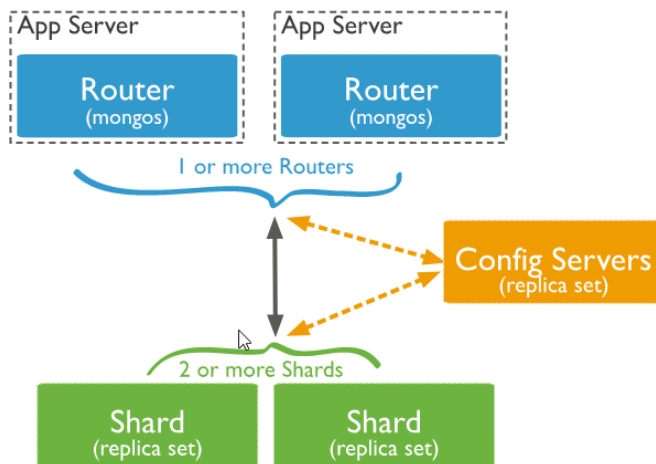
- MongoDB là một rich query language tức là nó có sẵn các method để thực hiện create, read, update, delete dữ liệu (CRUD)

Tính sẵn có:

- MongoDB hỗ trợ replica set nhằm đảm bảo việc sao lưu và khôi phục dữ liệu

Khả năng mở rộng Horizontal Scalability:

- Trong MongoDB có một khái niệm cluster là cụm các node chứa dữ liệu giao tiếp với nhau, khi muốn mở rộng hệ thống ta chỉ cần thêm một node với vào cluster:



*Nhược điểm

- MongoDB không có các tính chất ràng buộc như trong RDBMS → dễ bị làm sai dữ liệu
- Không hỗ trợ join giống như RDBMS nên khi viết function join trong code ta phải làm bằng tay khiến cho tốc độ truy vấn bị giảm.
- Sử dụng nhiều bộ nhớ: do dữ liệu lưu dưới dạng key-value, các collection chỉ khác về value do đó key sẽ bị lặp lại. Không hỗ trợ join nên sẽ bị dư thừa dữ liệu (trong RDBMS thì ta chỉ cần lưu 1 bản ghi rồi các bản ghi khác tham chiếu tới còn trong MongoDB thì không)
- Bị giới hạn kích thước bản ghi: mỗi document không được có kích thước > 16Mb và không mức độ các document con trong 1 document không được > 100

1.3 Khi nào thì nên sử dụng MongoDB?

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

MongoDB tốt cho:

- Danh mục sản phẩm thương mại điện tử.
- Blog và quản lý nội dung.
- Phân tích thời gian thực và ghi nhật ký tốc độ cao, bộ nhớ đệm và khả năng mở rộng cao.
- Quản lý cấu hình.
- Duy trì dữ liệu dựa trên vị trí – Dữ liệu không gian địa lý.
- Các trang web di động và mạng xã hội.
- Phát triển yêu cầu dữ liệu.
- Mục tiêu không chặt chẽ – thiết kế có thể thay đổi theo thời gian.

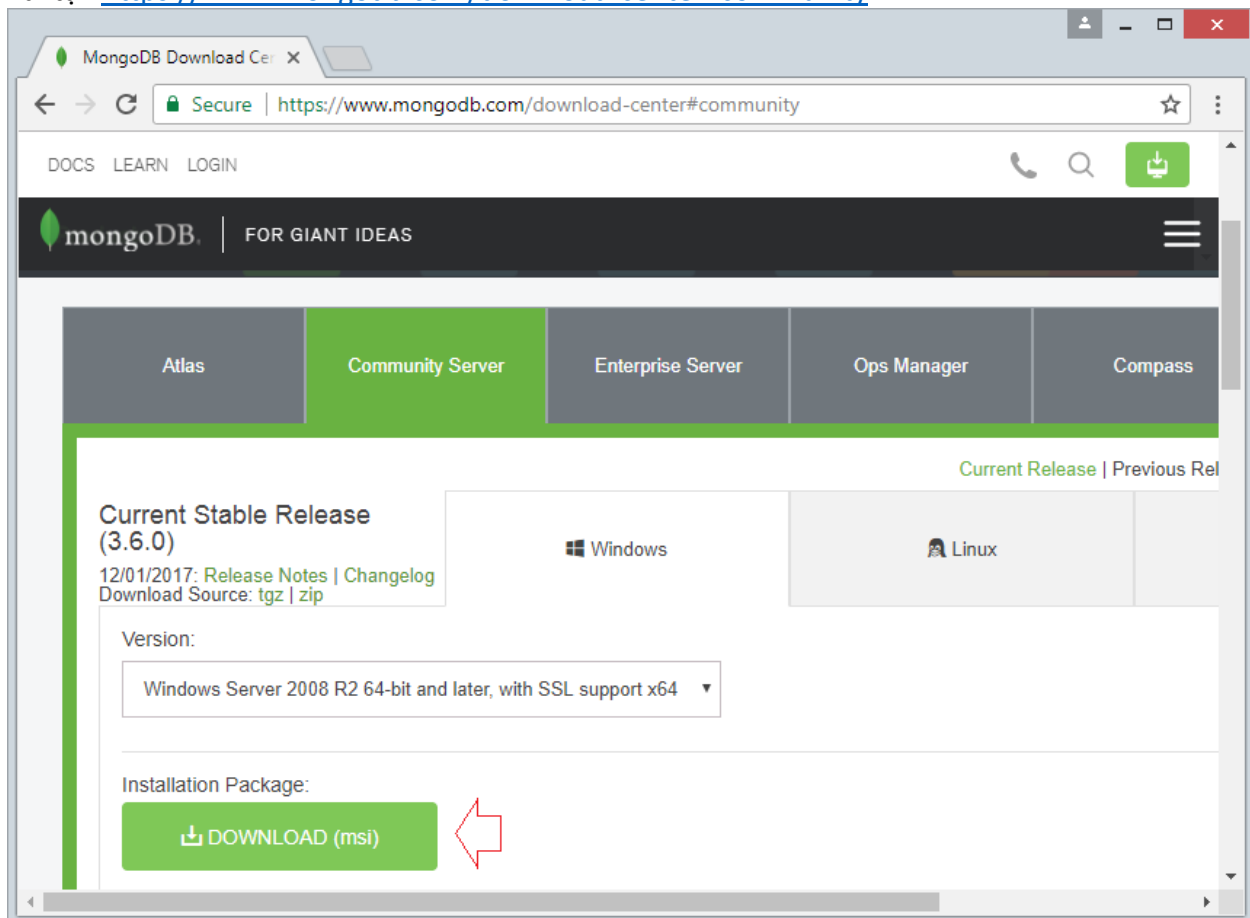
MongoDB không tốt cho:

- Hệ thống giao dịch cao hoặc nơi mô hình dữ liệu được thiết kế trước.
- Hệ thống kết hợp chặt chẽ.

