BÁO CÁO ĐỒ ÁN CUỐI KÌ

ĐỀ TÀI: TÌM HIỂU MONGODB

[MÔN CƠ SỞ DỮ LIỆU NÂNG CAO]

NHÓM 27:

43.01.104.137 – LÊ THỊ TRÚC PHƯƠNG

43.01.104.091 – TRẦN THỊ KIỀU LINH

43.01.104.164 – LÊ THỊ HIẾU THẢO

MỤC LỤC

LỜI NÓI ĐẦU

Cơ sở dữ liệu là một phần không thể thiếu trong các phần mềm, hệ thống từ di động đến các trang web ứng dụng cao hiện nay. Bởi đây là công cụ dùng để lưu trữ nguồn dữ liệu lớn của người dùng hay các thông tin về sản phẩm của phần mềm giúp hệ thống có thể tương tác gián tiếp đến người sử dụng và mang lại hiệu quả cao trong mục đích kinh doanh. Các hệ quản trị cơ sở dữ liệu quan hệ (RDBMS) là các cơ sở dữ liệu được học và dùng phổ biến hiện nay nhưng song song đó vẫn tồn tại những mặt hạn chế khi sử dụng. Bên cạnh đó sự ra bùng nổ của Web 2.0 và sự phát triển của Internet và số lượng người dùng tăng lên rất lớn, người dùng có thể thỏa mái tạo nội dung trên Web, làm cho dữ liệu trở nên khổng lồ vượt quá giới hạn xử lý của các hệ quản trị cơ sở dữ liệu quan hệ truyền thống. Vì thế, cần có một cơ sở dữ liệu có thể lưu trữ dữ liệu với số lượng lớn và có thể truy xuất một cách nhanh chóng và hiệu quả. Từ đó, MongoDB ra đời với mục đích nhằm khắc phục những khuyết điểm của RDBMS và để đáp ứng nhu cầu lưu trữ hiện nay.

Nhóm đã tìm hiểu sơ nét về MongoDB, cách truy vấn dữ liệu khác với SQL như thế nào và cách sử dụng cơ sở dữ liệu này trên PHP. Vì còn hạn chế về mặt kiến thức, kinh nghiệm nên trong quá trình sử dụng chúng tôi còn những khuyết điểm, sai sót, kính mong thầy góp ý, nhận xét để cải thiện hệ thống hoàn chỉnh và tốt hơn.

I. SƠ NÉT NOSQL

1. Khái niệm

NoSQL là Non-Relational: tạm dịch là không có quan hệ. NoSQL là cơ sở dữ liệu không quan hệ, ràng buộc giữa các Collection (hay còn gọi là bảng trong cơ sở dữ liệu bình thường). Tức là giữa các Collection (Bảng) sẽ không có khóa chính, khóa ngoại như trong cơ sở dữ liệu bình thường.

Cơ sở dữ liệu NoSQL là cơ sở dữ liệu được xây dựng dành riêng cho mô hình dữ liệu có sơ đồ linh hoạt để xây dựng các ứng dụng hiện đại. Cơ sở dữ liệu NoSQL được công nhận rộng rãi vì khả năng dễ phát triển, chức năng cũng như hiệu năng ở quy mô lớn. Ngoài ra, NoSQL sinh ra để giải quyết các khuyết điểm của RDBMS.

2. Phân loại

Gồm 4 loại với các dạng lưu trữ khác nhau:

- ***Key-Value Database***: Dữ liệu được lưu trữ trong database dưới dạng key-value
- ***Document Database:*** Mỗi object sẽ được lưu trữ trong database dưới dạng một document. Dữ liệu sẽ được lưu trữ dưới dạng BSON/JSON/XML dưới database. Dữ liệu không schema cứng như SQL, do đó ta có thể thêm/sửa field, thay đổi table, … rất nhanh và đơn giản.
- ***Column-Family Database***: Dữ liệu được lưu trong database dưới dạng các cột, thay vì các hàng như SQL. Mỗi hàng sẽ có một key/id riêng. Điểm đặt biệt là các hàng trong một bảng sẽ có số lượng cột khác nhau.
- ***Graph Database:*** Dữ liệu trong graph database được lưu dưới dạng các node. Mỗi node sẽ có 1 label, 1 số properties như một row trong SQL. Các node này được kết nối với nhau bằng các relationship.

Trong đó MongoDB là loại Document Database.

3. Ưu nhược điểm của NoSQL

❖ Ưu điểm của NoSQL:

- Dữ liệu trong NoSQL DB được lưu dưới dạng document, object. Truy vấn dễ dàng và nhanh hơn RDBMS.
- NoSQL có thể làm việc hoàn toàn tốt với dữ liệu dạng không có cấu trúc.
- Việc đổi cấu trúc dữ liệu (Thêm, xóa trường hoặc bảng) rất dễ dàng và nhanh gọn trong NoSQL.
- NoSQL DB có thể mở rộng, chạy trên nhiều máy một cách dễ dàng.

❖ Nhược điểm:

- **Hỗ trợ không đồng đều cho các doanh nghiệp:** trong khi các nhà cung cấp chủ chốt của các RDBMS như SQL Server, Oracle,…thường đưa ra sự hỗ trợ tốt cho khách hàng thì các nhà cung cấp nguồn mở mới thành lập không thể được mong đợi sẽ cung cấp hỗ tốt hơn.
- **Chưa đủ "chín" cho các doanh nghiệp**: dù chúng đã được triển khai tại một số công ty lớn thì các CSDL NoSQL vẫn đối mặt với một vấn đề về sự tin cậy chính với nhiều doanh nghiệp. Vấn đề lớn của NoSQL là thiếu về độ chín muồi và các vấn đề về tính không ổn định, trong khi đó tính chín muồi, hỗ trợ đầy đủ chức năng và tính ổn định của các RDBMS được thiết lập đã từ lâu.
- **Những hạn chế về tri thức nghiệp vụ:** các CSDL NoSQL không có nhiều sự đeo bám tới các công cụ BI thường được sử dụng, trong khi những yêu cầu và phân tích hiện đại đơn giản nhất thì cũng liên quan khá nhiều tới sự tinh thông về lập trình.

4. So sánh NoSQL và CSDL quan hệ

	CSDL quan hệ	NoSQL
Định nghĩa	Là các hệ cơ sở dữ liệu quan hệ (RDBMS)	Là cơ sở dữ liệu phân tán.
Yêu cầu phần cứng cho dữ liệu lớn	Yêu cầu phần cứng máy chủ cao.	Không nhất thiết máy chủ cao, có thể mở rộng máy chủ ở nhiều nơi với tài nguyên phần cứng thấp.
Mô hình	CSDL dựa trên bảng (table)	CSDL dựa trên tài liệu (document), cặp khóa-giá trị (key-value), cơ sở dữ liệu biểu đồ (graph), cột (column-family)
Hiệu xuất	Kém hơn. Vì có SQL Relational giữa các bảng.	Cực tốt. Bỏ qua SQL Bỏ qua các ràng buộc dữ liệu.
Tính năng tốt nhất	Hỗ trợ đa nền tảng, bảo mật và miễn phí	Dễ sử dụng, hiệu suất cao và công cụ linh hoạt.
Hiệu xuất đọc ghi	Kém do thiết kế để đảm bảo sự ra vào của dữ liệu.	Tốt với mô hình xử lý lô và những tối ưu về đọc – ghi dữ liệu.

II. GIỚI THIỆU MONGODB

1. Khái niệm

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 và đa dạng và không cố định (hay còn gọi là Big Data). MongoDB là cơ sở dữ liệu hàng đầu của NoSQL nên được thừa hưởng những ưu điểm và cấu trúc của NoSQL.

2. Đặc trưng

+ MongoDB sử dụng cơ chế NoSQL để truy vấn.
+ MongoDB lưu dữ liệu theo định dạng JSON (chúng ta gọi nó là BSON (Binary JSON))
+ JSON là viết tắt của JavaScript Object Notations và trông nó kiểu kiểu như {"name": "nhat"}
+ JSON document lưu trữ các dữ liệu theo kiểu định dạng key-value, trong ví dụ trên, "name" là key, "nhat" là value. Có 2 kiểu cấu trúc cơ bản trong JSON là:
    - Array: Một danh sách chứa các thể hiện của một list các key-value
    - Dictionaries: Lưu trữ các cặp key-value, ai đã từng dùng ngôn ngữ lập trình python sẽ rất quen thuộc với kiểu dictionaries này. Bạn có thiểu nó tương tự kiểu con trỏ trong C, C++, nghĩa là key trỏ đến gía trị mà nó lưu trữ, bạn có thể ,lấy giá trị nó lưu trữ thông qua key.
+ MongoDb có cấu trúc Schema động, nó không cần phải định nghĩa một mô hình với những mối quan hệ phải được định sẵn khi thiết kế, điều này MongoDB kế thừa từ NoSQL.
+ MongoDB không hỗ trợ Transactions.

