Đại học Quốc gia Hà Nội
Trường Đại học Khoa học Tự nhiên
-----*-----**

BÁO CÁO GIỮA KỲ HỌC PHẦN CƠ SỞ DỮ LIỆU

TÌM HIỂU VỀ MONGODB

Giảng viên : **T.S Vũ Tiến Dũng**
Thành viên nhóm : **Nguyễn Hà Nam - 21002160**
Lê Quang Đạt - 21002129
Lê Quốc Lâm - 21002153
Lê Thị Phương - 21002167
Bùi Nguyên Phong - 21002165

Hà Nội - 2024

Mục lục

Mục lục hình ảnh

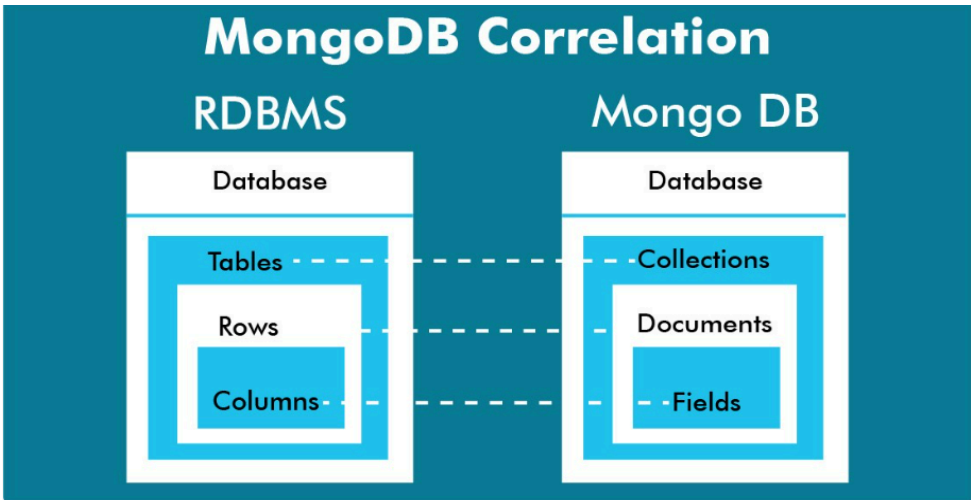
I. Giới thiệu về MongoDB và kiến trúc cơ bản

1.1 Khái niệm về MongoDB

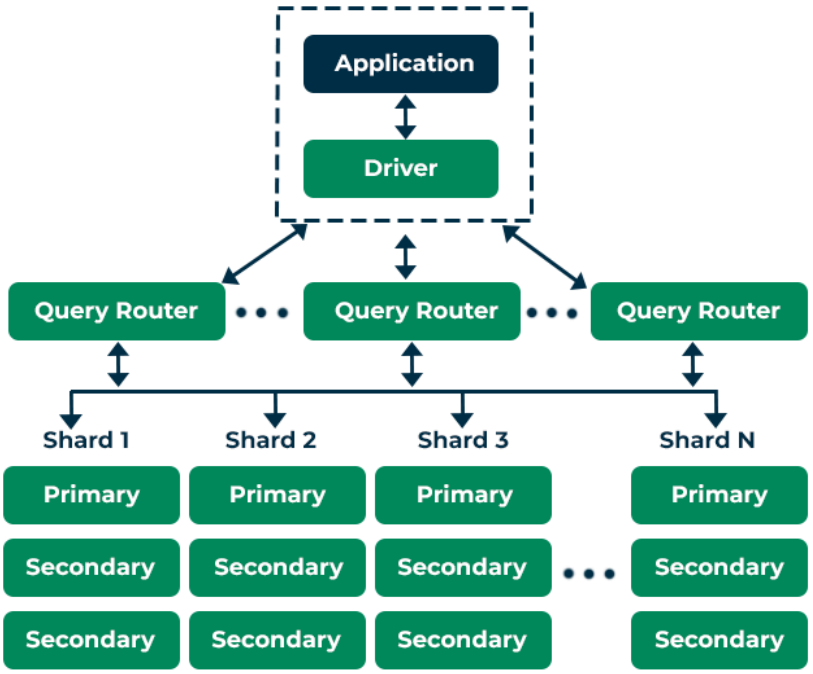
MongoDB lần đầu ra đời bởi MongoDB Inc., tại thời điểm đó là thế hệ 10, vào tháng Mười năm 2007, nó là một phần của sản phẩm PaaS (Platform as a Service) tương tự như Windows Azure và Google App Engine. Sau đó nó đã được chuyển thành nguồn mở từ năm 2009.

MongoDB là một hệ quản trị cơ sở dữ liệu NoSQL (Not Only SQL) mã nguồn mở được phát triển bởi MongoDB Inc. Đây là một trong những hệ thống cơ sở dữ liệu phổ biến nhất hiện nay nhờ khả năng lưu trữ linh hoạt, hiệu năng cao, và dễ dàng mở rộng. Khác với cơ sở dữ liệu quan hệ truyền thống (RDBMS) như MySQL hay PostgreSQL, MongoDB không dựa trên bảng (tables) mà sử dụng mô hình tài liệu (document-oriented).

Trong MongoDB, dữ liệu được lưu trữ dưới dạng Document, sử dụng định dạng JSON (JavaScript Object Notation) hoặc BSON (Binary JSON). Điều này giúp MongoDB trở nên lý tưởng cho việc lưu trữ các loại dữ liệu phi cấu trúc hoặc bán cấu trúc, mà không cần phải định nghĩa trước cấu trúc dữ liệu (schema).