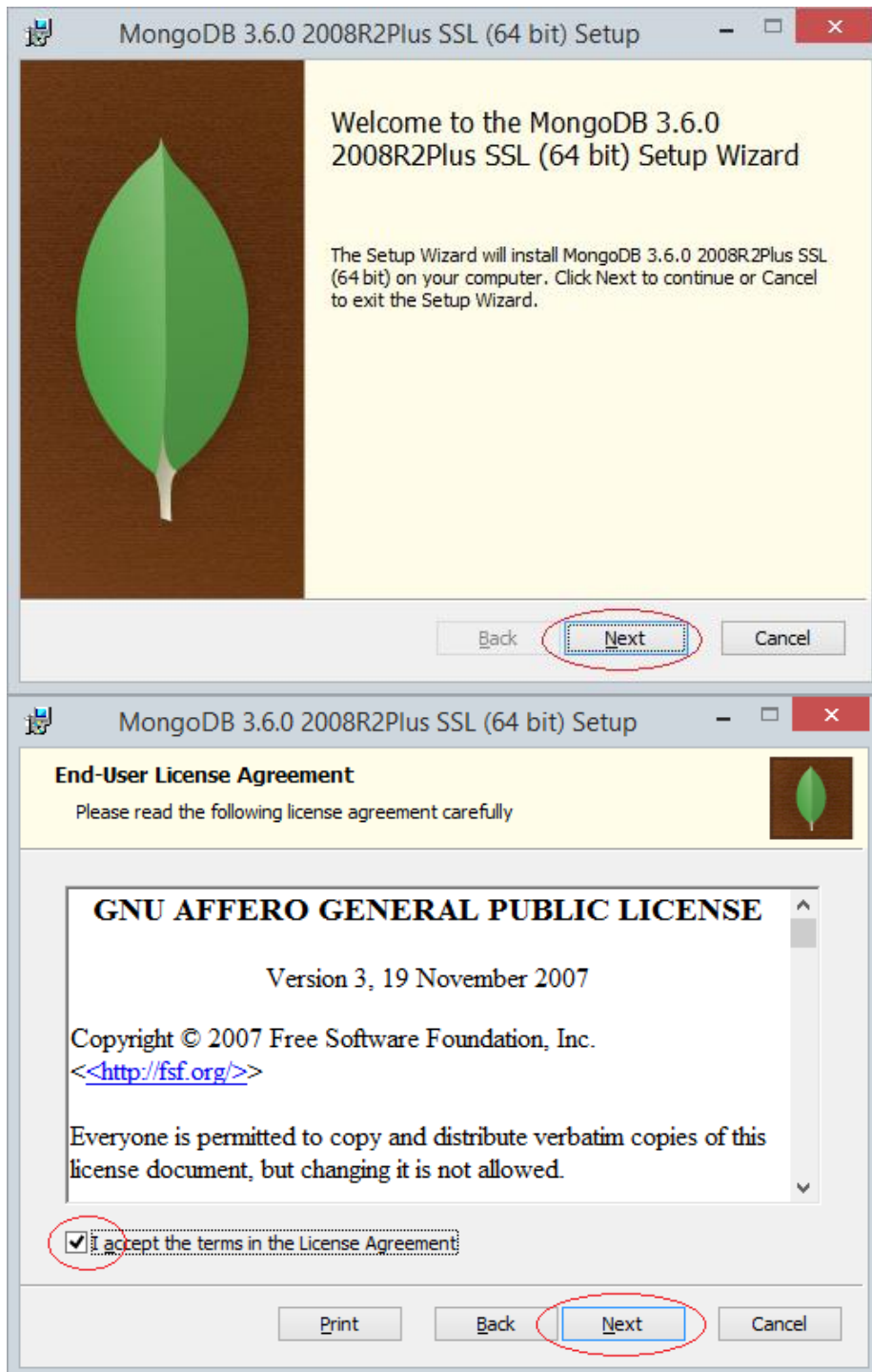
2. Cài đặt và cấu hình MongoDB.

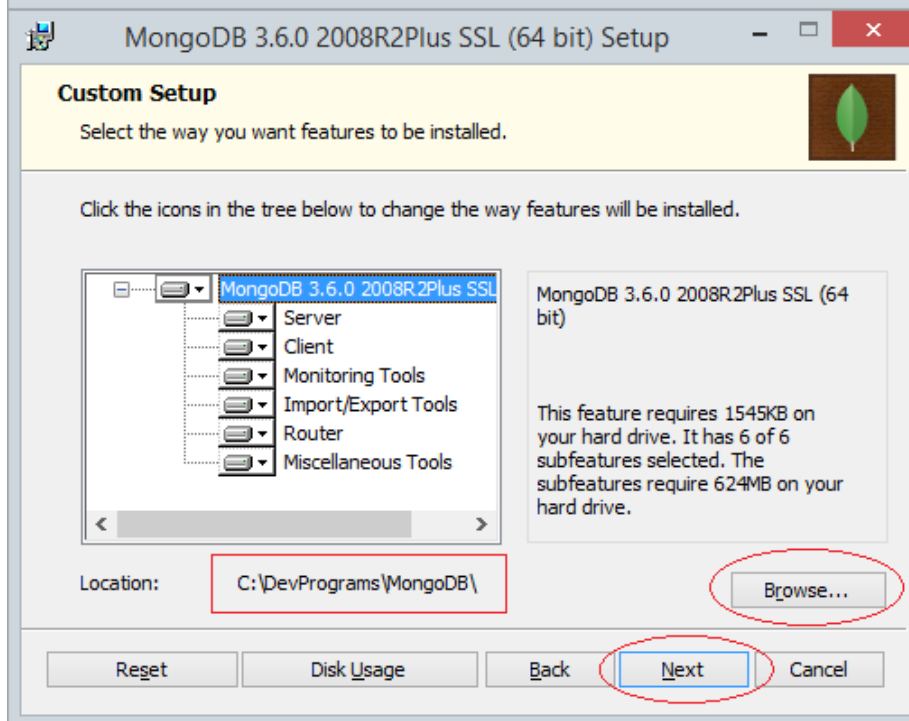
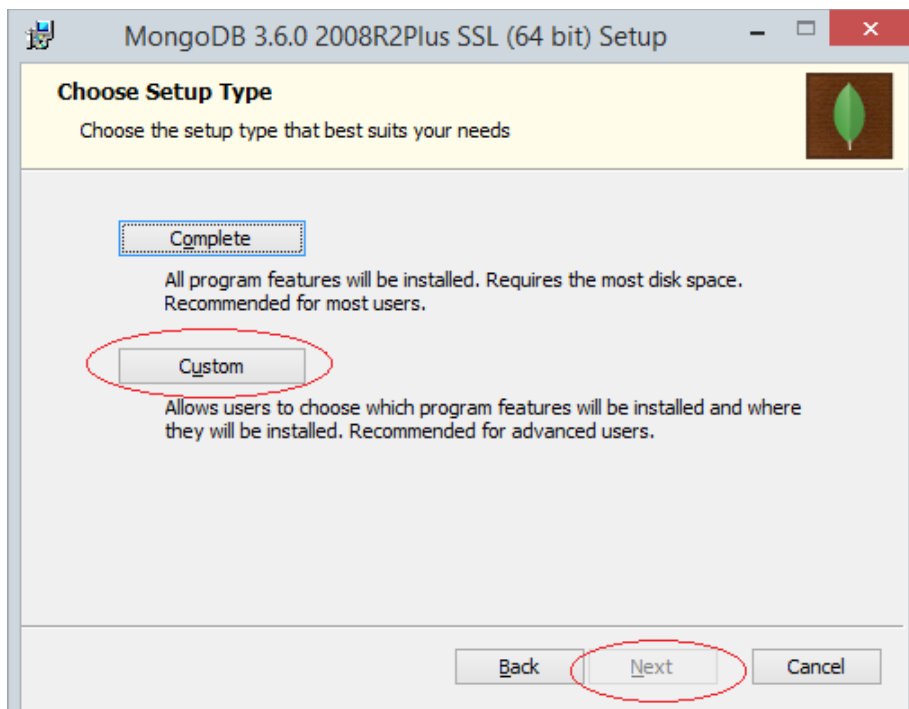
2.1 Tải và cài đặt.