3. 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 của MongoDB có thể nhanh tới gấp 100 lần so với MySQL

Các lí do dẫn đến hiệu năng cao của MongoDB:

- 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, join nó không cần phải mất thời gian kiểm tra xem có thỏa mãn các bảng liên quan như trong RDBMS hay không cản trở tốc độ truy vấn của MongoDB.
- 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… vì thế có thể insert thoải mái dữ liệu.
+ **Là Rich Query Language:**
  - Có sẵn các method để thực hiện create, read, update, delete dữ liệu.
+ **Tính sẵn có:**
  - MongoDB hỗ trợ **replica set** nhằm đảm bảo việc sao lưu và khôi phục dữ liệu.

4. Nhược điểm

+ MongoDB không có các tính chất ràng buộc như trong RDBMS –> dễ bị làm sai dữ liệu
+ MongoDB không thể sử dụng những phép join quá phức tạp.
+ 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 dữ 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.

5. Các trường hợp sử dụng MongoDB

+ Website **có tính chất INSERT cao**, bởi vì mặc định MongoDB có sẵn cơ chế ghi với tốc độ cao và an toàn. Ví dụ như **website quản lý nội dung và blog.**
+ Website của chúng ta của ở dạng **real-time nhiều**, nghĩa là nhiều người cùng thao tác với ứng dụng trong thời gian thực. Ví dụ như **ứng dụng chat hay stream trực tuyến**.
+ Các hệ thống **có nhiều dữ liệu – big data** với yêu cầu truy vấn, tìm kiếm nhanh. Ví dụ như: **các trang web di động, các mạng xã hội hay các catalog thương mại điện tử**.

6. Các thương hiệu nổi tiếng sử dụng mongoDB

7. Mô hình dữ liệu của Embedded Documents

a. Khái niệm của Embedded Documents

Embedded documents (Tài liệu nhúng) là documents có lược đồ riêng và là 1 phần của documents khác. Hiểu đơn giản thì embedded documents là 1 field nằm trong 1 collection thay vì lưu dữ liệu kiểu References ta phải thiết kế 2 collection để thể hiện mối quan hệ One-to-Many.

b. Cách tổ chức

+ Mô tả mô hình dữ liệu dựa trên embedded documents để thể hiện mối quan hệ 1-nhiều.

Ví dụ: Mô hình diễn tả mối quan hệ của người bảo trợ với nhiều địa chỉ khác nhau. Một người bảo trợ sẽ có nhiều địa chỉ khác nhau (mqh 1-nhiều). Đối với khai báo bình thường thì sẽ nhập dữ liệu vào từng collection như hình 1. Nhưng nếu ứng dụng của bạn thường xuyên truy xuất dữ liệu địa chỉ với thông tin tên, thì ứng dụng của bạn cần đưa ra nhiều truy vấn để giải quyết các tham chiếu.

Một lược đồ tối ưu hơn sẽ là nhúng các thực thể dữ liệu địa chỉ vào dữ liệu bảo trợ (dùng embedded documents), như hình 2.

```
{
_id: "joe",
name: "Joe Bookreader"
}
{
patron_id: "joe",
street: "123 Fake Street",
city: "Faketon",
state: "MA",
zip: "12345"
}
{ patron_id: "joe",
street: "1 Some Other Street",
city: "Boston",
state: "MA",
zip: "12345"
}
```

Hình 1: Mô tả dữ liệu của từng collection người bảo hộ và 2 collection địa chỉ

```
{
  _id: "joe",
  name: "Joe Bookreader",
  addresses: [
            {
              street: "123 Fake Street",
              city: "Faketon",
              state: "MA",
              zip: "12345"
            },
            {
              street: "1 Some Other Street",
              city: "Boston",
              state: "MA",
              zip: "12345"
            }
          ]
}
```

Hình 2: Mô tả dữ liệu theo dạng Embedded Documents

c. Cách truy vấn

+ Khi thiết kế dữ liệu kiểu Embedded thì chúng ta không cần tạo ra nhiều collection để lưu trữ, mà sẽ lưu trữ tất cả vào một collection.

Ví dụ: Tạo collection inventory. Thay vì việc phải tạo ra 2 collection rồi lưu kiểu reference thì chỉ cần lưu ở 1 bảng duy nhất.

```
{ item: "journal", instock: [ { warehouse: "A", qty: 5 }, { warehouse: "C", qty: 15 } ] },
    { item: "notebook", instock: [ { warehouse: "C", qty: 5 } ] },
    { item: "paper", instock: [ { warehouse: "A", qty: 60 }, { warehouse: "B", qty: 15 } ] },
    { item: "planner", instock: [ { warehouse: "A", qty: 40 }, { warehouse: "B", qty: 5 } ] },
    { item: "postcard", instock: [ { warehouse: "B", qty: 15 }, { warehouse: "C", qty: 35 } ] }
```

+ Truy vấn dữ liệu được lồng vào mảng

```
db.inventory.find( { "instock": { warehouse: "A", qty: 5 } } )
```

Hình: Trong Collection inventory tìm "*warehouse: "A", qty: 5*" trong mảng "*instock*"

```
{ item: "journal", instock: [ { warehouse: "A", qty: 5 }, { warehouse: "C", qty: 15 } ] }
```

Hình: kết quả đoạn truy vấn trên

Chú ý: Khi query mà một element lưu dạng embedded/nested thì yêu cầu phải đúng chính xác với document được chỉ đinh, bao gồm cả thứ tự các trường trong document được chỉ định.

```
db.inventory.find( { "instock": { qty: 5, warehouse: "A" } } )
# không có kết quả nào được tìm thấy
```

+ Truy vấn một điều kiện trên một trường trong một mảng tài liệu
  - Muốn tìm kiếm các document với điều kiện cho một field embeded trong một array của document.Ta sẽ dùng dấu (.) để nối tên của array field với tên của field trong nested document. ví dụ:

```
db.inventory.find( { 'instock.qty': { $lte: 20 } } )
```

- Trong ví dụ trên sẽ trả lại tất cả các document mà instock array có ít nhất một embedded document có trường qty có giá trị nhỏ hơn hoặc bằng 20.
- Có thể sử dụng index của array để chỉ rõ muốn search ở document thứ bao nhiêu trong nested document:

```
db.inventory.find( { 'instock.0.qty': { $lte: 20 } } )
```

+ Truy vấn nhiều điều kiện trên một trường trong một mảng tài liệu
  - Trong phần 1 ở trên có phần chú ý khi chỉ dùng find thì yêu cầu về thứ tự các field trong collection được chỉ định. Nếu không muốn phải chú ý tới vấn đề order của field thì dùng elemMatch

```
db.inventory.find( { "instock": { $elemMatch: { qty: 5, warehouse: "A"}}})

#result:

{ item: "journal", instock: [ { warehouse: "A", qty: 5 }, { warehouse: "C", qty: 15}]}
```

- Có thể kết hợp với các Comparison Query Operators trong elemMatch

```
db.inventory.find( { "instock": { $elemMatch: { qty: { $gt: 10, $lte: 20}}}})

#result

{item: "journal", instock: [ { warehouse: "A", qty: 5 }, { warehouse: "C", qty: 15 }]}

{item: "paper", instock: [ { warehouse: "A", qty: 60 }, { warehouse: "B", qty: 15 }]}

{item: "postcard", instock: [ { warehouse: "B", qty: 15 }, { warehouse: "C", qty: 35 }
```

III. CÁCH THỨC TRUY VẤN VÀ XỬ LÝ CSDL TRÊN MONGODB

1. Tạo database trong MONGODB

+ Để tạo một database sử dụng cú pháp sau:

```
use DatabaseName
```

Trong đó: *DatabaseName* là tên của database muốn tạo.