Hình 1: Sự khác nhau giữa RDMS và Mongo DB



Hình 2: Kiến trúc của mongoDB

Quy trình request từ ứng dụng đến các shard trong MongoDB:

a. *Ứng dụng gửi request:*

- Ứng dụng gửi truy vấn hoặc thao tác cơ sở dữ liệu (insert, find, update, delete) đến Query Router (mongos).

b. *Query Router xử lý:*

- mongos nhận request và:
  - Tra cứu thông tin metadata từ Config Servers để xác định:
    - Dữ liệu cần truy vấn nằm ở shard nào.
    - Khóa phân mảnh (shard key) được sử dụng.
  - Chuyển tiếp truy vấn đến shard phù hợp.

c. *Shard nhận request:*

- Mỗi shard có thể là một replica set (bao gồm một Primary và các Secondary).
- Primary shard chịu trách nhiệm xử lý các truy vấn ghi (write).
- Các Secondary shards phục vụ các truy vấn đọc (read), nếu được cấu hình.

d. *Thực hiện truy vấn:*

- Shard được xác định thực hiện truy vấn:
  - Nếu dữ liệu chỉ thuộc một shard: Shard đó xử lý trực tiếp.
  - Nếu dữ liệu phân tán trên nhiều shard: Tất cả các shard liên quan sẽ thực hiện truy vấn song song.

e. Kết hợp kết quả:

- Kết quả từ một hoặc nhiều shard được gửi về mongos.
- mongos tập hợp kết quả và gửi lại cho ứng dụng.

Khi nào sử dụng MongoDB:

- **Quản lý và truyền tải content** – Quản lý đa dạng nhiều product của content chỉ trong một kho lưu trữ data cho phép thay đổi và phản hồi nhanh chóng mà không chịu thêm phức tạp thêm từ hệ thống content.
- **Cấu trúc Mobile và Social** – MongoDB cung cấp một platform có sẵn, phản xạ nhanh, và dễ mở rộng cho phép rất nhiều khả năng đột phá, phân tích real-time, và hỗ trợ toàn cầu.
- **Quản lý data khách hàng** – Tận dụng khả năng query nhanh chóng cho phân tích real-time trên cơ sở dữ liệu người dùng cực lớn với các mô hình data phức tạp bằng các schema linh hoạt và tự động sharding cho mở rộng chiều ngang.

1.2. Các thành phần cơ bản của MongoDB

Thay vì sử dụng bảng và dòng như trong cơ sở dữ liệu quan hệ, kiến trúc của MongoDB (một cơ sở dữ liệu NoSQL) được xây dựng từ các bộ sưu tập (collections) và tài liệu (documents). Tài liệu được tạo thành từ các cặp khóa-giá trị (key-value pairs) - đơn vị dữ liệu cơ bản của MongoDB. Các bộ sưu tập, tương đương với các bảng trong SQL, chứa các tập hợp tài liệu.

a. Database (Cơ sở dữ liệu):

- Database trong MongoDB là một tập hợp các Collection.
- Tương tự như Database trong hệ thống cơ sở dữ liệu quan hệ, nhưng MongoDB không yêu cầu định nghĩa trước cấu trúc dữ liệu (schema).

b. Collection (Tập hợp):

- Collection là một nhóm các Document liên quan.
- Tương tự như một bảng (Table) trong cơ sở dữ liệu quan hệ, nhưng không yêu cầu các Document trong một Collection phải có cấu trúc giống nhau.
- Một Collection có thể được coi như một thư mục, trong đó chứa nhiều tệp (Document).

c. Document (Tài liệu):

- Document là đơn vị dữ liệu nhỏ nhất trong MongoDB, được lưu trữ dưới dạng JSON hoặc BSON.
- Một Document trong MongoDB tương ứng với một bản ghi (record) trong cơ sở dữ liệu quan hệ. Tuy nhiên, Document linh hoạt hơn vì nó có thể chứa các trường (field) hoặc giá trị (value) có cấu trúc khác nhau.

1.3 Ví dụ minh hoạ về Database trong MongoDB

Giả sử một database tên là school để quản lý dữ liệu của một trường học. Trong MongoDB, database này sẽ có các Collection như students, teachers, và classes. Một Document trong Collection students có thể như sau:

```
{
  "_id": 1,
  "name": "John Doe",
  "age": 30,
  "email": "john.doe@example.com"
}
```

Hình 3: Document 1

```
{
  "_id": 2,
  "name": "Jane Smith",
  "is_active": true
}
```

Hình 4: Document 2

Trong khi đó, nếu sử dụng MySQL để lưu trữ dữ liệu tương tự, cần phải tạo một bảng với cấu trúc cố định trước, và các bản ghi cần tuân thủ theo cấu trúc đó:

```
CREATE TABLE students (
  student_id VARCHAR(10),
  name VARCHAR(50),
  age INT,
  subjects JSON,
  graduated BOOLEAN
);
```

Hình 5: Tạo bảng với cấu trúc cố định

```
INSERT INTO students (student_id, name, age, subjects, graduated)
VALUES ('A001', 'Nguyen Van A', 20, '["Math", "Physics"]', false);
```

Hình 6: Thêm hàng với cấu trúc cho trước

So sánh 2 dạng cơ sở dữ liệu

Tiêu chí	MongoDB	MySQL
Cấu trúc dữ liệu	Dạng JSON/BSON, linh hoạt, không cần schema cố định	Dựa trên bảng, yêu cầu schema.
Hiệu năng	Nhanh với dữ liệu phi cấu trúc hoặc bán cấu trúc	Tốt hơn với dữ liệu có cấu trúc.
Hỗ trợ mở rộng	Theo chiều ngang (sharding)	Chủ yếu theo chiều dọc (scaling up).
Độ phức tạp của quản lý dữ liệu	Đơn giản hơn nhờ không yêu cầu schema.	Phức tạp hơn vì yêu cầu tuân thủ schema.