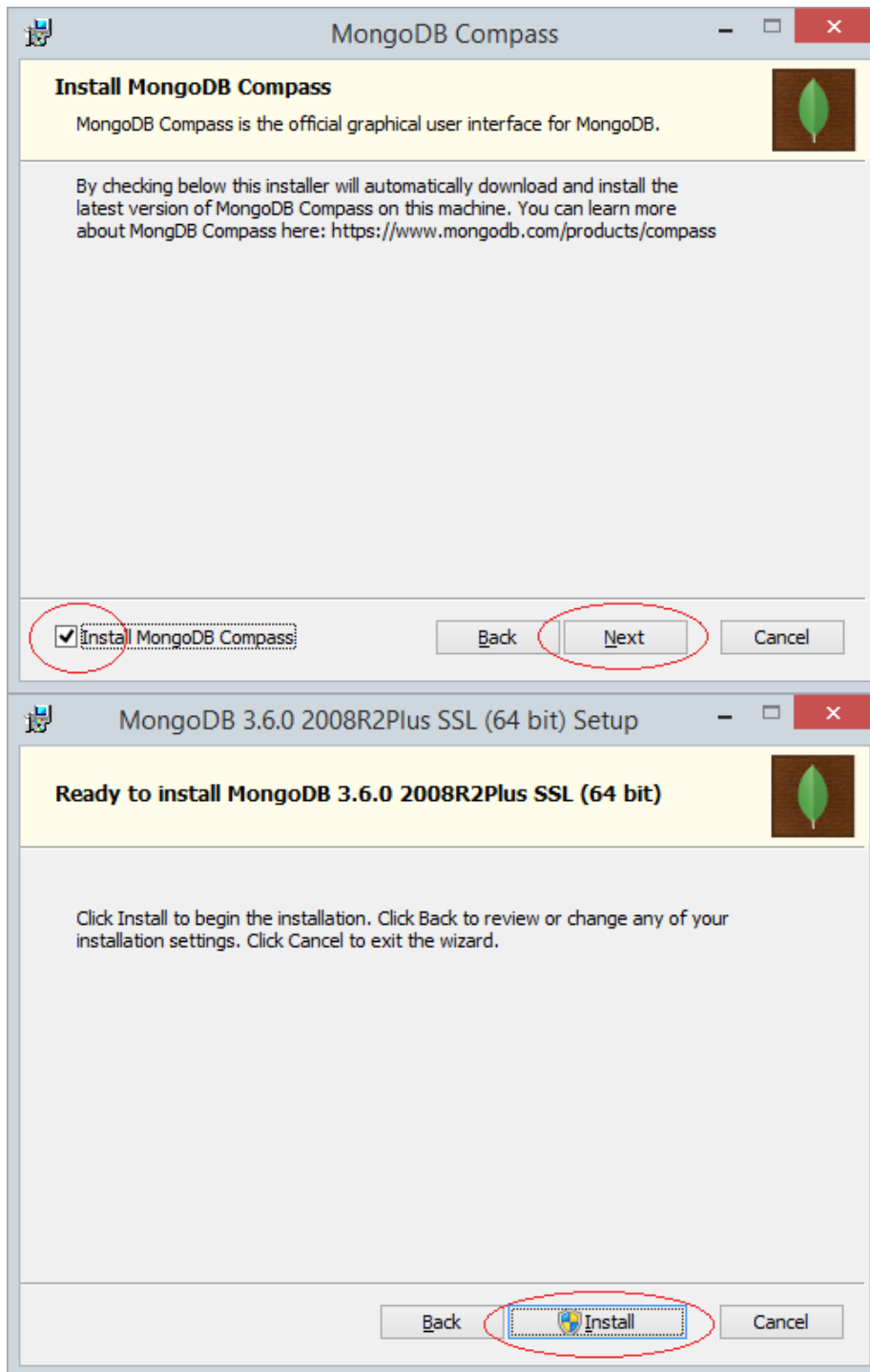
Tải tại: <https://www.mongodb.com/download-center#community>

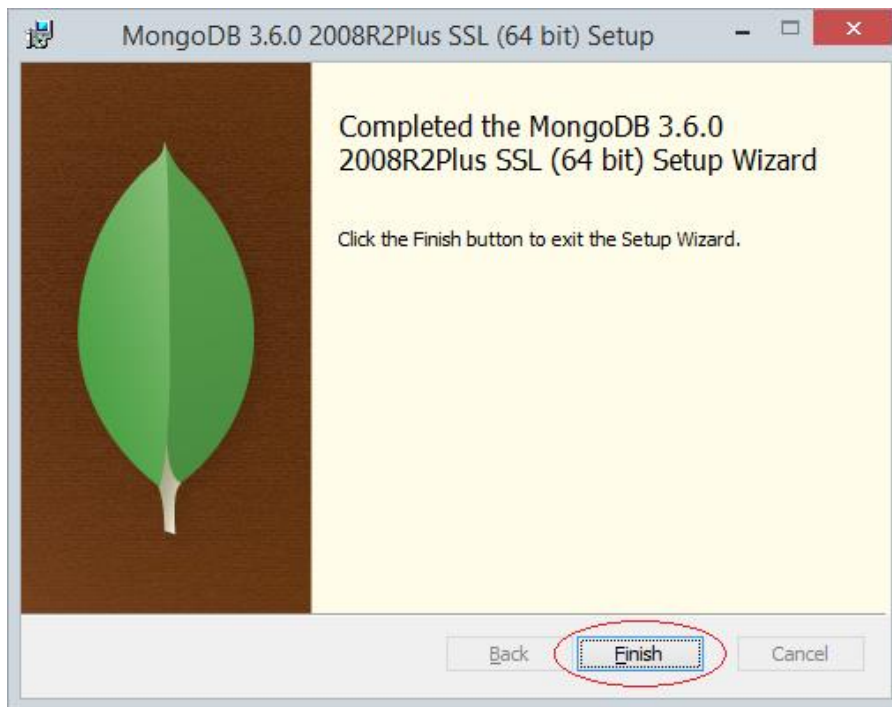


Các bước cài đặt như sau:



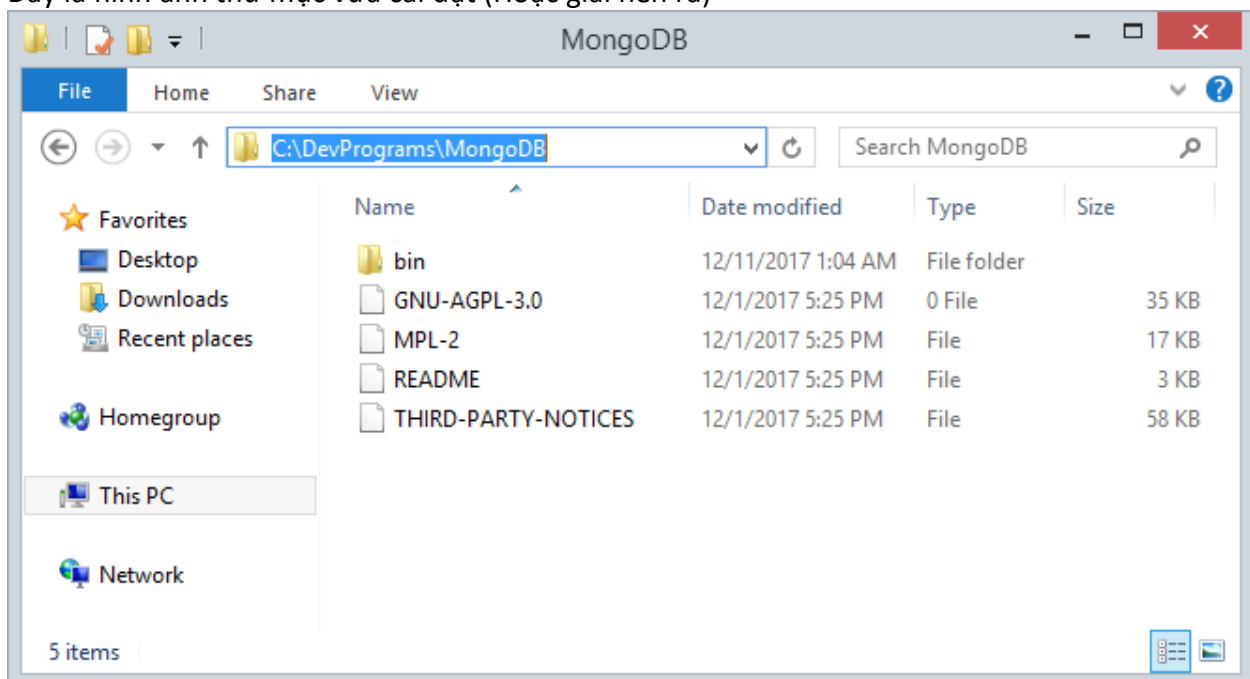






2.2 Cấu hình MongoDB.

Đây là hình ảnh thư mục vừa cài đặt (Hoặc giải nén ra)



Chúng ta tạo một thư mục chứa dữ liệu của Database, thư mục chứa file log, file cấu hình,....
Tốt nhất hãy để nó ở một ổ an toàn, không nên để ở ổ C.