Chú ý: Nếu như database muốn tạo đã tồn tại rồi thì nó sẽ không tạo nữa mà sử dụng luôn database đó.

VD: Tạo database có tên **KhoGao**.

```
use KhoGao
```

2. Xem database

+ Để xem database đang sử dụng (current database) thì sử dụng lệnh:

```
db
```

+ Để xem tất cả các database đã được tạo trên MongoDB thì sử dụng lệnh:

```
show dbs
```

3. Xóa database

+ Để xóa database sử dụng cú pháp sau:

```
db.dropDatabase()
```

Chú ý: Lệnh này dùng để xóa current database, nên khi sử dụng nó thì bắt buộc bạn phải switch vào database cần xóa thì mới thành công.

VD: Xóa database **KhoGao**

```
MongoDB Enterprise > show dbs
admin       0.000GB
local       0.000GB
KhoGao      0.000GB
MongoDB Enterprise > use KhoGao
switched to db KhoGao
MongoDB Enterprise > db.dropDatabase()
{ "dropped" : " KhoGao ", "ok" : 1 }
MongoDB Enterprise >
```

+ Kết quả trả về dạng:

```
{ "dropped" : " KhoGao ", "ok" : 1 }
```

Là đã xóa thành công database trên MongoDB rồi, nhưng nếu cần kiểm tra một cách chắc chắn thì có thể dùng lệnh **show dbs** để kiểm tra lại.

4. Thao tác với collection trong MongoDB

+ Nếu như trong các hệ quản trị cơ sở dữ liệu trước đây có các table để lưu trữ dữ liệu, thì trong MongoDB chúng được thay thế với một khái niệm hoàn toàn mới là các Collection.

a. Tao collection

+ Để tạo collection sử dụng cú pháp:

```
db.createCollection(collectionName, option)
```

*Trong đó:

- **collectionName** là tên của collection các bạn muốn tạo.
- **option** (*Nếu như ứng dụng của bạn không cần quá tối ưu, thì bạn hoàn toàn có thể không cần cấu hình option cho collection*) là một đối tượng chứa các tùy chọn riêng cho collection. Các tùy chọn bao gồm:
  - ***capped*** - đây là thông số cấu hình hành động sẽ xảy ra khi collection vượt quá dung lượng cho phép (thông số size). Nếu capped: true thì khi dung lượng quá hạn mức cho phép nó sẽ ghi đè các dữ liệu cũ nhất.

- ***autoIndexId*** - đây là thông số cấu hình xem có đánh chỉ mục cho trường _id không. Nếu autoIndexId: true thì sẽ đánh chỉ mục cho trường _id (Phiên bản 3.4 tùy chọn này sẽ bị xóa).
- ***size*** - Xác định kích cỡ tối ta collection có thể chứa (đơn vị byte).
- ***max*** - Xác định số tài liệu tối ta mà một capped collection có thể chứa.
- ***storageEngine*** - Cấu hình storageEngine cho collection
- validator - cấu hình định dạng cho dữ liệu cảu các trường (xem thêm)
- ***validationLevel*** - xác định độ nghiêm ngặt của validator ở trên. Giá trị có thể là:
  - "off" - không validator khi insert hoặc update.
  - "strict" - Đây là giá trị mặc định. Thiết lập validator với mọi câu lệnh insert và update.
  - "moderate" - thiết lập validator cho các rule được liệt kê ở validator, nhếu trường nào không có thì nó sẽ không áp dụng.
- ***validationAction*** - thiết lập trạng thái khi dữ liệu không khớp với validator. Giá trị có thể điền vào là "error" hoặc "warn".
- ***indexOptionDefaults***
- ***viewOn***
- ***pipeline***
- ***collation***

Lưu ý: Để có thể thực hiện được lệnh này thì cần khải khai báo sử dụng database.

VD: Tạo một collection có tên là admin, và đồng thời validator dữ liệu cho collection.

```
MongoDB Enterprise > use KhoGao
switched to db KhoGao
MongoDB Enterprise > db.createCollection("admin",{validator:{$or:
[{name: {$type:
 "string" }},{password: {$type: "string"}},{email: { $regex:
"/@gmail\.com"}}
]}}
)
{ "ok" : 1 }
MongoDB Enterprise >
```

*Trong đoạn trên đã ràng buộc dữ liệu cho trường name và password có kiểu dữ liệu là **string**, email bắt buộc phải có đuôi là **@gmail.com.**

b. Xóa collection

+ Để xóa một collection sử dụng cú pháp:

```
db.collectionName.drop()
```

VD: Xóa collection admin.

```
MongoDB Enterprise > db.admin.drop()
true
MongoDB Enterprise >
```

c. Xem danh sách các collection có trong database.

+ Để xem danh sách các collection đang có trong databasesử dụng cú pháp:

```
show collections
```

VD: xem danh sách các collection trong database **KhoGao.**

```
switched to db KhoGao
MongoDB Enterprise > show collections
admin
MongoDB Enterprise >
```

5. Thêm dữ liệu

+ MongoDB đã cung cấp 3 phương thức để thực hiện việc thêm mới dữ liệu vào trong collection. Bao gồm các phương thức sau: ***insert, insertOne, inserMany.***

a. Phương thức insert

+ Dùng để thêm mới một hoặc nhiều dữ liệu vào trong MongoDB với cú pháp:

```
db.conlectionName.insert(data)
```

***Trong đó :**

- **CollectionName** là tên của collection chúng ta cần thêm dữ liệu vào.
- **Data** có thể là 1 object chứa các trường và giá trị của nó hoặc cũng có thể là một mảng đối tượng (nếu như bạn muốn thêm nhiều bản ghi trên một lần khai báo).

VD: Thêm mới nhiều dữ liệu vào collection có tên là admin.

```
db.admin.insert([
 {
  name: "Kieu Linh",
  password: "admin",
  email: "kieulinh@gmail.com"
 },
 {
  name: "administrator",
  password: "admin123",
  email: "admin@gmail.com"
 }
])
```

*Với ví dụ này nếu như tham số **nInserted** trả về có giá trị là 2 thì tức nào dữ liệu bạn đã thêm thành công. Hoặc nếu bạn chỉ thêm một dữ liệu thì tham số trả về sẽ là 1 nếu thêm thành công.

b. Phương thức insertOne

+ Cho phép insert **một dữ liệu** vào trong MongoDB trên một lần khai báo với cú pháp:

```
db.collectionName.insertOne(data)
```

*__Trong đó__:

- **CollectionName** là tên của collection chúng ta cần thêm dữ liệu vào.
- **Data** là một obejct chứa dữ liệu chúng ta cần thêm vào.

VD: Thêm mới một dữ liệu vào trong MongoDB.

```
db.admin.insertOne({
  name: "Kieu Linh",
  password: "admin",
  email: "kieulinh@gmail.com"
})
```

+ Nếu như thêm thành công thì hệ thống sẽ trả về cho chúng ta _id của dữ liệu vừa được thêm.

c. Phương thức insertMany

+ Cho phép chúng ta thêm mới **nhiều dữ liệu** vào trong MongoDB với cú pháp:

```
db.collectionName.insertMany(data)
```

*Trong đó:

- **CollectionName** là tên của collection chúng ta cần thêm dữ liệu vào.
- **Data** là một mảng obejct chứa dữ liệu chúng ta cần thêm vào.

VD: Thêm nhiều dữ liệu vào trong MongoDB.

```
db.admin.insertMany([
 {
  name: "Kieu Linh",
  password: "admin",
  email: "kieulinh@gmail.com"
 },
 {
  name: "administrator",
  password: "admin123",
  email: "admin@gmail.com"
 }
])
```

+ Nếu như thành công thì nó sẽ trả về **_id** của các dữ liệu vừa được thêm.

Lưu ý: Với cả ba phương thức trên nếu như **collectionName** chưa tồn tại trong hệ thống thì mặc định MongoDB sẽ tự động thêm mới và đồng thời insert dữ liệu.

6. Truy vấn dữ liệu

a. Lấy tất cả dữ liệu trong Collection.

+ Để lấy tất cả dữ liệu ở trong collection sử dụng cú pháp:

```
db.collectionName.find()
```

***Trong đó**:

- **collectionName** là tên của collection mà các bạn muốn truy vấn.