II. Đặc điểm của MongoDB và một số truy vấn cơ bản

2.1. Đặc điểm của MongoDB

a. Sao lưu và nhân bản dữ liệu (Replication)

i. Một tập hợp bản sao (replica set) bao gồm hai hoặc nhiều phiên bản MongoDB, được sử dụng để cung cấp khả năng sẵn sàng cao.

ii. Tập hợp bản sao bao gồm các máy chủ chính (primary) và phụ (secondary). Máy chủ chính thực hiện các thao tác đọc và ghi, trong khi các bản sao phụ giữ bản sao dữ liệu.

iii. Nếu máy chủ chính gặp sự cố, bản sao phụ sẽ được sử dụng làm máy chủ chính thay thế.

b. Khả năng mở rộng (Scalability)

MongoDB hỗ trợ mở rộng dọc và mở rộng ngang:

i. Mở rộng dọc (Vertical scaling): Bổ sung tài nguyên, như CPU hoặc RAM, vào một máy chủ hiện có.

ii. Mở rộng ngang (Horizontal scaling): Thêm nhiều máy chủ vào hệ thống để tăng tài nguyên lưu trữ và xử lý.

c. Cân bằng tải (Load Balancing)

i. MongoDB có khả năng tự xử lý cân bằng tải mà không cần một hệ thống cân bằng tải riêng biệt.

ii. Điều này được thực hiện thông qua mở rộng dọc hoặc mở rộng ngang, đảm bảo hiệu năng ổn định ngay cả khi lưu lượng truy cập lớn.

d. Không cần cấu trúc dữ liệu cố định (Schema-less)

i. MongoDB là cơ sở dữ liệu không cần cấu trúc dữ liệu cố định (schema-less), nghĩa là bạn không cần thiết lập một khuôn mẫu cố định để quản lý dữ liệu.

ii. Điều này giúp MongoDB linh hoạt trong việc xử lý các loại dữ liệu khác nhau mà không cần điều chỉnh cấu trúc trước.

e. Định hướng tài liệu (Document)

i. Dữ liệu trong MongoDB được lưu trữ dưới dạng tài liệu với các cặp khóa-giá trị (key-value pairs), thay vì các hàng và cột như trong cơ sở dữ liệu SQL.

ii. Phương pháp này mang lại sự linh hoạt cao hơn khi so sánh với cơ sở dữ liệu quan hệ, giúp việc quản lý dữ liệu trở nên dễ dàng và thích ứng với nhiều trường hợp sử dụng.

2.2. Một số truy vấn cơ bản

2.2.1. Chèn dữ liệu

Trong MongoDB, phương thức insert được sử dụng để thêm mới một hoặc nhiều tài liệu (document) vào một collection. Các tài liệu này được lưu trữ dưới dạng JSON hoặc BSON.

Có hai phương thức chính để thực hiện thao tác chèn dữ liệu:

- **insertOne()**: Chèn một tài liệu duy nhất vào collection.

Cú pháp:

```
db.students.insertOne({
  "name": "John Doe",
  "age": 28,
  "subject": "Mathematics",
```

```
    "email": "john.doe@example.com"
})
```

Thao tác này tương đương với câu lệnh SQL sau:

```
INSERT INTO students (name, age, subject, email)
VALUES ('John Doe', 28, 'Mathematics', 'john.doe@example.com');
```

- **insertMany()**: Chèn nhiều tài liệu cùng một lúc, giúp tối ưu hóa khi cần thêm một lượng lớn dữ liệu.

Cú pháp:

```
db.students.insertMany([{
  "name": "Jane Smith",
  "age": 32,
  "subject": ["Physics", "Mathematics"],
  "email": "jane.smith@example.com"
},
{
  "name": "Michael Brown",
  "age": 25,
  "subject": ["Chemistry", "Physics"] ,
  "email": "michael.brown@example.com"
}])
```

Tương đương với câu lệnh SQL sau:

```
INSERT INTO students (name, age, subject, email)
VALUES
 ('Jane Smith', 32, '["Physics", "Mathematics"]',    'jane.smith@example.com'),
 ('Michael Brown', 25, '["Chemistry", "Physics"]', 'michael.brown@example.com');
```

Khi thêm dữ liệu, nếu không chỉ định trường _id, MongoDB sẽ tự động tạo trường _id duy nhất cho từng tài liệu. Phương thức insert không yêu cầu cấu trúc dữ liệu cứng nhắc. Điều này có nghĩa là các tài liệu được chèn vào cùng một collection có thể có cấu trúc khác nhau, phù hợp với tính linh hoạt của MongoDB.

2.2.2. Truy vấn dữ liệu

Phương thức find

Trong MongoDB, phương thức ***find*** được sử dụng để thực hiện các truy vấn dữ liệu với các tham số cần truy vấn . Truy vấn trả về một tập hợp con các bản ghi trong một bộ. Bản ghi trả về được xác định dựa trên đối số truyền vào hàm ***find()*** - được coi là tiêu chí của truy vấn.

- Trả về tất cả bản ghi trong Collection: Không truyền tham số vào trong hàm find(), MongoDB sẽ trả về tất cả các tài liệu có trong collection.

Cú pháp:

```
db.students.find()
```

Tương đương với câu lệnh trong SQL:

SELECT * FROM students;

- Trả về tài liệu với điều kiện cụ thể: Phương thức find() có thể được sử dụng với các cặp key-value làm tham số để chỉ định điều kiện truy vấn. MongoDB sẽ chỉ trả về những tài liệu thỏa mãn điều kiện này.

Ví dụ tìm tất cả các bản ghi trong đó giá trị của "age" là 28 :

```
db.students.find({"age": 28})
```

Tương đương với câu lệnh SQL: SELECT * FROM students WHERE age = 28

- Có thể kết hợp nhiều điều kiện với nhau bằng cách thêm nhiều cặp key - value làm tham số truy vấn, có thể được hiểu là “condition1 AND condition2 AND … AND conditionN.”