Tạo thư mục:

E:/MongoStore

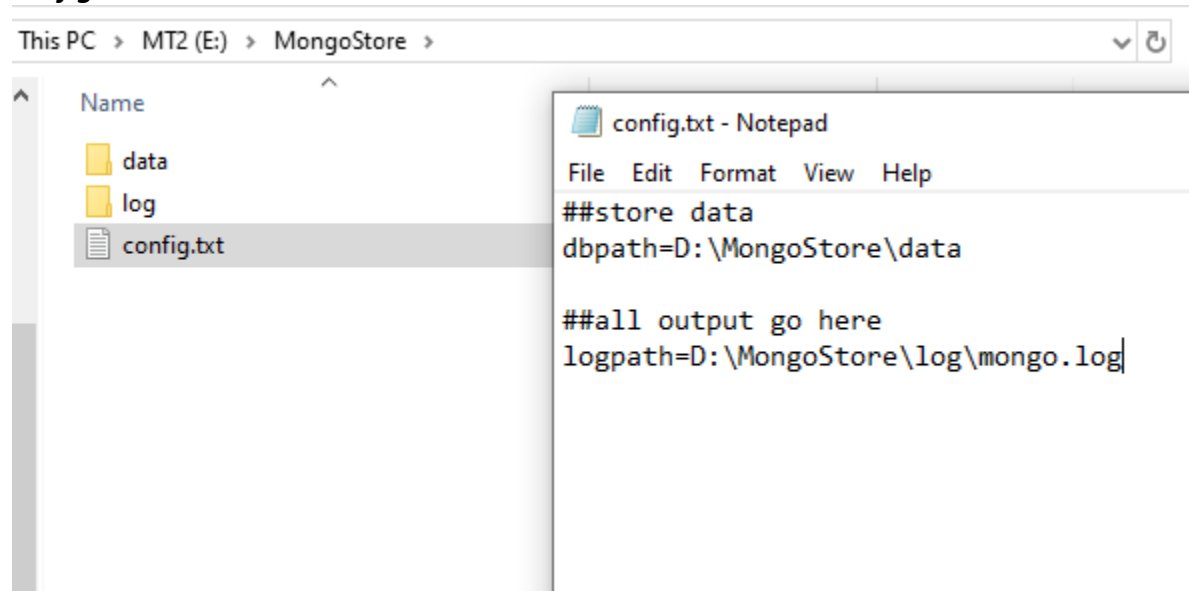
Sau đó tạo 2 thư mục con

data

log

Và một file cấu hình:

config.txt



config.txt

```
##store data
dbpath=E:\MongoStore\data

##all output go here
logpath=E:\MongoStore\log\mongo.log
```

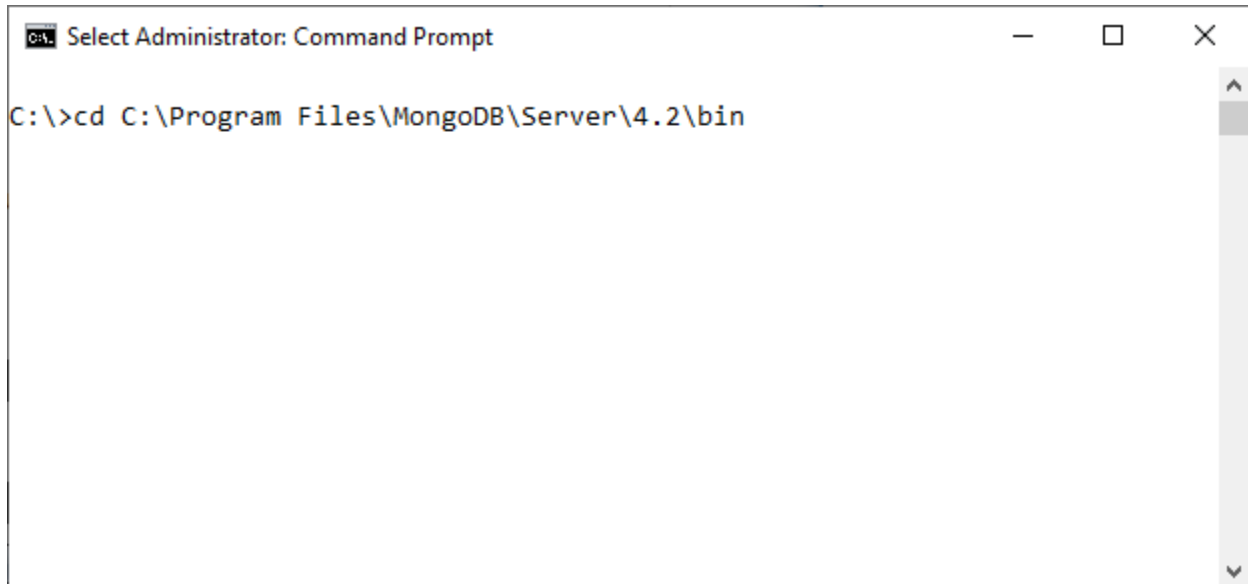
Tới đây việc cấu hình **MongoDB** thành công.

Chạy MongoDB

Ở đây chúng ta chạy **MongoDB** bằng lệnh **CMD**, bạn có thể tạo một Service (dịch vụ) của Windows để nó tự động khởi động (Start) mỗi khi Windows chạy.

Mở CMD và CD tới thư mục **bin** của **mongodb**.

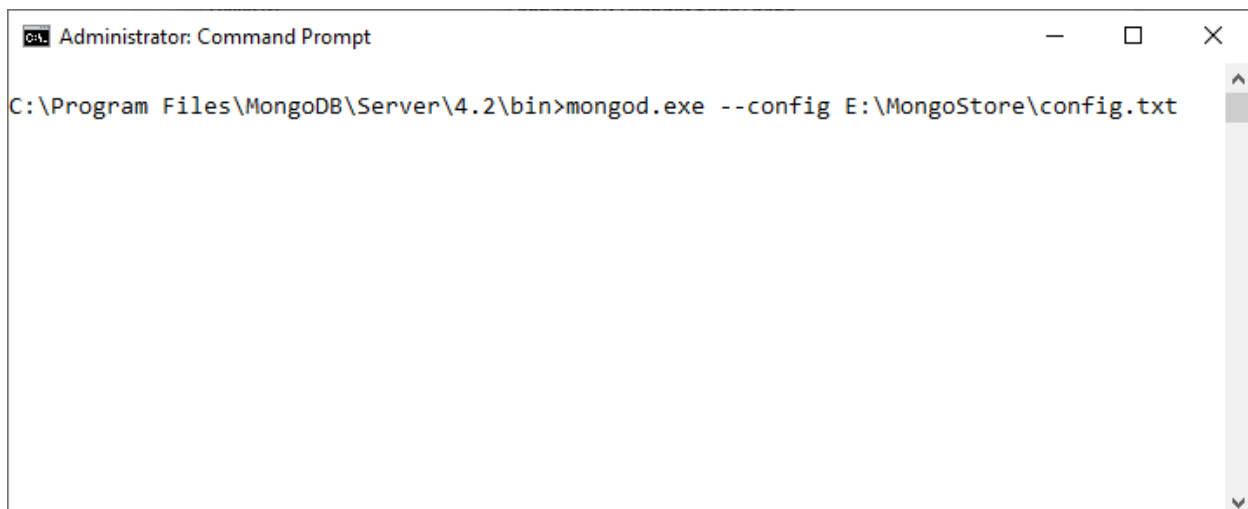
```
cd C:\xxxxx\MongoDB\bin (Một số phiên bản sau này sẽ có đường dẫn khác)
```



```
C:\>cd C:\Program Files\MongoDB\Server\4.2\bin
```

Chạy lệnh sau để khởi động (start) **mongodb**:

```
mongod.exe --config E:\MongoStore\config.txt
```



```
C:\Program Files\MongoDB\Server\4.2\bin>mongod.exe --config E:\MongoStore\config.txt
```

Kết quả:

This PC > MT2 (E:) > MongoStore > data >

Name	Date modified	Type
diagnostic.data	11/18/2019 13:17	File folder
journal	11/18/2019 13:17	File folder
_mdb_catalog.wt	11/18/2019 13:18	WT File
collection-0--9022599363582263712.wt	11/18/2019 13:18	WT File
collection-2--9022599363582263712.wt	11/18/2019 13:18	WT File
collection-4--9022599363582263712.wt	11/18/2019 13:17	WT File
index-1--9022599363582263712.wt	11/18/2019 13:18	WT File
index-3--9022599363582263712.wt	11/18/2019 13:18	WT File
index-5--9022599363582263712.wt	11/18/2019 13:17	WT File
index-6--9022599363582263712.wt	11/18/2019 13:18	WT File
mongod.lock	11/18/2019 13:17	LOCK File
sizeStorer.wt	11/18/2019 13:17	WT File
storage.bson	11/18/2019 13:17	BSON File
WiredTiger	11/18/2019 13:17	File
WiredTiger.lock	11/18/2019 13:17	LOCK File
WiredTiger.wt	11/18/2019 13:18	WT File