+ Tuy nhiên, khi chỉ sử dụng mỗi phương thức **find()** thì dữ liệu trả về sẽ dưới dạng object nhưng không theo một cấu trúc nào cả.
+ Và nếu như bạn muốn dữ liệu được trả về được hiển thị theo cấu trúc đã được định sẵn thì chỉ cần thêm hàm **pretty()** vào phía sau hàm **find()**.

```
db.collectionName.find().pretty()
```

b. Truy vấn có điều kiện

+ Để truy vấn có điều kiện trong MongoDB thì bạn cũng sử dụng cú pháp tương tự như phần lấy tất cả dữ liệu ở trong collection, nhưng lúc này chúng ta sẽ chèn thêm điều kiện vào trong hàm **find()** với cú pháp sau:

db.collection.find(condition)

*Trong đó:

- **CollectionName** là tên của collection mà các bạn muốn truy vấn.
- **Condition** là object chứa mệnh đề điều kiện.

Bảng phép toán và câu lệnh truy vấn :

Phép Toán	Cú Pháp	Ví dụ	Câu lệnh tương ứng trong SQL
Bằng (Equality)	{key: value}	db.admin.find({name: "Kieu Linh" }).pretty()	.. WHERE name = "Vu Thanh Tai"
Nhỏ hơn (Less Than)	{key: {$lt: value}}	db.admin.find({age: { $lt: 18}).pretty()	... WHERE age < 18
Nhỏ hơn bằng (Less Than Equals)	{key: {$lte: value}}	db.admin.find({age: { $lte: 18}).pretty()	... WHERE age <= 18
Lơn hơn (Greater Than)	{key: {$gt: value}}	db.admin.find({age: { $gt: 12}).pretty()	... WHERE age > 12
Lớn hơn bằng (Greater Than Equals)	{key: {$gte: value}}	db.admin.find({age: { $gte: 12}).pretty()	... WHERE age >= 12
Khác (Not Equals)	{key: {$ne: value}}	db.admin.find({age: { $ne: 12}).pretty()	... WHERE age != 12
Trong (In)	{key: {$in: [value1, value2,..]}}	db.admin.find({age: { $in: [12, 18]}).pretty()	... WHERE age IN (12, 18)

Không Thuộc (Not In)	{key: {$nin: [value1, value2,..]}}	db.admin.find({age: { $nin: [12, 18]}).pretty()	... WHERE age NOT IN (12, 18)

VD: in ra tất cả các admin có tên là **Kieu Linh** có trong colletion admin.

```
db.collection.find({name: "Kieu Linh"}).pretty()
```

c. Truy vấn nhiều điều kiện

+ Trong MongoDB cũng có hỗ trợ chúng ta truy vấn nhiều điều kiện trên một lần khai báo, với các toán tử AND,OR
+ **Toán tử AND**
  - Để thực hiện phép toán này thì các bạn chỉ cần thêm các điều kiện của câu truy vấn vào trong object chứa điều kiện bind vào trong phương thức find. Nếu như bạn muốn and bao nhiêu điều kiện thì thêm bấy nhiêu vào trong object.

    VD: Lấy ra admin có tên Kieu Linh và có tuổi là 18 trong collection admin.

```
db.admin.find({
   name: "Kieu Linh",
   age: 18
})
```

+ **Toán tử OR**
  - Để sử dụng mệnh đề **OR** trong MongoDB cần phải truyền một key scope có tên là **OR** vào làm key chứa mảng các điều kiện hoặc, theo cú pháp:

```
db.collectionName.find({
   $or : [
      {key1: value1},
      {key2: value2},
      ...,
      {keyn: valuen}
   ]
}).pretty()
```

VD: Lấy ra tất cả các admin có **tuổi bằng 12 hoặc name là Kieu Linh** trong collection admin.

```
db.admin.find({
    $or : [
        {age: 12},
        {name: "Kieu Linh"},
    ]
}).pretty()
```

+ Kết hợp AND và OR
  - Để kết hợp giữa AND và OR thì bạn chỉ cần làm tương tự như cách thực hiện truy vấn AND, và nếu truy vấn nào là OR thì object đó lại làm tương tự như truy vấn OR.
    VD: Lấy ra tất cả các admin có **tuổi bằng 12 hoặc password bằng admin và có name là Kieu Linh** trong collection admin.

```
db.admin.find({
    $or : [
        {age: 12},
        {password: "admin"},
    ],
    name: "Kieu Linh"
}).pretty()
```

+ Chọn lọc các trường cần lấy ra trong MongoDB
  - Để chọn lọc các trường cần hiển thị ra trong 1 collection thì các bạn sử dụng phương thức find() với cú pháp sau:

```
db.collectionName.find(objectwhere,objectselect)
```

*Trong đó:
- **objectwhere** là object chứa các điều kiện ở các phần trên. Nếu bạn không muốn lọc theo điều kiện thì bạn để một object rỗng vào.
- **objectselect** là obejct chứa các trường dữ liệu cần lấy ra. Mặc định thì nó sẽ lấy cả _id, nên nếu như bạn không muốn hiển thị _id thì bạn cần thêm **_id: 0** vào object.

VD: Lấy ra trường _id, name, password của tất cả các document có trong collection admin.

```
db.admin.find({},{name: 1,password: 1}).pretty()
```

VD: Lấy ra name,password của những bản ghi có name = Kieu Linh có trong admin collection.

```
db.admin.find({name: "Kieu Linh"},{name: 1, password: 1, _id: 0}).pretty()
```

+ Giới hạn số lượng bản ghi
  - Để có thể giới hạn số lượng bản ghi được lấy ra khi truy vấn chỉ cần thêm phương thức limit() vào phía sau phương thức find() theo cú pháp sau:

```
db.collectionName.find().limit(number)
```

*Trong đó:

- **number** là số lượng bản ghi bạn muốn lấy ra.

VD: Lấy ra tối đa 5 bản trong collection admin.

```
db.admin.find().limit(5).pretty()
```

+ Bỏ qua số lượng bản ghi
  - Trong một số trường hợp, bạn muốn bỏ qua một hoặc nhiều bản ghi đầu trong số bản ghi được lấy ra thì trong MongoDB đã cung cấp cho chúng ta phương thức skip() để thực hiện điều đó.

```
db.collectionName.find().skip(numberSkip)
```

*Trong đó:

- **numberSkip** là số lượng bản ghi mà bạn muốn bỏ qua. Mặc định thì skip bằng 0

VD: Bỏ qua 3 bản ghi đầu trong số bản ghi được lấy ra.

```
db.admin.find().skip(3)
```

+ Sắp xếp dữ liệu trong MongoDB
  - Để có thể sắp xếp dữ liệu được trả về trong MongoDB thì các bạn sử dụng phương thức sort() theo cú pháp sau:

```
db.admin.find().sort({field: orderMode})
```

*Trong đó:

- **field** là trường chọn để sắp xếp dữ liệu.
- **orderMode** là kiểu sắp xếp. Trong đó:
- Nếu orderMode = 1 là sắp xếp theo chiều tăng dần.
- orderMode = -1 là sắp xếp theo chiều giảm dần.

VD: Lấy ra tất cả các document có trong collection và sắp xếp theo độ dài của trường name giảm dần.

```
db.admin.find().sort({name: -1})
```

7. Sửa dữ liệu

+ MongoDB đã cung cấp 3 phương thức để thực hiện việc sữa đổi dữ liệu. Bao gồm các phương thức sau: **update, updateOne, updateMany.**

a. Phương thức updateOne

+ Cho phép sửa đổi một bản ghi duy nhất trong MongoDB với cú pháp:

```
db.collectionName.updateOne(
   filter,
   update,
   {
      upsert: <boolean>,
      writeConcern: <document>
      collation: <document>,
   }
)
```

*Trong đó:

- **filter** là một object chứa các tiêu chí lựa chọn bản ghi update (sử dụng cú pháp truy vấn dữ liệu).
- **update** là object chứa dữ liệu sửa đổi trên bản ghi.
- **upsert** là một boolean cấu hình điều gì sẽ xảy ra khi không có bản khi khớp với filter. Nếu upsert = true thì nó sẽ thêm mới bản ghi đó nếu không có bản ghi nào khớp với filter và sẽ không có điều gì xảy ra nếu upsert = false. Mặc định thì upsert = false.
- **writeConcern** là một document chứa write concern.
- **collation** là một document chứa các quy tắc.

Lưu ý: Khi sử dụng phương thức **updateOne** nếu như dữ liệu khớp với filter nhiều hơn một bản ghi thì nó sẽ chỉ sửa đổi cho một bản ghi đầu tiên.