Ví dụ để lấy ra các người dùng có độ tuổi 28 và tên là “John Doe”, ta thực hiện truy vấn sau:

```
db.students.find({"name": "John Doe", "age": 28})
```

Thao tác này tương ứng với câu lệnh SQL sau:

```
SELECT * FROM students WHERE age = 28 AND name=“John Doe”
```

Xác định Keys cần trả về

Đôi khi chúng ta không cần lấy ra tất cả các cặp key - value, khi đó, ta có thể thay đổi đối số thứ 2 truyền vào hàm find() để lấy ra Keys mà ta muốn. Value có thể là bất kì sau đây:

- 1 hoặc true để trả về Key đó
- 0 hoặc false để loại Key đó
- toán tử projection khác

Ví dụ, nếu có bộ Collection các thông tin về người dùng và ta chỉ muốn lấy ra bản ghi có key “name” và “email”, ta thực hiện truy vấn:

```
db.students .find({}, {"name" : 1, "email" : 1})
```

Kết quả trả về:

```
{ "_id" : ObjectId('6742b94fbe5e1e58c4468471'),
  "name" : "John Doe",
  "email" : "john.doe@example.com"  }
```

Ta thấy rằng, bản ghi trả về có trường “_id”, “name”, “email” vì “_id” được mặc định trong kết quả trả về.
Nếu không muốn bản ghi trả về “_id”, ta thực hiện truy vấn sau:

```
db.students.find({}, {"name": 1, "_id": 0})
```

Kết quả: {'name': 'John Doe'}

Toán tử so sánh:

Trong MongoDB, các toán tử so sánh được sử dụng để tạo điều kiện lọc trong truy vấn dữ liệu. Một số toán tử phổ biến bao gồm:

- $lt (less than): “nhỏ hơn”: Tìm các giá trị nhỏ hơn giá trị chỉ định.
- $lte (less than or equal): “nhỏ hơn hoặc bằng”: Tìm các giá trị nhỏ hơn hoặc bằng giá trị chỉ định.

- $gt (greater than): "lớn hơn": Tìm các giá trị lớn hơn giá trị chỉ định.
- $gte (greater than or equal): "lớn hơn hoặc bằng": Tìm các giá trị lớn hơn hoặc bằng giá trị chỉ định.
- $ne (not equal): "khác": Tìm các tài liệu có giá trị không bằng giá trị chỉ định.
  Các toán tử này có thể kết hợp với nhau để tạo các điều kiện lọc phức tạp hơn.
  Ví dụ: tìm tất cả người dùng có độ tuổi trong khoảng từ 18 đến 30, ta thực hiện truy vấn:

```
db.students.find({"age": {"$gte": 18, "$lte": 30 } })
```

Truy vấn này tương đương với câu lệnh SQL:

```
SELECT * FROM students WHERE age >= 18 AND age <= 30
```

Dạng truy vấn này cũng thường sử dụng với kiểu dữ liệu ngày tháng. Ví dụ, để tìm người mà đã đăng ký trước ngày 1/1/2007, ta sử dụng truy vấn:

```
startDate = new Date("01/01/2007");
db.students.find({ "registered": { "$lt": startDate } })
```

Toán tử logic

Toán tử logic trong MongoDB được sử dụng để kết hợp nhiều điều kiện trong một truy vấn, cho phép tạo các truy vấn phức tạp hơn với nhiều điều kiện.
Các toán tử logic phổ biến bao gồm:

- $and: Toán tử này kết nối nhiều điều kiện và yêu cầu tất cả các điều kiện phải được thỏa mãn.

Ví dụ: Tìm tất cả người dùng có độ tuổi từ 18 đến 30.

```
db.students.find({"$and": [{"age": {"$gte": 18}}, {"age": {"$lte": 30}}]})
```

- $or: Toán tử này kết nối nhiều điều kiện và yêu cầu ít nhất một trong các điều kiện phải được thỏa mãn.

Ví dụ: Tìm tất cả người dùng có độ tuổi từ 18 trở lên hoặc dưới 30.

```
db.students.find({"$or": [{"age": {"$gte": 18}}, {"age": {"$lte": 30}}]})
```

- $nor: Toán tử này là sự phủ định của toán tử $or. Nó chỉ trả về các tài liệu không khớp với bất kỳ điều kiện nào trong mảng.

Ví dụ: Tìm tất cả người dùng có độ tuổi ngoài khoảng từ 18 đến 30.

```
db.students.find({"$nor": [{"age": {"$gte": 18}}, {"age": {"$lte": 30}}]})
```

- $not: Toán tử này phủ định kết quả của một điều kiện.

Ví dụ: Tìm tất cả người dùng có độ tuổi dưới 30.

```
db.students.find({ "age": { "$not": { "$gte": 30 } } })
```

Toán tử tìm kiếm chuỗi ($regex)

Toán tử $regex trong MongoDB được sử dụng để tìm kiếm chuỗi theo mẫu (pattern matching), cho phép tìm kiếm các giá trị chuỗi có thể khớp với một mẫu nhất định. Nó rất hữu ích khi cần thực hiện các tìm kiếm linh hoạt, chẳng hạn như tìm kiếm các chuỗi con trong một trường dữ liệu chuỗi.
Cú pháp:

```
db.collection.find({ "field": { "$regex": "pattern" } })
```

Trong đó: pattern: là chuỗi hoặc biểu thức chính quy muốn tìm kiếm.

Biểu thức chính quy hỗ trợ các ký tự đặc biệt như ^, $, *, v.v., cho phép tìm kiếm chuỗi ở đầu, cuối hoặc bất kỳ vị trí nào trong chuỗi.
Ví dụ: Tìm tài liệu trong đó trường name chứa chuỗi “bc”:

```
db.students.find({"name": {"$regex": "bc"}})
```

Tìm chuỗi kết thúc với mẫu:

```
db.students.find({"name": {"$regex": "bc$"}})
```

Tìm kiếm chuỗi chứa mẫu ở bất kì vị trí nào

Ví dụ: Tìm tất cả người dùng có tên chứa “li” ở bất kì đâu trong chuỗi

```
db.students.find({"name": {"$regex": "li"}})
```

Tìm kiếm không phân biệt chữ hoa chữ thường

MongoDB cho phép thực hiện tìm kiếm không phân biệt chữ hoa chữ thường bằng cách sử dụng tùy chọn "i" trong biểu thức chính quy.