3. Truy vấn dữ liệu trong MongoDB.

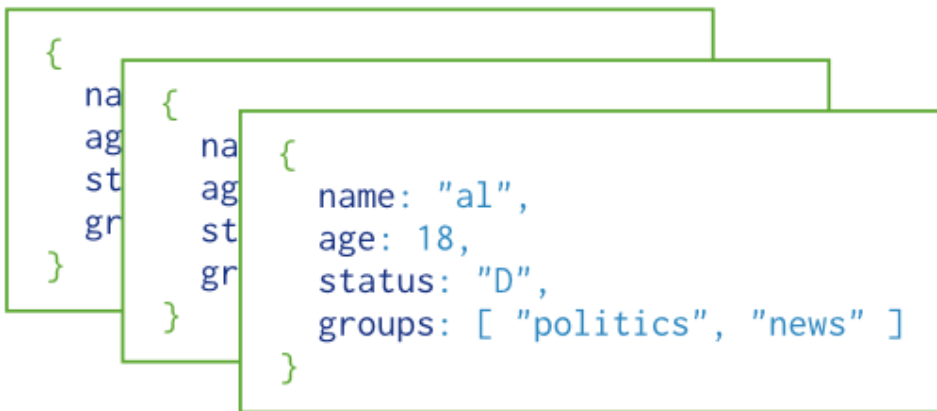
Trước khi tìm hiểu các câu lệnh, cùng bắt đầu với những khái niệm cơ bản:

Database

Database là một container vật lý cho các collection. Mỗi DB được thiết lập cho riêng nó một danh sách các files hệ thống files. Một máy chủ MongoDB đơn thường có nhiều DB.

Collection

Collection là một nhóm các documents của MongoDB. Nó tương đương với một table trong RDBMS. Một Collection tồn tại trong một cơ sở dữ liệu duy nhất. Các collection ko tạo nên một schema. Documents trong collection có thể có các fields khác nhau. Thông thường, tất cả các documents trong collections có mục đích khá giống nhau hoặc liên quan tới nhau



Collection

Document

Một document là một tập hợp các cặp key-value. Documents có schema động. Schema động có nghĩa là documents trong cùng một collection không cần phải có cùng một nhóm các fields hay cấu trúc giống nhau, và các fields phổ biến trong các documents của collection có thể chứa các loại dữ liệu khác nhau.

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

← field: value
← field: value
← field: value
← field: value

Bảng dưới đây cho thấy mối quan hệ của các thuật ngữ RDBMS với MongoDB

RDBMS	MongoDB
Database	Database
Table	Collection
Tuple/Row	Document
column	Field
Table Join	Embedded Documents
Primary Key	Primary Key (Default key _id provided by mongodb itself)
Database Server and Client	
Mysqld/Oracle	mongod
mysql/sqlplus	mongo

Các câu lệnh thao tác với cơ sở dữ liệu:

Mở màn hình CMD, trở tới thư mục bin của MongoDB rồi gõ lệnh sau

```
mongo
```

3.1 Database MongoDB

```
>use DATABASE_NAME
```

```
Administrator: Command Prompt - mongo
> use MongoStore
switched to db MongoStore
>
```

Check đang sử dụng DB nào

```
>db
```

```
Administrator: Command Prompt - mongo
> db
MongoStore
>
```

Show danh sách các DB trong server.

```
>show dbs
Administrator: Command Prompt - mongo
> show dbs
MongoStore  0.000GB
admin       0.000GB
config      0.000GB
local       0.000GB
>
```

Xóa database

```
>db.dropDatabase()
Administrator: Command Prompt - mongo
> db.dropDatabase()
{ "dropped" : "MongoStore", "ok" : 1 }
> show dbs
admin  0.000GB
config 0.000GB
local  0.000GB
>
```


3.2 MongoDB - Create/Drop Collection

Dưới đây là syntax để tạo một Collection

```
db.createCollection("name")
```

Và syntax để xóa một Collection

```
db.COLLECTION_NAME.drop()
```

Ví dụ:

```
Administrator: Command Prompt - mongo
> db.createCollection("giaovien")
{ "ok" : 1 }
> db.giaovien.drop()
true
>
```

Trong MongoDB thì không nhất thiết phải tạo collection. Nó sẽ tự động tạo collection cho người dùng khi insert document.

```
Administrator: Command Prompt - mongo
> show collections
> db.giaovien.insert({msgv:"gv001"})
WriteResult({ "nInserted" : 1 })
> show collections
giaovien
>
```

Kiểu dữ liệu

- String: String trong MongoDB phải là UTF-8 hợp lệ.
- Integer: Số nguyên có thể là 32 bit hoặc 64 bit tùy thuộc vào máy chủ của bạn.
- Boolean
- Double
- Min/ Max keys: Loại này được sử dụng để so sánh giá trị đối với các yếu tố thấp nhất và cao nhất BSON.
- Array
- Timestamp
- Object
- Null
- Symbol

- Date
- Object ID
- Binary data
- Code
- Regular expression

3.3 Một số toán tử truy vấn so sánh trong MongoDB

\$eq trả về các document trong đó giá trị bằng với một giá trị được chỉ định

cú pháp : { <field>: { \$eq: <value> } }

\$gt trả về các document trong đó giá trị của trường lớn hơn một giá trị được chỉ định

cú pháp : {field: { \$gt: value } }

\$gte trả về các document trong đó giá trị của trường lớn hơn hoặc bằng 1 giá trị được chỉ định

cú pháp : {field: { \$gte: value } }

\$in trả về các document trong đó giá trị nằm trong mảng được chỉ định

cú pháp : { field: { \$in: [<value1>, <value2>, ... <valueN>] } }

\$lt so sánh các giá trị nhỏ hơn giá trị được chỉ định

cú pháp : {field: { \$lt: value } }

\$lte so sánh các giá trị nhỏ hơn một giá trị được chỉ định

cú pháp : { field: { \$lte: value } }

\$ne so sánh các giá trị không bằng giá trị được chỉ định

cú pháp : {field: { \$ne: value } }

\$nin lấy ra các giá trị không có trong mảng được chỉ định

cú pháp : { field: { \$nin: [<value1>, <value2> ... <valueN>] } }

MongoDB có hỗ trợ chúng ta truy vấn nhiều điều kiện trong 1 lần khai báo với các toán tử AND, OR như trong SQL.

\$and

- thực hiện một logic AND hoạt động trên một mảng của một hoặc nhiều biểu thức và chọn ra các tài liệu đáp ứng tất cả các biểu thức trong mảng. cú pháp :

{ \$and: [{ <expression1> }, { <expression2> }, ... , { <expressionN> }] }

3.4 MongoDB - Insert

Insert một Document

```
>db.COLLECTION_NAME.insertOne(document)
```

hoặc

```
>db.COLLECTION_NAME.insert(document)
```

Ví dụ

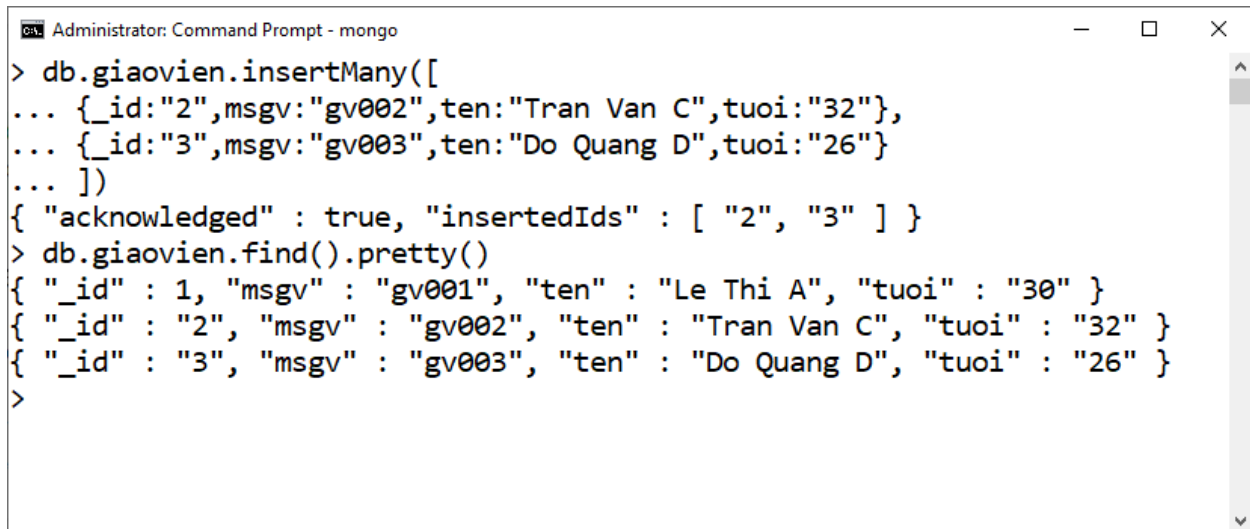
```
Administrator: Command Prompt - mongo
> db.giaovien.insertOne({msgv:"gv001",ten:"Le Thi A",tuoi:"30"})
{
  "acknowledged" : true,
  "insertedId" : ObjectId("5dd35d763c27a1813c2fbf89")
}
> db.giaovien.find().pretty()
{
  "_id" : ObjectId("5dd35d763c27a1813c2fbf89"),
  "msgv" : "gv001",
  "ten" : "Le Thi A",
  "tuoi" : "30"
}
>
```