VD: Sửa name của admin có tuổi = 18 thành KieuLinh.

```
db.admin.updateOne(
   {age: 18},
   {
      $set: {
         name: "KieuLinh"
      }
   }
)
```

b. Phương thức updateMany

+ Cho phép sửa đổi nhiều bản ghi trên một lần khai báo trong MongoDB với cú pháp tương tự như phương thức updateOne.

Lưu ý: Phương thức này chỉ khác với phương thức updateOne() ở chỗ: Nếu như số lượng bản ghi so khớp với filter lớn hơn 1 bản ghi thì nó sẽ sửa dữ liệu trên tất cả các bản ghi đó.

c. Phương thức update

+ Phương thức update() có thể cấu hình updateOne() hoặc updateMany(), sử dụng theo cú pháp sau:

```
db.collection.update(
   filter,
   update,
   {
     upsert: <boolean>,
     multi: <boolean>,
     writeConcern: <document>,
     collation: <document>
   }
)
```

*Trong đó: các thuộc tính còn lại được định nghĩa giống phương thức updateOne, có thêm thuộc tính multi được định nghĩa sau đây:

- **multi** là một boolean cấu hình xem có cho phép sửa đổi nhiều bản ghi hay không, multi bằng true là cho phép và ngược lại bằng false thì là không. Mặc định thì thuộc tính này có giá trị là false.
  VD: Sửa đổi name của một bản ghi duy nhất có name là "*Linh*" thành "*KieuLinh*

```
db.admin.updateOne(
   {name: "Linh"},
   {
      $set: {
         name: "KieuLinh"
      }
   },
   {
      multi : false
   }
)
```

8. Xóa dữ liệu

+ Để có thể xóa dữ liệu trong MongoDB thì sử dụng phương thức remove() với cú pháp sau:

```
db.collectionName.remove(
  query,
  {
    justOne: <boolean>,
    writeConcern: <document>,
    collation: <document>
  }
)
```

*Trong đó**:**

- **CollectionName** là tên của collection mà bạn muốn thực thi lệnh xóa document.
- query là object (hay còn gọi là document) chứa các **câu truy vấn** để lọc dữ liệu.
- **justOne** là tham số cấu hình số lượng bản ghi có thể xóa khi query thực thi khớp.
- Nếu **justOne**: true thì nó sẽ chỉ xóa 1 bản ghi duy nhất.
- Nếu **justOne**: false thì nó sẽ xóa tất cả các bản ghi khớp với điều kiện query.
- **writeConcern** là một document chứa write concern.
- **collation** là một document chứa các quy tắc.

VD: Xóa một admin có name = "*KieuLinh*" và có age = 18.

```
db.admin.remove(
  {
    name: "KieuLinh",
    age: 18
  }
)
```

IV. CÁCH THỨC TRUY VẤN & XỬ LÝ DỮ LIỆU CỦA MONGODB TRÊN NGÔN NGỮ LẬP TRÌNH PHP

Mặc định PHP không hỗ trợ cho MongoDB, nên để có thể sử dụng MongoDB cần phải cài đặt và thiết lập MongoDB vào PHP. Sau khi MongoDB đã được cài đặt, để thực hiện kết nối PHP đến MongoDB cần sử dụng đến class *Client* nằm trong gói *MongoDB*.

```
<?php
        use MongoDB\Client;
?>
```

Class này bao gồm các phương thức:

- __construct()
- __get()
- dropdatabase()
- getManager()
- getReadConcern()
- getReadPreference()
- getTypeMap()
- getWriteConcern()
- listDatabases()
- selectCollection()
- selectDatabase()

1. Thực thi kết nối:

+ Để sử dụng MongoDB, cần phải require file *autoload.php* trong thư mục *vendor*.

```
<?php
                require 'vendor/autoload.php';
?>
```

+ Khởi tạo class *Client* với cú pháp:

```
new Client($uri, $option, $driverOption);
```

*Trong đó:

- *$uri* là đường dẫn đến MongoDB cần kết nối. Mặc định thì *$uri* = *'mongodb://127.0.0.1/'*
- *$option* là mảng chứa các thông số cấu hình tùy chọn thêm cho *$uri*, các thông số này sẽ ghi đè các thông số mặc định trong $uri.
- *$driverOption* là mảng chứa các thông số cấu hình driver thêm cho *$uri* như SSL,...

Ví dụ:

```
<?php
    use MongoDB\Client;
    require 'vendor/autoload.php';
    $conn = new Client("mongodb://127.0.0.1:27017");
?>
```

Trong ví dụ ở trên: MongoDB được chạy trên port *27017* , khởi tạo biến *$conn* ánh xạ tới class *Client* chứa kết nối đến MongoDB và không có các tham số *$option*, *$driverOption*.

2. Truy xuất database trong MongoDB bằng PHP

+ Để select đến database trong MongoDB bằng PHP sử dụng phương thức *selectDatabase()* trong class *Client* với cú pháp:

```
$conn->selectDatabase($databaseName, $options);
```

*Trong đó:

- *$conn* là biến ánh xạ class *Client* đã được khởi tạo.
- *$database* là tên database muốn select.
- *$options* là mảng chứa các thông số cấu hình như: readConcern, typemap…

Ví dụ: Select database qlKhoGao

```
<?php
    use MongoDB\Client;
    require 'vendor/autoload.php';
    $conn = new Client("mongodb://127.0.0.1:27017");
    $db = $conn->selectDatabase('qlKhoGao');
?>
```

*Trong ví dụ trên, select database qlKhoGao trong MongoDB. Ngoài ra, có thể select database thông qua phương thức *__get()* trong class *Client* sau:

```
$db = $conn->qlKhoGao;
```

+ Để xem tất cả database có trong MongoDB sử dụng phương thức *listDatabases()* theo cú pháp:

```
$conn->listDatabases();
```

+ Để xóa một database khỏi MongoDB sử dụng phương thức *dropDatabase()* theo cú pháp:

```
$conn->dropDatabase(name, options);
```

*Trong đó:

- *$conn* là biến ánh xạ class *Client* đã được khởi tạo.
- *name* là tên database muốn xóa
- *options* mảng chứa các tham số tùy chỉnh như: typeMap, writeConcern…

3. Truy xuất Collection trong MongoDB bằng PHP

+ Để truy xuất đến một collection trong MongoDB sử dụng phương thức *selectCollection()* trong class *Client* với cú pháp:

```
$conn->selectCollection($databaseName, $collectionName, $options);
```

*Trong đó:

- *$conn* là biến ánh xạ class *Client* đã được khởi tạo.
- *$databaseName* là tên database muốn select
- *$collectionName* là tên collection muốn select ở trong database
- *$options* là mảng chứa các thông số cấu hình như readConcern, typeMap…

Ví dụ: Select collection NhaCungCap trong database qlKhoGao

```
<?php
    use MongoDB\Client;
    require 'vendor/autoload.php';
    $conn = new Client("mongodb://127.0.0.1:27017");
    $collection = $conn->selectCollection('qlKhoGao', 'NhaCungCap');
?>
```

*Trong ví dụ trên, sau khi khởi tạo $conn ánh xạ class Client, sử dụng phương thức selectCollection() để truy xuất đến collection NhaCungCap trong database qlKhoGao.

+ Ngoài ra, có thể sử dụng phương thức *selectCollection()* trong class *Database* nhưng trước đó phải select đến database rồi, cú pháp:

```
$db->selectCollection($collectionName, $options);
```

*Trong đó:

- *$db* là biến ánh xạ đến database.
- *$collectionName* là tên collection muốn select ở trong database

- *$options* là mảng chứa các thông số cấu hình như readConcern, typeMap…

```
<?php
    use MongoDB\Client;
    require 'vendor/autoload.php';
    $conn = new Client("mongodb://127.0.0.1:27017");
    $db = $conn→qlKhoGao;
    $collection = $db→selectCollection('NhaCungCap');
?>
```

+ Trong ví dụ trên, sau khi truy xuất đến database *qlKhoGao* bằng phương thức *__get()* của class *Client*, dùng phương thức *selectCollection()* của class Database truy xuất đến collection tên NhaCungCap.