Ví dụ: tất cả các tài liệu trong bộ sưu tập students có trường name là "john", "John", "JOHN", hoặc bất kỳ dạng kết hợp chữ hoa chữ thường nào của "john".

```
db.students.find({"name": {"$regex": "john", "$options": "i"}})
```

Truy vấn với mảng

MongoDB hỗ trợ việc lưu trữ và truy vấn các mảng (arrays) trong tài liệu (documents). Việc truy vấn mảng có thể được thực hiện bằng cách sử dụng các toán tử như $in, $all để lọc và tìm kiếm các phần tử trong mảng.

- Toán tử $in để tìm kiếm các tài liệu mà một trường nào đó có giá trị bằng một trong các giá trị trong một mảng. Điều này có thể được sử dụng khi muốn tìm các bản ghi có trường đó khớp với một danh sách các giá trị cụ thể.

Ví dụ: Tìm kiếm tất cả các tài liệu có trường age là 25, 30, hoặc 35.

```
db.students.find({ "age": { "$in": [25, 30, 35] } })
```

- Toán tử $all được sử dụng để tìm kiếm các tài liệu có trường mảng chứa tất cả các phần tử được chỉ định trong mảng.

Ví dụ: Tất cả các môn học mà trường subject chứa cả "Mathematics" và "Physics".

```
db.students.find({ "subject": { "$all": ["Mathematics", "Physics"] } })
```

2.2.3. Cập nhật dữ liệu

Cập nhật dữ liệu trong MongoDB được thực hiện thông qua các phương thức như ***updateOne()***, ***updateMany()***, và ***replaceOne()***. Những phương thức này cho phép thay đổi dữ liệu của các tài liệu theo những điều kiện nhất định.

Phương thức ***updateOne()*** được sử dụng để cập nhật một tài liệu duy nhất trong cơ sở dữ liệu. Phương thức này nhận vào hai tham số chính: điều kiện tìm kiếm (filter) và các thay đổi cần thực hiện (update).

Cú pháp:

```
db.collection.updateOne(
   <filter>,
   <update>,
   { options }
)
```

- filter: Điều kiện để xác định tài liệu cần cập nhật.
- update: Các thay đổi cần thực hiện.
- options: Các tùy chọn bổ sung (tùy chọn này là không bắt buộc).

Ví dụ: Cập nhật tài liệu có name là Jane Smith và thay đổi trường age thành 33.

```
db.students.updateOne(
   {"name": "Jane Smith"},
   {"$set": {"age": 33}}
)
```

Phương thức ***updateMany()*** cho phép cập nhật nhiều tài liệu trong cùng một lúc. Nó hoạt động tương tự như ***updateOne()***, nhưng có thể thay đổi tất cả các tài liệu khớp với điều kiện tìm kiếm.

Cú pháp:

```
db.collection.updateMany(
   <filter>,
   <update>,
   { options }
)
```

Ví dụ: Cập nhật tất cả tài liệu có age lớn hơn hoặc bằng 25 và thay đổi trường age thành 35.

```
db.students.updateMany(
   {"age": {"$gt": 30}},
```

```
    {"$set": {"age": 35}}
)
```

Phương thức ***replaceOne()*** thay thế toàn bộ tài liệu khớp với điều kiện tìm kiếm bằng một tài liệu mới. Phương thức này khác với ***updateOne()*** ở chỗ nó thay thế toàn bộ tài liệu, không chỉ một số trường cụ thể.

Cú pháp:

```
db.collection.replaceOne(
   <filter>,
   <new_document>,
   { options }
)
```

Ví dụ:

```
db.students.replaceOne(
   {"name": "Jane Smith"},
   {
      "name": "Jane Smith",
      "age": 33,
      "subject": ["Physics", "Mathematics"],
      "email": "jane.smith.new@example.com"
   },
   upsert=True  )
```

2.2.4. Xóa dữ liệu

Để xóa dữ liệu trong MongoDB, ta sử dụng các phương thức ***deleteOne()*** và ***deleteMany()***. Các phương thức này cho phép xóa một hoặc nhiều tài liệu trong bộ sưu tập.

- **deleteOne()**: Xóa một tài liệu đầu tiên khớp với điều kiện tìm kiếm.
- **deleteMany()**: Xóa tất cả các tài liệu khớp với điều kiện tìm kiếm.

Ví dụ: Xóa một tài liệu đầu tiên có trường name là "Jane Smith"

```
db.students.deleteOne({"name": "Jane Smith"})
```

Ví dụ: Xóa tất cả tài liệu trong collection students có trường "age" có giá trị nhỏ hơn 28

```
db.students.deleteMany({"age": {"$lt": 28}})
```

2.3 Aggregation framework

2.3.1 Khái niệm

Aggregation là một **cơ chế xử lý dữ liệu mạnh mẽ** trong MongoDB, cho phép thực hiện các phép biến đổi và tổng hợp phức tạp trên dữ liệu. Thay vì chỉ trả về các tài liệu riêng lẻ như truy vấn find() aggregation giúp người dùng **phân tích, thống kê, nhóm dữ liệu và tính toán** dựa trên nhiều tiêu chí.

Aggregation được triển khai dưới dạng **pipeline**, tức là một chuỗi các bước xử lý (stage) mà dữ liệu sẽ lần lượt đi qua. Mỗi stage thực hiện một tác vụ cụ thể và trả về dữ liệu cho stage tiếp theo.

2.3.2 Cấu trúc của một Aggregation Pipeline

```
db.students.aggregate([
  { $match: { class: "10A" } },
  { $group: { _id: "$subject", avgScore: { $avg: "$score" } } },
  { $sort: { avgScore: -1 } }
])
```

Hình 7: Minh hoạ cấu trúc của Aggregation Pipeline

(Đoạn mã được thực thi thông qua MongoDB Shell hoặc từ công cụ code nào đó)

Trong ví dụ trên:

1. $match: Lọc học sinh thuộc lớp 10A.