Giá trị của field `_id` được tạo tự động, người dùng có thể khai báo giá trị mình muốn khi insert.

```
Select Administrator: Command Prompt - mongo
> db.giaovien.insertOne({_id:1,msgv:"gv001",ten:"Le Thi A",tuoi:"30"})
{ "acknowledged" : true, "insertedId" : 1 }
> db.giaovien.find().pretty()
{ "_id" : 1, "msgv" : "gv001", "ten" : "Le Thi A", "tuoi" : "30" }
>
```

Nếu bạn muốn insert nhiều documents trong một lệnh, bạn cần cho array vào trong lệnh insert().

```
>db.COLLECTION_NAME.insertMany([
    {document},
    {document}
])
hoặc
>db.COLLECTION_NAME.insert([
    {document},
    {document}
])
```



```
Administrator: Command Prompt - mongo
> db.giaovien.insertMany([
... {_id:"2",msgv:"gv002",ten:"Tran Van C",tuoi:"32"},
... {_id:"3",msgv:"gv003",ten:"Do Quang D",tuoi:"26"}
... ])
{ "acknowledged" : true, "insertedIds" : [ "2", "3" ] }
> db.giaovien.find().pretty()
{ "_id" : 1, "msgv" : "gv001", "ten" : "Le Thi A", "tuoi" : "30" }
{ "_id" : "2", "msgv" : "gv002", "ten" : "Tran Van C", "tuoi" : "32" }
{ "_id" : "3", "msgv" : "gv003", "ten" : "Do Quang D", "tuoi" : "26" }
>
```

3.5 MongoDB - Find

```
>db.COLLECTION_NAME.find()
```

Để hiện thể kết quả đẹp hơn. Bạn có thể dùng thêm phương thức `pretty()`

```
Administrator: Command Prompt - mongo
> db.giaovien.find()
{ "_id" : 1, "msgv" : "gv001", "ten" : "Bich Phuong", "namsinh" : 1995 }
{ "_id" : 2, "msgv" : "gv002", "ten" : "Bao Thy", "namsinh" : 1993 }
{ "_id" : 3, "msgv" : "gv003", "ten" : "Phan Manh Quynh", "namsinh" : 1992 }
{ "_id" : 4, "msgv" : "gv004", "ten" : "Ha Anh Tuan", "namsinh" : 1987 }
>
```

Tìm kiếm thỏa mãn điều kiện cho trước

```
Administrator: Command Prompt - mongo
> db.giaovien.find({namsinh:1995})
{ "_id" : 1, "msgv" : "gv001", "ten" : "Bich Phuong", "namsinh" : 1995 }
>
```

Cách sử dụng find AND trong MongoDB

```
>db.COLLECTION.find({key1:value1, key2:value2})
```

```
Administrator: Command Prompt - mongo
> db.giaovien.find({ten:"Bich Phuong",namsinh:1995})
{ "_id" : 1, "msgv" : "gv001", "ten" : "Bich Phuong", "namsinh" : 1995 }
>
```

Cách sử dụng find OR trong MongoDB

```
>db.mycol.find({$or:[ {key1: value1}, {key2:value2}]})
```

```
Administrator: Command Prompt - mongo
> db.giaovien.find({$or:[{namsinh:1995},{ten:"Phan Manh Quynh"}]})
{ "_id" : 1, "msgv" : "gv001", "ten" : "Bich Phuong", "namsinh" : 1995 }
{ "_id" : 3, "msgv" : "gv003", "ten" : "Phan Manh Quynh", "namsinh" : 1992 }
>
```

Dưới đây là một ví dụ sử dụng cả AND và OR trong MongoDB

```
Administrator: Command Prompt - mongo
> db.giaovien.find({ten:"Bich Phuong",$or:[{namsinh:1995},{msgv:"gv003"}]})
{ "_id" : 1, "msgv" : "gv001", "ten" : "Bich Phuong", "namsinh" : 1995 }
>
```

Tìm kiếm và xuất những field cần xem. Lưu ý là field `_id` mặc định sẽ được hiển thị trừ khi người dùng khai báo trước "`_id:0`", lúc này field `_id` sẽ ẩn đi.

```
Administrator: Command Prompt - mongo
> db.giaovien.find({}, {_id:0, ten:1, namsinh:1})
{ "ten" : "Bich Phuong", "namsinh" : 1993 }
{ "ten" : "Phan Manh Quynh", "namsinh" : 1992 }
{ "ten" : "Ha Anh Tuan", "namsinh" : 1987 }
>
```

Tìm kiếm và xuất những field cần xem đồng thời sắp xếp theo thứ tự tăng dần (1) hoặc giảm dần (-1) của một field nhất định

```
Administrator: Command Prompt - mongo
> db.giaovien.find({}, {_id:0, ten:1, namsinh:1}).sort({namsinh:-1})
{ "ten" : "Bich Phuong", "namsinh" : 1993 }
{ "ten" : "Phan Manh Quynh", "namsinh" : 1992 }
{ "ten" : "Ha Anh Tuan", "namsinh" : 1987 }
>
```

```
Administrator: Command Prompt - mongo
> db.giaovien.find({}, {_id:0, ten:1, namsinh:1}).sort({namsinh:1})
{ "ten" : "Ha Anh Tuan", "namsinh" : 1987 }
{ "ten" : "Phan Manh Quynh", "namsinh" : 1992 }
{ "ten" : "Bich Phuong", "namsinh" : 1993 }
>
```


3.6 MongoDB - Update

Update Operators

Name	Description
\$currentDate	Đặt value của field thành ngày hiện tại.
\$inc	Cộng value của field với một value cho trước.
\$min	Chỉ update field nếu value được chỉ định nhỏ hơn value của field hiện có.
\$max	Chỉ update field nếu value được chỉ định lớn hơn value của field hiện có.
\$mul	Nhân value của field với một value cho trước.
\$rename	Đổi tên một field.
\$set	Đặt value của field đó thành một value khác.
\$setOnInsert	Đặt value của một field nếu kết quả việc update dẫn tới việc insert một document.
\$unset	Xóa field được chỉ định.

```
>db.COLLECTION_NAME.update(SELECTIOIN_CRITERIA, UPDATED_DATA)
```

Cơ sở dữ liệu ban đầu

```
Administrator: Command Prompt - mongo
> db.giaovien.find()
{ "_id" : 1, "msgv" : "gv001", "ten" : "Bich Phuong", "namsinh" : 1995 }
{ "_id" : 2, "msgv" : "gv002", "ten" : "Bao Thy", "namsinh" : 1993 }
{ "_id" : 3, "msgv" : "gv003", "ten" : "Phan Manh Quynh", "namsinh" : 1992 }
{ "_id" : 4, "msgv" : "gv004", "ten" : "Ha Anh Tuan", "namsinh" : 1987 }
>
```

Sau khi chạy lệnh update

```
Administrator: Command Prompt - mongo
> db.giaovien.update({msgv:"gv001"},{$set: {namsinh:1993}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.giaovien.find()
{ "_id" : 1, "msgv" : "gv001", "ten" : "Bich Phuong", "namsinh" : 1993 }
{ "_id" : 2, "msgv" : "gv002", "ten" : "Bao Thy", "namsinh" : 1993 }
{ "_id" : 3, "msgv" : "gv003", "ten" : "Phan Manh Quynh", "namsinh" : 1992 }
{ "_id" : 4, "msgv" : "gv004", "ten" : "Ha Anh Tuan", "namsinh" : 1987 }
>
```

Update nhiều document một lúc

```
>db.collection.updateMany(filter, <operator>update_data)
```