+ Hoặc có thể dùng phương thức *__get()* của class *Client* để truy xuất đến collection, ví dụ:

```
<?php
    use MongoDB\Client;
    require 'vendor/autoload.php';
    $conn = new Client("mongodb://127.0.0.1:27017");
    $collection = $conn→qlKhoGao→NhaCungCap;
?>
```

+ Để hiển thị tất cả collection có trong database sử dụng phương thức *listCollections()* theo cú pháp:

```
$db→listCollections();
```

+ Để xóa một collection khỏi database sử dụng phương thức *dropCollection()* theo cú pháp:

```
$db→dropCollection($collectionName, $options);
```

*Trong đó:

- *$db* là biến ánh xạ đến database
- *$collectionName* là tên collection muốn select ở trong database
- *$options* là mảng chứa tham số tùy chỉnh như writeConcern, typeMap…

4. Xử lý dữ liệu trong MongoDB bằng PHP

a. Thêm dữ liệu:

+ Có 2 phương thức giúp thêm dữ liệu vào trong các collection bằng PHP:

  - insertOne(): cho phép thêm một document vào collection.
  - insertMany(): cho phép thêm nhiều document vào collection.

+ Cú pháp:

```
$collection→insertOne($data, $options);
$collection→insertMany($data, $options);
```

*Trong đó:

- *$collection* là objection chứa collection đã trỏ tới.
- *$data* là mảng dữ liệu muốn thêm vào collection.
- *$options* là mảng chứa các thông số cấu hình: writeConcern,…

Ví dụ 1: Thêm một document vào collection NhaCungCap trong database qlKhoGao

```
<?php
    use MongoDB\Client;
    require 'vendor/autoload.php';
    $conn = new Client("mongodb://127.0.0.1:27017");
    $collection = $conn→qlKhoGao→NhaCungCap;
    $insertResult = $collection->insertOne([
        'maNCC' => 3,
        'tenNCC' => 'Lotus Rice VietNam',
        'diaChi' => '52 Đường số 10, Khu đô thị Him Lam, Quận 7, Hồ Chí Minh',
        'sdt' => '0937352617'
    ]);
    var_dump($insertResult);
?>
```

**Trong ví dụ trên: sau khi thực hiện phương thức insertOne(), hiển thị thông tin của biến $insertResult ra màn hình bằng lệnh var_dump().*

+ Và MongoDB sẽ trả về *Write Result Classes* có các thông số như sau nếu dữ liệu được thêm thành công. Còn ngược lại nếu như không thành công thì nó sẽ trả về một *error exception*.

```
object(MongoDB\InsertOneResult)[14]
  private 'writeResult' =>
    object(MongoDB\Driver\WriteResult)[13]
      public 'nInserted' => int 1
      public 'nMatched' => int 0
      public 'nModified' => int 0
      public 'nRemoved' => int 0
      public 'nUpserted' => int 0
      public 'upsertedIds' =>
        array (size=0)
          empty
      public 'writeErrors' =>
        array (size=0)
          empty
      public 'writeConcernError' => null
      public 'writeConcern' =>
        object(MongoDB\Driver\WriteConcern)[11]
  private 'insertedId' =>
    object(MongoDB\BSON\ObjectId)[12]
      public 'oid' => string '5dd17bf75d9cd5224c004f6b' (length=24)
  private 'isAcknowledged' => boolean true
```

Hình: Write Result Classes khi insert document thành công

Ví dụ 2: Thêm nhiều document vào collection NhaCungCap .

```php
<?php
    use MongoDB\Client;
    require 'vendor/autoload.php';
    $conn = new Client("mongodb://127.0.0.1:27017");
    $collection = $conn→qlKhoGao→NhaCungCap;
    $insertResult = $collection->insertMany([
        ['maNCC' => 3,
        'tenNCC' => 'Lotus Rice VietNam',
        'diaChi' => '52 Đường số 10, Khu đô thị Him Lam, Quận 7, Hồ Chí Minh',
        'sdt' => '0937352617'],
        ['maNCC' => 4,
        'tenNCC' => 'GREENFARMV',
        'diaChi' => '113 Nguyễn Hữu Cảnh Phường 22 Quận Bình Thạnh, TP. HCM, Việt Nam',
        'sdt' => '0937352617']
    ]);
    var_dump($insertResult);
?>
```

+ Và MongoDB sẽ trả về *Write Result Classes* có các thông số như sau nếu dữ liệu được thêm thành công. Còn ngược lại nếu như không thành công thì nó sẽ trả về một *error exception*.

```
object(MongoDB\InsertOneResult)[14]
  private 'writeResult' =>
    object(MongoDB\Driver\WriteResult)[13]
      public 'nInserted' => int 1
      public 'nMatched' => int 0
      public 'nModified' => int 0
      public 'nRemoved' => int 0
      public 'nUpserted' => int 0
      public 'upsertedIds' =>
        array (size=0)
          empty
      public 'writeErrors' =>
        array (size=0)
          empty
      public 'writeConcernError' => null
      public 'writeConcern' =>
        object(MongoDB\Driver\WriteConcern)[11]
  private 'insertedId' =>
    object(MongoDB\BSON\ObjectId)[12]
      public 'oid' => string '5dd17bf75d9cd5224c004f6b' (length=24)
  private 'isAcknowledged' => boolean true
```

Hình: Write Result Classes khi insert document thành công

Ví dụ 2: Thêm nhiều document vào collection NhaCungCap .

+ Tương tự như phương thức *insertOne()*, sau khi thêm nhiều document thành công, kết quả trả về *Write Result Class* như sau:

```
object(MongoDB\InsertManyResult)[15]
  private 'writeResult' =>
    object(MongoDB\Driver\WriteResult)[14]
      public 'nInserted' => int 2
      public 'nMatched' => int 0
      public 'nModified' => int 0
      public 'nRemoved' => int 0
      public 'nUpserted' => int 0
      public 'upsertedIds' =>
        array (size=0)
          empty
      public 'writeErrors' =>
        array (size=0)
          empty
      public 'writeConcernError' => null
      public 'writeConcern' =>
        object(MongoDB\Driver\WriteConcern)[11]
  private 'insertedIds' =>
    array (size=2)
      0 =>
        object(MongoDB\BSON\ObjectId)[12]
          public 'oid' => string '5dd182035d9cd5224c004f6e' (length=24)
      1 =>
        object(MongoDB\BSON\ObjectId)[13]
          public 'oid' => string '5dd182035d9cd5224c004f6f' (length=24)
  private 'isAcknowledged' => boolean true
```

Hình: Write Result Classes khi insert nhiều document thành công

b. Sửa dữ liệu

+ Có 2 phương thức để update dữ liệu trong các collection bằng PHP:

  - updateOne(): cho phép sửa (update) một document.
  - updateMany(): cho phép sửa (update) nhiều document.

+ Cú pháp:

```
$collection->updateOne($filter, $update, $options);
$collection->updateMany($filter, $update, $options);
```

*Trong đó:

  - *$filter* là mảng dữ liệu muốn so khớp, lọc (mảng dữ liệu muốn sửa).
  - *$ update* là mảng dữ liệu muốn thay thế cho dữ liệu được filter (mảng dữ liệu sửa).

- *$options* là mảng chứa các thông số tùy chỉnh thêm.

Chú ý: Với phương thức updateOne(), nếu điều kiện filter trả về nhiều hơn một document thì dữ liệu chỉ được thay thế cho document đầu tiên.