2. $group: Gom nhóm theo môn học và tính điểm trung bình.
3. $sort: Sắp xếp kết quả theo điểm trung bình giảm dần.

2.3.3. Các Stage thường dùng trong Aggregation

$match	WHERE	Lọc các tài liệu theo điều kiện
$project	SELECT	Chọn trường cần hiển thị, có thể đổi tên, tính toán trường mới
$group	GROUP BY , SUM, AVG	Gom nhóm và tính toán các chỉ số tổng hợp
$sort	ORDER BY	Sắp xếp kết quả tăng/giảm dần
$limit	LIMIT	Giới hạn số lượng kết quả
$skip	OFFSET	Bỏ qua một số dòng đầu (thường dùng phân trang)
$unwind	Không có tương đương trực tiếp	Tách từng phần tử trong mảng thành tài liệu riêng
$lookup	JOIN	Kết nối hai collection (giống JOIN giữa hai bảng trong SQL)

III. Chỉ mục (Index)

3.1. Khái niệm

Chỉ mục (Index) là các cấu trúc dữ liệu đặc biệt, lưu giữ một phần nhỏ của tập hợp dữ liệu, hỗ trợ việc truy vấn hiệu quả hơn. Index trong Database cũng giống như mục lục của một cuốn sách. Thay vì tìm từng trang của cuốn sách, Database tạo một mục lục, và nó chỉ việc tìm nội dung của cuốn sách qua mục lục đó. Qua đó giúp cho câu lệnh truy vấn nhanh hơn. Một câu truy vấn không có Index được gọi là Table Scan. Nghĩa là Database phải xem qua toàn bộ các Document để tìm được kết quả truy vấn, và đối với các collection lớn, câu truy vấn sẽ rất chậm vì MongoDB phải xử lý một số lượng lớn dữ liệu.

Vai trò của chỉ mục:

- Giúp truy vấn dữ liệu nhanh hơn bằng cách tránh việc quét toàn bộ collection (collection scan).
- Tăng tốc độ tìm kiếm, sắp xếp, và lọc dữ liệu.
- Hỗ trợ các hoạt động như tìm kiếm bằng một hoặc nhiều trường (fields), sắp xếp dữ liệu (sort), và truy vấn phạm vi (range query).

3.2. Cách thức hoạt động

Khi bạn tạo chỉ mục trên một hoặc nhiều trường của một collection, MongoDB sẽ:

- Tạo cấu trúc dữ liệu chỉ mục:
  + Tạo cấu trúc dữ liệu chỉ mục (thường là dưới dạng cây B-tree).
  + Mỗi nút của cây hay chỉ mục lưu trữ: các giá trị từ trường được chỉ định (key), cùng với con trỏ đến vị trí vật lý của document trong collection.
- Sử dụng chỉ mục trong truy vấn:
  + Khi chạy một truy vấn, MongoDB kiểm tra xem có chỉ mục phù hợp hay không.
  + Nếu có, chỉ mục được sử dụng để tìm địa chỉ vật lý của các document cần thiết, thay vì quét toàn bộ collection.
- Ánh xạ giá trị đến Document:
  + Tìm key trong cây chỉ mục
  + Lấy địa chỉ vật lý của document tương ứng
  + Truy cập trực tiếp document đó

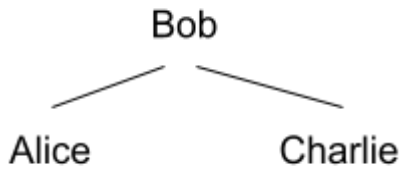
VD: Giả sử ta có collection về tên tuổi của học sinh

```
db.students.insertMany([
{ _id: 1, name: "Alice", age: 24 },
{ _id: 2, name: "Bob", age: 27 },
{ _id: 3, name: "Charlie", age: 22 } ])
```

- Khi ta tạo chỉ mục trên trường ‘name’:

```
db.students.createIndex({ name: 1 })
```

- MongoDB sẽ xây dựng cây B-Tree như sau:



- Mỗi nút chứa một giá trị trong trường ‘name’ và trỏ tới document tương ứng.

3.3. Đặc điểm của chỉ mục trong MongoDB

3.3.1. Đặc điểm

- Một Collection không thể có nhiều hơn 64 chỉ mục.
- Độ dài của tên chỉ mục không thể dài hơn 125 ký tự.
- Một chỉ mục phức hợp có thể có tối đa là 31 trường được lập chỉ mục.
- Chỉ mục lưu giữ giá trị của một trường cụ thể hoặc tập hợp các trường, được sắp xếp bởi giá trị của trường như đã được xác định.
- Các giá trị của chỉ mục không được trùng lặp.
- Chỉ mục ẩn: ẩn chỉ mục khi không cần dùng đến nhưng không thực sự xóa chỉ mục. Điều này giúp người lập trình dự đoán hiệu quả làm việc khi có hoặc không có chỉ mục đó.

Ưu điểm: Chỉ mục giúp tăng tốc độ truy vấn và hiệu năng.

Nhược điểm: Mỗi chỉ mục không chỉ chiếm dụng một phần không gian bộ nhớ mà còn tốn bộ nhớ mỗi khi thực hiện thêm, sửa và xóa dữ liệu trong database. Vì thế, nếu hiếm khi sử dụng Collection cho các hoạt động truy vấn dữ liệu, thì không nên sử dụng chỉ mục.

Sử dụng chỉ mục trong các trường hợp sau:

- Truy vấn thường xuyên trên một hoặc nhiều trường cụ thể.
- Dữ liệu lớn và việc quét toàn bộ collection trở nên chậm chạp.
- Truy vấn cần sắp xếp hoặc lọc nhiều dữ liệu.

3.3.2. Phân loại

Chỉ mục trong MongoDB có nhiều loại, nhưng phổ biến nhất là chỉ mục trường đơn và chỉ mục phức.