Ví dụ:

```
>db.mycol.updateMany({'title':'MongoDB Overview'},{$set:{'title':'New MongoDB Tutorial'}})
```

```
Administrator: Command Prompt - mongo
> db.test.updateMany({tuoi: "29"},{$set:{ten:"XXX"}})
{ "acknowledged" : true, "matchedCount" : 2, "modifiedCount" : 2 }
>
```

3.7 MongoDB - Remove

Dùng để xóa một document thỏa mãn điều kiện cho trước.

```
>db.COLLECTION_NAME.remove(SELECTIOIN_CRITERIA)
```

Cơ sở dữ liệu ban đầu

```
Administrator: Command Prompt - mongo
> db.giaovien.find()
{ "_id" : 1, "msgv" : "gv001", "ten" : "Bich Phuong", "namsinh" : 1993 }
{ "_id" : 2, "msgv" : "gv002", "ten" : "Bao Thy", "namsinh" : 1993 }
{ "_id" : 3, "msgv" : "gv003", "ten" : "Phan Manh Quynh", "namsinh" : 1992 }
{ "_id" : 4, "msgv" : "gv004", "ten" : "Ha Anh Tuan", "namsinh" : 1987 }
>
```

Thử xóa document với “msgv” có giá trị bằng “gv002”

```
Administrator: Command Prompt - mongo
> db.giaovien.remove({msgv:"gv002"})
WriteResult({ "nRemoved" : 1 })
> db.giaovien.find()
{ "_id" : 1, "msgv" : "gv001", "ten" : "Bich Phuong", "namsinh" : 1993 }
{ "_id" : 3, "msgv" : "gv003", "ten" : "Phan Manh Quynh", "namsinh" : 1992 }
{ "_id" : 4, "msgv" : "gv004", "ten" : "Ha Anh Tuan", "namsinh" : 1987 }
>
```

Để xóa toàn bộ documents trong một collection, dùng lệnh sau:

```
>db.COLLECTION.remove({})
```

3.8 Embedded/Nested Documents

3.8.1 Embedded Documents là gì?

Embedded Documents là một cấu trúc mạnh mẽ, cho phép bạn lưu trữ tất cả các thông tin liên quan trong một tài liệu. Cấu trúc cơ sở dữ liệu này đã được chuẩn hóa và cung cấp hiệu năng tốt hơn và thường cho phép các ứng dụng thực hiện ít truy vấn hơn.

Ví dụ dữ liệu được lưu theo dạng thông thường:

```
Administrator: Command Prompt - mongo
> db.test.find({_id: 1}).pretty()
{
  "_id" : 1,
  "ten" : "Le Van A",
  "tuoi" : "26",
  "sdt" : "0123456789",
  "email" : "vana@gmail.com",
  "diachi" : "75 Ngo Gia Tu, phuong 4, quan 5, TP HCM"
}
```

Ví dụ dữ liệu được lưu theo dạng Embedded Document

```
Administrator: Command Prompt - mongo
{
  "_id" : 2,
  "ten" : "Le Van A",
  "tuoi" : "26",
  "ttlienhe" : {
    "sdt" : "0123456789",
    "email" : "vana@gmail.com",
    "diachi" : "75 Ngo Gia Tu, phuong 4, quan 5, TP HCM"
  }
}
```

Những dữ liệu liên quan đến thông tin liên hệ như “sdt”, “email”, “diachi” đều được nhóm trong “ttlienhe”.

3.8.2 Truy vấn Embedded Documents

Cách truy vấn Embedded Documents cũng tương tự so với các documents thông thường.

Insert

```
>db.Collection.insert({"field1": "value", "field2" : {"field2-1" : "value", "field2-2": "value"}})
```

```
Administrator: Command Prompt - mongo
> db.test.insert({_id:1, ten: "Le Van A", tuoi: "26",
... ttlienhe:{sdt: "0123456789", email:"vana@gmail.com", diachi: "75 Ngo Gi
a Tu, phuong 4, quan 5, TP HCM"}})
```

Ví dụ một câu insert phức tạp hơn

```
Administrator: Command Prompt - mongo
> db.test.insert([ {ten: "Nguyen Thi B", tuoi: "30", ttlienhe:{sdt:"0186785
65", email:"thib@gmail.com", diachi:"TPHCM"}}, {ten: "Tran Van C", tuoi: "
29", ttlienhe:{sdt:"043435345", email:"vanc@gmail.com", diachi:"Vung Tau"}
,chisocothe:{cao:"180", nang:"60"}}])
BulkWriteResult({
  "writeErrors" : [ ],
  "writeConcernErrors" : [ ],
  "nInserted" : 2,
```

Kết quả

```
Administrator: Command Prompt - mongo
> db.test.find().pretty()
{
  "_id" : ObjectId("5dd65720027cb447242246ba"),
  "ten" : "Nguyen Thi B",
  "tuoi" : "30",
  "ttlienhe" : {
    "sdt" : "018678565",
    "email" : "thib@gmail.com",
    "diachi" : "TPHCM"
  }
}
{
  "_id" : ObjectId("5dd65720027cb447242246bb"),
  "ten" : "Tran Van C",
  "tuoi" : "29",
  "ttlienhe" : {
    "sdt" : "043435345",
    "email" : "vanc@gmail.com",
    "diachi" : "Vung Tau"
  },
  "chisocothe" : {
    "cao" : "180",
    "nang" : "60"
  }
}
>
```

Find

```
>db.Conlection.find("field.field1":"value")
```

Ví dụ tìm document có "diachi" trong "ttlienhe" là "Vung Tau"

```
Administrator: Command Prompt - mongo
> db.test.find({"ttlienhe.diachi": "Vung Tau"}).pretty()
{
  "_id" : ObjectId("5dd65720027cb447242246bb"),
  "ten" : "Tran Van C",
  "tuoi" : "29",
  "ttlienhe" : {
    "sdt" : "043435345",
    "email" : "vanc@gmail.com",
    "diachi" : "Vung Tau"
  },
  "chisocothe" : {
    "cao" : "180",
    "nang" : "60"
  }
}
```

Ví dụ tìm document có "diachi" trong "ttlienhe" là "Vung Tau" và "tuoi" là "29"

```
Administrator: Command Prompt - mongo
> db.test.find({"ttlienhe.diachi": "Vung Tau", "tuoi":"29"}).pretty()
{
  "_id" : ObjectId("5dd65720027cb447242246bb"),
  "ten" : "Tran Van C",
  "tuoi" : "29",
  "ttlienhe" : {
    "sdt" : "043435345",
    "email" : "vanc@gmail.com",
    "diachi" : "Vung Tau"
  },
  "chisocothe" : {
    "cao" : "180",
    "nang" : "60"
  }
}
```

Update

Update Operators

Name	Description
\$currentDate	Đặt value của field thành ngày hiện tại.
\$inc	Cộng value của field với một value cho trước.
\$min	Chỉ update field nếu value được chỉ định nhỏ hơn value của field hiện có.
\$max	Chỉ update field nếu value được chỉ định lớn hơn value của field hiện có.
\$mul	Nhân value của field với một value cho trước.
\$rename	Đổi tên một field.
\$set	Đặt value của field đó thành một value khác.
\$setOnInsert	Đặt value của một field nếu kết quả việc update dẫn tới việc insert một document.
\$unset	Xóa field được chỉ định.

```
>db.Collection.update({SELECTIOIN_CRITERIA },{ <operator>: { <field>: <value>, ... }, })
```

Ví dụ update “sdt” trong “ttlienhe” từ document có field “ten” là “Tran Van C”

```
Administrator: Command Prompt - mongo
> db.test.update({ten:"Tran Van C"},{$set:{"ttlienhe.sdt":"0000000000"}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
>
```

4. Truy vấn dữ liệu trên C# .Net.

4.1 Các gói NuGet cần thiết.

Để truy vấn dữ liệu MongoDB trên C# .Net, cần phải cài đặt các gói NuGet sau:

- **MongoDB.Bson**
- **MongoDB.Driver.Core**
- **MongoDB.Driver**

4.2 Tạo kết nối đến server

- Một constructor không tham số theo mặc định sẽ kết nối trên cổng 27017:

```
var client = new MongoClient();
```

- Truyền tham số chuỗi kết nối:

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

- Sử dụng MongoClient:

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

- Sử dụng phương thức Create (tĩnh) từ lớp Client:

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

4.3 Các câu truy vấn cơ bản

Phương thức Filter

Driver .Net cung cấp hàng loạt các cách thức Filter để đáp ứng hầu hết mọi nhu cầu của người dùng khi xuất dữ liệu vào menu View/ Export