Ví dụ 1: Sửa thông tin nhà cung cấp có *_id* là *5dd182035d9cd5224c004f6e*

```
<?php
    use MongoDB\Client;
    require 'vendor/autoload.php';
    $conn = new Client("mongodb://127.0.0.1:27017");
    $collection = $conn→qlKhoGao→NhaCungCap;
    $updateResult = $collection->updateOne(
      [ '_id' => new
      MongoDB\BSON\ObjectId('5dd182035d9cd5224c004f6e' ) ],
      [ '$set' =>[
      'tenNCC' => 'Gạo sạch Thanh Hải']
      ]
    );
    var_dump($updateResult);
?>
```

+ Phương thức update sẽ trả về một đối tượng *Update Result* có các thông số như sau nếu dữ liệu được sửa thành công. Còn ngược lại nếu như không thành công thì sẽ trả về một *error exception*.

```
object(MongoDB\UpdateResult)[15]
  private 'writeResult' =>
    object(MongoDB\Driver\WriteResult)[14]
      public 'nInserted' => int 0
      public 'nMatched' => int 1
      public 'nModified' => int 1
      public 'nRemoved' => int 0
      public 'nUpserted' => int 0
      public 'upsertedIds' =>
        array (size=0)
          empty
      public 'writeErrors' =>
        array (size=0)
          empty
      public 'writeConcernError' => null
      public 'writeConcern' =>
        object(MongoDB\Driver\WriteConcern)[13]
  private 'isAcknowledged' => boolean true
```

Hình: Kết quả khi update một document thành công

+ Sử dụng tương tự như phương thức *updateOne()*, *updateMany()* sửa nhiều document và trả về một đối tượng *Update Result* có các thông số như sau nếu dữ liệu được sửa thành công, ví dụ 2:

```
object(MongoDB\UpdateResult)[14]
  private 'writeResult' =>
    object(MongoDB\Driver\WriteResult)[13]
      public 'nInserted' => int 0
      public 'nMatched' => int 2
      public 'nModified' => int 2
      public 'nRemoved' => int 0
      public 'nUpserted' => int 0
      public 'upsertedIds' =>
        array (size=0)
          empty
      public 'writeErrors' =>
        array (size=0)
          empty
      public 'writeConcernError' => null
      public 'writeConcern' =>
        object(MongoDB\Driver\WriteConcern)[12]
  private 'isAcknowledged' => boolean true
```

Hình: Kết quả khi update nhiều document thành công

c. Xóa dữ liệu

+ Có 2 phương thức để xóa dữ liệu của một collection bằng PHP:

- *deleteOne()*: cho phép xóa một document của collection.
- *deleteMany()*: cho phép xóa nhiều document của collection.

+ Cú pháp:

```
$collection->deleteOne($filter, $options);
$collection->deleteMany($filter, $options);
```

*Trong đó:

- *$filter* là mảng hoặc object chứa các điều kiện so khớp.
- *$options* là mảng chứa các thông số tùy chỉnh.

Chú ý: Với phương thức deleteOne(), câu lệnh này chỉ xóa duy nhất 1 document đầu tiên mà kết quả so khớp trả về.

Ví dụ: Xóa nhà cung cấp có tên Gạo Vinh Hiển

```
<?php
    use MongoDB\Client;
    require 'vendor/autoload.php';
    $conn = new Client("mongodb://127.0.0.1:27017");
    $collection = $conn->qlKhoGao->NhaCungCap;
    $deleteResult = $collection->deleteMany(
      [ 'tenNCC' => 'Gạo Vinh Hiển' ]);
    echo $deleteResult->getDeletedCount();
?>
```

+ Trong ví dụ trên, câu lệnh xóa tương đương với câu lệnh *DELETE FROM NhaCungCap WHERE tenNCC = N'Gạo Vinh Hiển'* trong SQL, sau khi thực hiện câu lệnh xóa thì xuất ra màn hình số lượng document đã xóa bằng phương thức *getDeletedCount()*. Kết quả lúc này sẽ trả về 2. Tuy nhiên nếu sử dụng phương thức *deleteOne()*, kết quả sẽ trả về 1, phương thức *deleteOne()* sẽ chỉ xóa document đầu tiên trả về.

5. Truy vấn dữ liệu trong MongoDB bằng PHP

+ Để truy vấn dữ liệu trong MongoDB sử dụng 2 phương thức *findOne()* và *find()* với chức năng như sau:
  - *findOne()*: Truy vấn và trả về lớn nhất 1 document khớp với điều kiện truy vấn.
  - *find()*: Truy vấn và trả về tất cả document khớp với điều kiện truy vấn.

+ Cú pháp:

```
$collection->findOne($filter, $options);
$collection->find($filter, $options);
```

*Trong đó:

- *$ filter* là mảng chứa các điều kiện muốn truy vấn.
- *$options* là mảng chứa các thông số tùy chỉnh như chỉ định các trường được lấy ra (select) hoặc limit… (tham số này có thể được bỏ trống).

Ví dụ: Tìm các nhà cung cấp có tên là "Gạo Vinh Hiển"

```
<?php
    use MongoDB\Client;
    require 'vendor/autoload.php';
    $conn = new Client("mongodb://127.0.0.1:27017");
    $collection = $conn->qlKhoGao->NhaCungCap;
    $query = $collection->findOne(
      [ 'tenNCC' => 'Gạo Vinh Hiển' ]);
?>
```

*Trong ví dụ trên, sau khi câu lệnh *findOne()* được thực hiện, câu lệnh chỉ lấy duy nhất document đầu tiên so khớp với điều kiện. Kết quả trả về là một Object, để lấy giá trị của document thì cần trỏ *$query* đến các key trong database như sau:

```
<?php
    use MongoDB\Client;
    require 'vendor/autoload.php';
    $conn = new Client("mongodb://127.0.0.1:27017");
    $collection = $conn->qlKhoGao->NhaCungCap;
    $query = $collection->findOne(
      [ 'tenNCC' => 'Gạo Vinh Hiển' ]);
    $query->tenNCC // kết quả: Gạo Vinh Hiển
?>
```

+ Cũng với ví dụ trên nếu thay thế *findOne()* thành *find()*, phương thức này sẽ trả về tất cả document có *tenNCC* là "*Gạo Vinh Hiển*". Kết quả trả về là một Object, để lấy kết quả trả về cần phải dùng phương thức *toArray()* để chuyển kết quả thành mảng hoặc dùng vòng lặp *foreach*, ví dụ:

```
//Cách 1: dùng foreach:
    foreach($query as $row){
        $row->_id;
        $row->tenNCC;
        $row->diaChi;
    }
```

```
//Cách 2: dùng toArray()
print_r($query->toArray())
```

+ Các tham số options có thể thêm vào trong phương thức *find()* và *findOne()* như:

- projection: mảng chứa các field muốn select
- skip: số lượng document bỏ qua
- limit: giới hạn document được lấy ra

…

+ Ví dụ:
  - Chỉ lấy field diaChi của nhà cung cấp tên "Gạo Vinh Hiển"

```
$collection = $conn->qlKhoGao->NhaCungCap;
$query = $collection->findOne(
    [ 'tenNCC' => 'Gạo Vinh Hiển' ],
    [ 'projection' =>[ 'diaChi' => '1' ]]);
```

Câu query trên tương ứng với câu truy vấn SQL sau:

SELECT diaChi FROM NhaCungCap WHERE tenNCC = 'Gạo Vinh Hiển'

- Tìm kiếm document có tenNCC chứa chữ "vi"

```
$collection = $conn->qlKhoGao->NhaCungCap;
$query = $collection->findOne(
    [
    'tenNCC' => [ '$regex' => 'vi', '$options' => 'i' ]
    ]);
```

Câu lệnh query trên tương ứng với câu truy vấn SQL sau:

*SELECT * FROM NhaCungCap WHERE tenNCC LIKE '%vi%'*

V. DEMO XÂY DỰNG HỆ THỐNG QUẢN LÝ KHO GẠO VỚI MONGODB BẰNG NGÔN NGỮ LẬP TRÌNH PHP

1. Hiển thị danh sách gạo

DANH SÁCH GẠO

TÊN GẠO	LOẠI GẠO	NHÀ CUNG CẤP	NGÀY NHẬP	KHO	TÌNH TRẠNG	THAO TÁC
Gao Tam Thai Do Bao Minh	Gạo dẻo	Quý Thu Rice Q&T	2019-11-20	Kho 1	Còn hàng	Xem
Gạo thơm hương lài	Gạo nở	Vinamic Organic Farm	2019-11-20	Kho 2	Còn hàng	Xem
Gạo thơm Jasmine	Gạo dẻo	Vinamic Organic Farm	2019-11-20	Kho 1	Còn hàng	Xem
Gạo Thái Lan	Gạo xốp	Quý Thu Rice Q&T	2019-11-20	Kho 2	Còn hàng	Xem
Gạo trắng hạt dài	Gạo nở	Quý Thu Rice Q&T	2019-11-20	Kho 2	Còn hàng	Xem