- Chỉ mục đơn: Đây là loại chỉ mục đơn giản nhất, được tạo trên một trường duy nhất trong một document.

- Chỉ mục phức: chỉ mục gồm nhiều trường.

Ngoài ra, có một số loại chỉ mục khác như:

- Multikey index: dùng cho dữ liệu được lưu trong mảng. MongoDB tạo chỉ mục cho từng phần tử trong mảng.

- Geospatial index: dùng trong hệ tọa độ 2 chiều.

- Text Indexes: tìm kiếm dữ liệu kiểu chuỗi.

- Hash index: chỉ mục băm dùng trong sharding.

3.3.3. Các bước thực hiện với chỉ mục

- Tạo một chỉ mục

```
db.COLLECTION_NAME.createIndex({KEY:1})
```

Ở đây, key là tên của trường mà bạn muốn tạo chỉ mục. Số 1 là cho thứ tự tăng dần, -1 là giảm dần. Ngoài ra, index có một số tùy chọn khác như background, unique, name,...

- Xóa chỉ mục

```
db.COLLECTION_NAME.dropIndex({KEY:1})  # Xoá một chỉ mục
```

```
db.COLLECTION_NAME.dropIndexes()   # Xóa tất cả index (trừ _id)
```

- Xem các chỉ mục hiện có:

```
db.COLLECTION_NAME.getIndexes()
```

- Truy vấn bằng index:

```
db.users.find({username: 'user112'}) // Tìm user có username là ‘user112’
db.users.createIndex({age: 1, gender: 1}) // Tạo index trên trường age và gender
db.users.find({gender: 'female', age: 29}) // Tìm user có giới tính nữ, 29 tuổi
```

IV. So sánh hiệu năng của MongoDB với MySQL

4.1. So sánh hiệu năng hai cơ sở dữ liệu

4.1.1. Hiệu năng truy vấn

- MongoDB: Hỗ trợ truy vấn nhanh hơn khi làm việc với dữ liệu phi cấu trúc hoặc dữ liệu lớn nhờ sharding và caching tích hợp nên MongoDB nhanh hơn khi truy vấn các tập dữ liệu lớn hoặc không đồng nhất.
- MySQL: Hiệu năng cao nếu dữ liệu đã được tối ưu hóa và đánh chỉ mục tốt. Tuy nhiên, có thể bị chậm khi xử lý dữ liệu phức tạp hoặc phi cấu trúc. MySQL nhanh hơn với các truy vấn phức tạp đòi hỏi mối quan hệ giữa các bảng (JOIN).

4.1.2.Hiệu suất ghi dữ liệu

- MongoDB: Xử lý ghi dữ liệu nhanh hơn nhờ mô hình không có giao dịch chặt chẽ trong các trường hợp không cần đảm bảo tính nhất quán cao và phù hợp với các hệ thống ghi dữ liệu theo thời gian thực
- MySQL:Hiệu quả hơn khi cần quản lý giao dịch tài chính hoặc dữ liệu quan trọng. Nhược điểm là ghi dữ liệu chậm hơn do phải đảm bảo tính toàn vẹn và nhất quán ACID trong các giao dịch.
    + Tính nguyên tử (*Atomicity*). Một giao dịch có nhiều thao tác khác biệt thì hoặc là toàn bộ các thao tác hoặc là không một thao tác nào được hoàn thành. Chẳng hạn việc chuyển tiền có thể thành công hay trục trặc vì nhiều lý do nhưng tính nguyên tử bảo đảm rằng một tài khoản sẽ không bị trừ tiền nếu như tài khoản kia chưa được cộng số tiền tương ứng.
    + Tính nhất quán (*Consistency*). Một giao dịch hoặc là sẽ tạo ra một trạng thái mới và hợp lệ cho dữ liệu, hoặc trong trường hợp có lỗi sẽ chuyển toàn bộ dữ liệu về trạng thái trước khi thực thi giao dịch.
    + Tính cô lập (*Isolation*). Một giao dịch đang thực thi và chưa được xác nhận phải bảo đảm tách biệt khỏi các giao dịch khác.
    + Tính bền vững (*Durability*). Dữ liệu được xác nhận sẽ được hệ thống lưu lại sao cho ngay cả trong trường hợp hỏng hóc hoặc có lỗi hệ thống, dữ liệu vẫn đảm bảo trong trạng thái chuẩn xác.

4.1.3. Hiệu suất đọc dữ liệu

MongoDB: Đọc nhanh hơn với các trường hợp không cần xử lý mối quan hệ phức tạp, đặc biệt khi dữ liệu đã được tối ưu hóa với các index.

MySQL: Đọc tốt hơn với các truy vấn phức tạp liên quan đến nhiều bảng (JOIN, GROUP BY, ORDER BY). Với hiệu năng cao nếu dữ liệu được chuẩn hóa và đánh chỉ mục hợp lý.

V. Kết luận

MongoDB là một cơ sở dữ liệu không quan hệ hướng tài liệu mã nguồn mở. Với cấu trúc linh hoạt , Hệ quản trị cơ sở dữ liệu này đem lại sự thuận tiện trong việc truy xuất các dữ liệu với tốc độ rất nhanh. Được thiết kế với giao diện thân thiện, dễ sử dụng, cấu trúc các câu lệnh rõ ràng , dễ hiểu gần với ngôn ngữ đời thường.

Khác với các hệ quản trị cơ sở dữ liệu quan hệ truyền thống, MongoDB phù hợp với dữ liệu có cấu trúc không cố định, dễ dàng mở rộng quy mô và hỗ trợ tốt cho các ứng dụng thời gian thực. Nhờ những ưu điểm này, MongoDB được ứng dụng rộng rãi trong nhiều lĩnh vực, từ phát

triển ứng dụng web, phân tích dữ liệu lớn đến IoT và các ứng dụng di động. Các tính năng nổi bật của MongoDB bao gồm hỗ trợ nhiều loại dữ liệu, khả năng phân tán dữ liệu, đảm bảo tính sẵn sàng cao và dễ dàng sao lưu, phục hồi dữ liệu.