Đối tượng Builders<BsonDocument>.Filter trong thư viện này cung cấp các hàm phổ biến như sau:

```
var builder = Builders<BsonDocument>.Filter;
```

Tên hàm	Ý nghĩa
builder.Eq(attribute,value)	Lọc ra các phần tử mà attribute của nó có giá trị là value
builder.Gt(attribute,value)	Lọc ra các phần tử mà attribute của nó có giá trị lớn hơn value
builder.Gte(attribute,value)	Lọc ra các phần tử mà attribute của nó có giá trị lớn hơn hoặc bằng value
builder.Lt(attribute,value)	Lọc ra các phần tử mà attribute của nó có giá trị nhỏ hơn value
builder.Lte(attribute,value)	Lọc ra các phần tử mà attribute của nó có giá trị nhỏ hơn hoặc bằng value
builder.Where (LINQ Expression)	Lọc kết hợp nhiều điều kiện

Lấy Database:


```
const string MongoDBConnectionString = "mongodb://localhost";
// Tạo một client connection tới MongoDB
var client = new MongoClient(MongoDBConnectionString);
var database = client.GetDatabase("MongoDBStore");
```

Tạo collection tạm để lưu dữ liệu khi truy xuất từ một collection khác trong database

```
// Dùng hàm GetCollection để truy xuất dữ liệu trong database,
// ví dụ ở đây là collection "products"
```

```
var products = database.GetCollection<Product>("products");
```

Xóa collection:

```
// Xóa collection
await database.DropCollectionAsync("products");
```

Tạo collection trong database:

```
// Tạo collection
await database.CreateCollectionAsync("products");
```

Tạo một session:

```
// Tạo một session để sử dụng lại trong transactions
using (var session = await client.StartSessionAsync())
```

Tạo và nhập một số dữ liệu mẫu, sử dụng session vừa tạo ở trên:

```
// Tạo một số dữ liệu mẫu
var tv = new Product { Description = "Television",
                        SKU = 4001,
                        Price = 2000 };
var book = new Product { Description = "A funny book",
                        SKU = 43221,
                        Price = 19.99 };
var dogBowl = new Product { Description = "Bowl for Fido",
                        SKU = 123,
                        Price = 40.00 };

// Nhập các dữ liệu vừa tạo ở trên
await products.InsertOneAsync(session, tv);
await products.InsertOneAsync(session, book);
await products.InsertOneAsync(session, dogBowl);
```

Xóa document:

```
//Xóa một document
var Deleteone = await COLLECTION_NAME.DeleteOneAsync(
    Builders<BsonDocument>.Filter.Eq(key, value));

//Xóa nhiều documents
var DelMultiple = await COLLECTION_NAME.DeleteManyAsync(
    Builders<BsonDocument>.Filter.Lt(key, value));
```

Update document:

```
// update một document
var filter = Builders<MyType>.Filter.Eq(s => s.Id, id);
var update = Builders<MyType>.Update.Set(s => s.Description, description);
var result = await collection.UpdateOneAsync(filter, update);
```

Update nhiều document:

```
//update nhiều document
var builder = Builders<SampleClass>.Update;
await myCollection.UpdateManyAsync(x => x.a > 10, builder.Inc(x => x.a, 1));
```

In dữ liệu ra màn hình:

```
//In dữ liệu ra màn hình
await COLLECTION_NAME.Find(new BsonDocument()).ForEachAsync(x => Console.WriteLine(x));
```

Ví dụ minh hoạt tổng quan:

```
1. using MongoDB.Bson;
2. using MongoDB.Bson.Serialization.Attributes;
3. using MongoDB.Driver;
4. using System;
5. using System.Threading.Tasks;
6.
7. namespace MongoDBTransaction
8. {
9.     public static class Program
10.    {
11.        public class Product
12.        {
13.            [BsonId]
14.            public ObjectId Id { get; set; }
15.            [BsonElement("SKU")]
16.            public int SKU { get; set; }
17.            [BsonElement("Description")]
18.            public string Description { get; set; }
19.            [BsonElement("Price")]
20.            public Double Price { get; set; }
21.        }
22.
23.
24.        const string MongoDBConnectionString = "mongodb://localhost";
25.
26.        public static async Task Main(string[] args)
27.        {
28.            if (!await UpdateProductsAsync()) { Environment.Exit(1); }
29.            Console.WriteLine("Finished updating the product collection");
30.            Console.ReadKey();
31.        }
32.
33.        private static async Task<bool> UpdateProductsAsync()
34.        {
35.            // Tạo một client connection đến MongoDB database
36.            var client = new MongoClient(MongoDBConnectionString);
37.            var database = client.GetDatabase("MongoDBStore");
38.
39.            // Dùng hàm GetCollection để truy xuất dữ liệu trong database,
40.            // ví dụ ở đây là collection "products"
41.
42.            var products = database.GetCollection<Product>("products");
43.
44.            // Xóa collection
45.            await database.DropCollectionAsync("products");
46.
47.            // Tạo collection
48.            await database.CreateCollectionAsync("products");
49.
50.
51.
52.
```

```

53.         // Tạo một session để sử dụng lại trong transactions
54.         using (var session = await client.StartSessionAsync())
55.         {
56.             // Mở transaction
57.             session.StartTransaction();
58.
59.             try
60.             {
61.                 // Tạo một số dữ liệu mẫu
62.                 var tv = new Product { Description = "Television",
63.                                         SKU = 4001,
64.                                         Price = 2000 };
65.                 var book = new Product { Description = "A funny book",
66.                                          SKU = 43221,
67.                                          Price = 19.99 };
68.                 var dogBowl = new Product { Description = "Bowl for Fido",
69.                                              SKU = 123,
70.                                              Price = 40.00 };
71.
72.                 // Nhập các dữ liệu vừa tạo ở trên
73.                 await products.InsertOneAsync(session, tv);
74.                 await products.InsertOneAsync(session, book);
75.                 await products.InsertOneAsync(session, dogBowl);
76.
77.                 var resultsBeforeUpdates = await products
78.                                         .Find<Product>(session, Builders<Product>.Filter.Empty)
79.                                         .ToListAsync();
80.                 Console.WriteLine("Original Prices:\n");
81.                 foreach (Product d in resultsBeforeUpdates)
82.                 {
83.                     Console.WriteLine(
84.                         String.Format("Product Name: {0}\tPrice: {1:0.00}
85.                                         d.Description, d.Price)
86.                     );
87.                 }
88.
89.                 // Giảm giá bán tất cả các sản phẩm xuống 10%
90.                 var update = new UpdateDefinitionBuilder<Product>()
91.                     .Mul<Double>(r => r.Price, 1.1);
92.                 await products.UpdateManyAsync(session, Builders<Product>
93.                     .Filter.Empty, update);
94.                 // Commit transaction
95.                 await session.CommitTransactionAsync();
96.             }
97.             catch (Exception e)
98.             {
99.                 Console.WriteLine("Error writing to MongoDB: " + e.Message);
100.                 await session.AbortTransactionAsync();
101.                 return false;
102.             }
103.
104.             // In kết quả ra màn hình
105.             Console.WriteLine("\n\nNew Prices (10% increase):\n");
106.             var resultsAfterCommit = await products

```

```
107.             .Find<Product>(session, Builders<Product>.Filter.Empty)
108.             .ToListAsync();
109.         foreach (Product d in resultsAfterCommit)
110.         {
111.             Console.WriteLine(
112.                 String.Format("Product Name: {0}\tPrice: {1:0.00}",
113.                               d.Description, d.Price)
114.             );
115.         }
116.
117.         return true;
118.     }
119. }
120. }
121. }
```

TRƯỜNG ĐẠI HỌC SƯ PHẠM THÀNH PHỐ HỒ CHÍ MINH
Khoa Công nghệ thông tin

BÁO CÁO ĐỀ TÀI CUỐI KÌ TÌM HIỂU MONGODB

Môn: Cơ sở dữ liệu nâng cao

GV hướng dẫn: ThS. Lương Trần Hy Hiến

Nhóm 5

Nguyễn Văn An	43.01.104.002
Võ Thế Duy	43.01.104.032
Nguyễn Minh Pháp	43.01.104.124
Châu Bảo Thịnh	43.01.104.169
Trần Minh Trường	43.01.104.192
Nguyễn Doãn Tú	43.01.104.196