THÊM GẠO TÌM KIẾM GẠO

Hình: hiển thị danh sách các gạo ở tất cả các kho

2. Chức năng thêm gạo

Danh sách gạo

THÊM GẠO

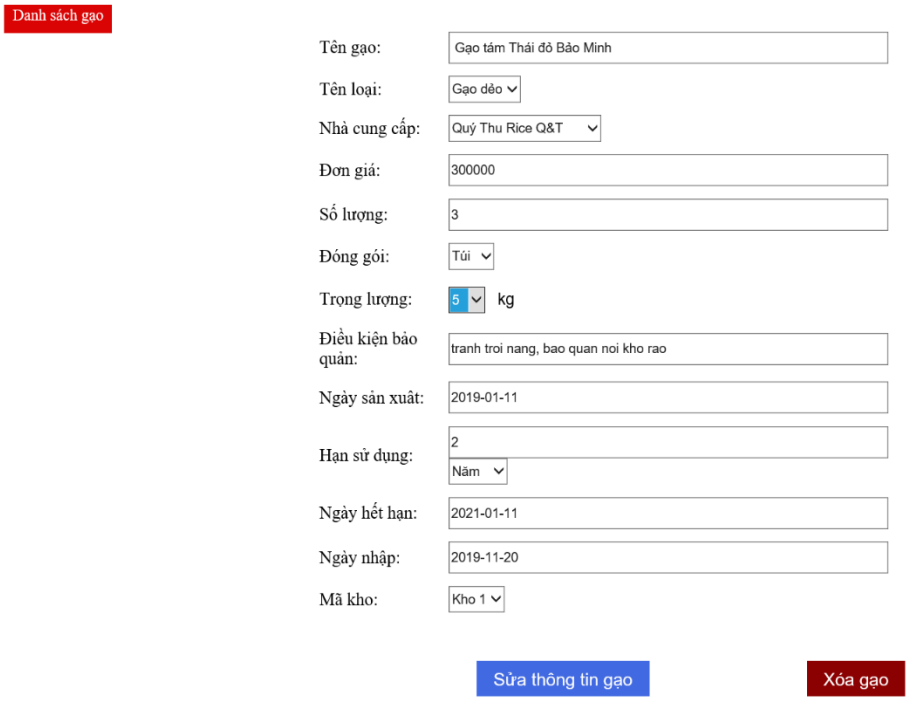
Tên gạo: Gạo Thái Lan
Tên loại: Gạo xốp
Nhà cung cấp: Quý Thu Rice Q&T
Đơn giá: 108500
Số lượng: 3
Đóng gói: Bao
Trọng lượng: 5 kg
Điều kiện bảo quản: Nơi thoáng mát, tránh gió và côn trùng.
Ngày sản xuất: 2019-10-29
Hạn sử dụng: 10 Tháng
Mã kho: Kho 2

Thêm gạo mới

Hình: giao diện chức năng thêm gạo

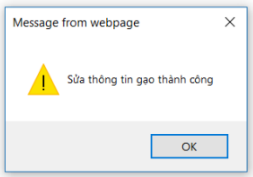
Hình: thông báo sau khi thêm gạo thành công

3. Chức năng sửa gạo



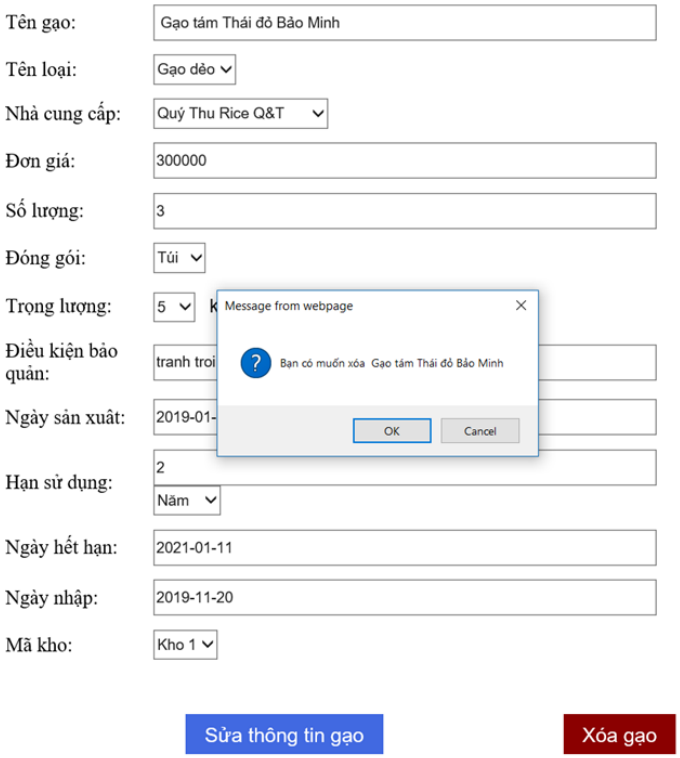
Hình: Giao diện chức năng sửa thông tin gạo

THÔNG TIN GẠO



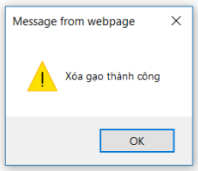
Hình: thông báo sau khi sửa thông tin gạo

4. Chức năng xóa gạo



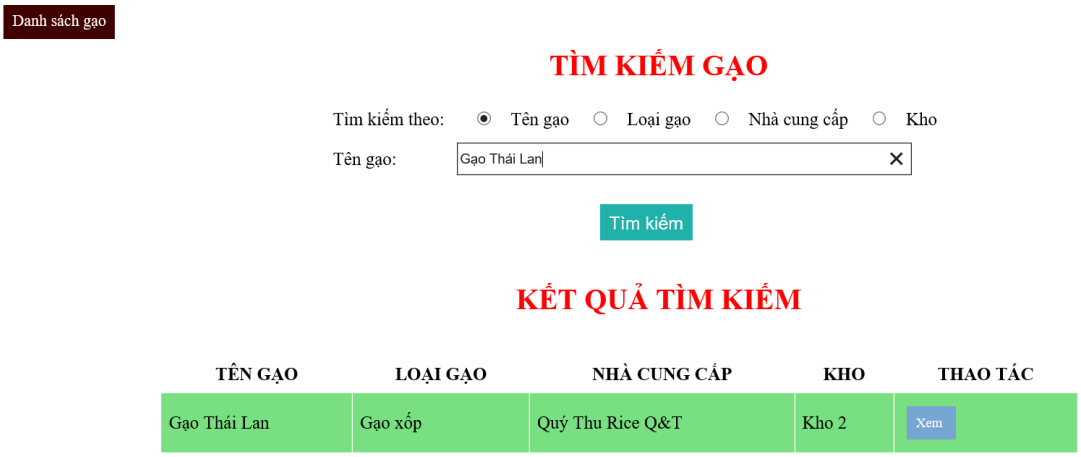
Hình Giao diện chức năng xóa gạo

THÔNG TIN GẠO



Hình: thông báo sau khi xóa gạo thành công

5. Chức năng tìm kiếm gạo
   a. Tìm kiếm gạo theo tên gạo



Hình: kết quả tìm kiếm gạo theo tên gạo

   b. Tìm kiếm gạo theo loại gạo

Danh sách gạo

TÌM KIẾM GẠO

Tìm kiếm theo: ○ Tên gạo ◉ Loại gạo ○ Nhà cung cấp ○ Kho

Tên loại: Gạo dẻo

Tìm kiếm

KẾT QUẢ TÌM KIẾM

TÊN GẠO	LOẠI GẠO	NHÀ CUNG CẤP	KHO	THAO TÁC
Gạo thơm hương lài	Gạo nở	Vinamic Organic Farm	Kho 2	Xem
Gạo trắng hạt dài	Gạo nở	Quý Thu Rice Q&T	Kho 2	Xem

Hình: kết quả tìm kiếm gạo theo loại gạo

c. Tìm kiếm gạo theo nhà cung cấp

Danh sách gạo

TÌM KIẾM GẠO

Tìm kiếm theo: ○ Tên gạo ○ Loại gạo ◉ Nhà cung cấp ○ Kho

Nhà cung cấp: Quý Thu Rice Q&T

Tìm kiếm

KẾT QUẢ TÌM KIẾM

TÊN GẠO	LOẠI GẠO	NHÀ CUNG CẤP	KHO	THAO TÁC
Gạo thơm hương lài	Gạo nở	Vinamic Organic Farm	Kho 2	Xem
Gạo trắng hạt dài	Gạo nở	Quý Thu Rice Q&T	Kho 2	Xem

Hình: kết quả tìm kiếm theo nhà cung cấp gạo

d. Tìm kiếm gạo theo kho

Hình: kết quả tìm kiếm theo tên kho gạo