Mặc dù còn một số hạn chế, MongoDB đang ngày càng khẳng định vị thế là một trong những hệ quản trị cơ sở dữ liệu NoSQL phổ biến nhất hiện nay và có tiềm năng phát triển lớn trong tương lai, đặc biệt khi các mạng xã hội và thương mại điện tử tiếp tục phát triển.

VI. Ứng dụng thực tiễn

Để ứng dụng những nội dung đã được học, nhóm đã tận dụng khả năng lưu trữ thông tin và truy vấn nhanh của MongoDB để làm Cơ sở Dữ liệu cho hệ thống thu thập thông tin báo chí từ nhiều domain báo chí Việt Nam.

6.1 Cấu trúc hệ thống

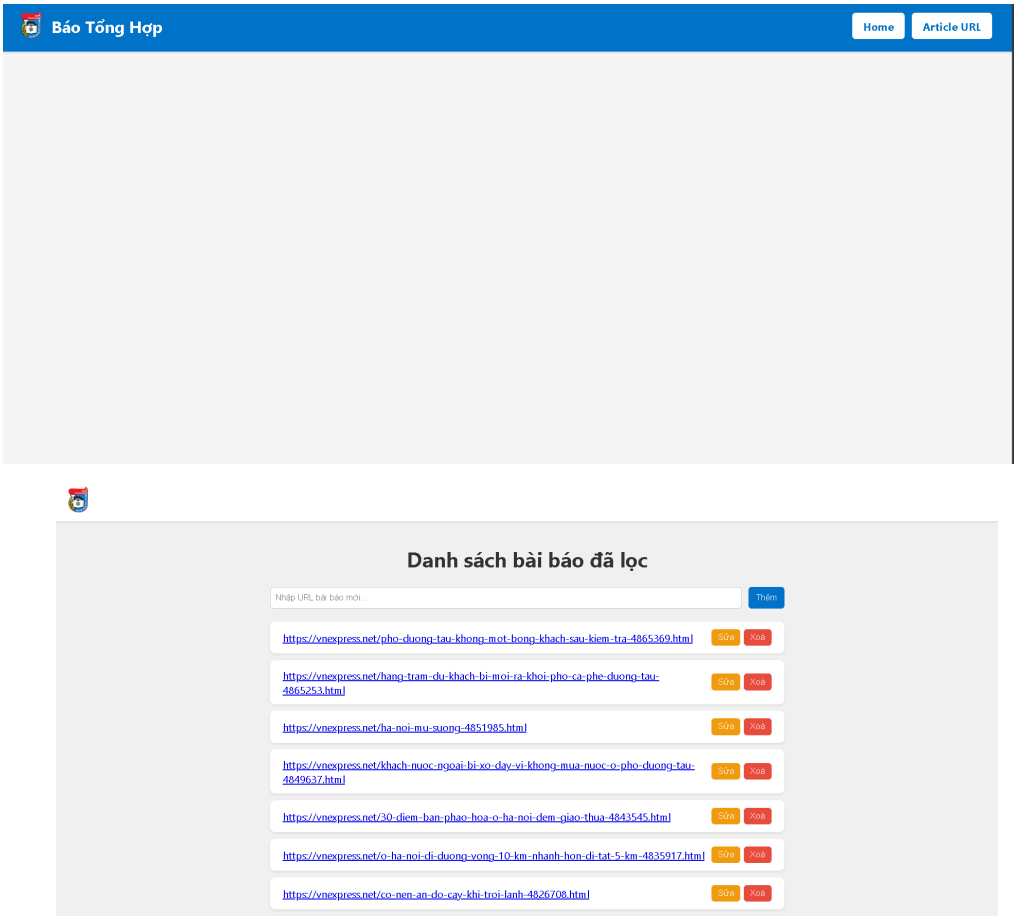
```
|   Dockerfile
|   filter_articles.py
|   README.md
|   recursive_crawler.py
|   requirements.txt
|
+---back-end
|   |   main.py
|   |
|   \---__pycache__
|           main.cpython-311.pyc
|
+---front-end
|   |   articles.html
|   |   home.html
|   |
|   \---assets
|       |   logo-hus-inkythuatso-inkythuatso.rar
|       |   logo.png
```

Hình 8: Cấu trúc dự án.

Nhằm đóng gói hệ thống một cách hoàn chỉnh, nhóm đã sử dụng Docker để chạy MongoDB. Ngoài ra trong dự án này nhóm sẽ thực hiện hai công việc chính. Đầu tiên sẽ thu thập các url có chứa domain trang tin bằng chiến thuật bò liếm. Công việc thứ hai cần làm là lọc ra các url báo chí từ các url đã thu thập được.

Ngoài việc thu thập thông tin và xử lý dữ liệu. Nhóm cũng tận dụng framework có sẵn của Python nhằm triển khai một web service đơn giản sử dụng html, fastapi, … Để tạo ra một trang web đơn giản giúp hiển thị, thêm, sửa, xóa các url báo chí đã thu thập được. Sau đây là mã nguồn của dự án:hausura/news-crawler-system: crawl data from many VN article domains .

6.2. Giao diện



Hình 9: Giao diện web nhóm tạo.

TÀI LIỆU THAM KHẢO

1. MongoDB, Inc. (2025). MongoDB Manual Documentation. Accessed April 10, 2025
2. Kristina Chodorow, Michael Dirolf . (2019). MongoDB: The Definitive Guide: Powerful and Scalable Data Storage. O'Reilly Media, Inc.
3. Wikipedia contributors. (n.d.). *ACID*. Wikipedia. Retrieved April 05, 2025
4. "MongoDB vs MySQL: A Detailed Comparison." *Integrate.io*, n.d. Accessed April 09, 2025.