Nhom3 Tìm Hieu Mongo DB

Nhập môn công nghệ phần mềm (Trường Đại học Công nghệ thông tin, Đại học Quốc gia Thành phố Hồ Chí Minh)

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA KHOA HỌC VÀ KỸ THUẬT THÔNG TIN

BÁO CÁO ĐỒ ÁN
TÌM HIỂU HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU MONGODB

Sinh viên thực hiện:
Nguyễn Trí Vũ - 20521042
Nguyễn Thanh Tuấn - 20520031
Hồ Cảnh Công - 20520006

Giảng viên:
ThS. Lưu Thanh Sơn

Thành phố Hồ Chí Minh, tháng 11 năm 2022

BÁO CÁO TÓM TẮT

1. Tiêu đề báo cáo: TÌM HIỂU HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU MONGODB

2. Danh sách thành viên

MSSV	Họ tên	Ghi chú
20521042	Nguyễn Trí Vũ	
20520031	Nguyễn Thanh Tuấn	
20520006	Hồ Cảnh Công	

3. Nội dung chi tiết

Nội dung 1: Giới thiệu về MongoDB

+ Đề mục 1: Giới thiệu sơ lược về MongoDB

+ Đề mục 2: Một số khái niệm cơ bản trong MongoDB

Nội dung 2: Các tính năng cơ bản

+ Đề mục 1: Các mô hình dữ liệu

+ Đề mục 2: Các phương thức CRUD trong MongoDB

+ Đề mục 3: Aggregation Operation

Nội dung 3: Các tính năng nâng cao

+ Đề mục 1: Phân quyền, xác thực, bảo mật trong MongoDB

+ Đề mục 2: Index

+ Đề mục 3: Replica Set

+ Đề mục 4: Phân tán dữ liệu

+ Đề mục 5: Sao lưu và khôi phục

+ Đề mục 6: Nhập xuất dữ liệu

+ Đề mục 7: Một số tính năng khác trên MongoDB

Nội dung 4: Demo tính năng trên MongoDB

+ Đề mục 1: Kịch bản demo

+ Đề mục 2: Các chức năng sẽ demo

+ Đề mục 3: Kết quả demo

Nội dung 5: Kết luận và hướng phát triển

4. Phân công công việc

MSSV	Họ tên	Nội dung được phân công
20521042	Nguyễn Trí Vũ	Nội dung 1 mục 1.2, Nội dung 3 mục 3.1 đến 3.4, mục 3.7, Nội dung 4, Nội dung 5. Tổng hợp nội dung và quay video
20520031	Nguyễn Thanh Tuấn	Nội dung 2 (tất cả) , Nội dung 4
20520006	Hồ Cảnh Công	Nội dung 1 mục 1.1, Nội dung 3 mục 3.5, 3.6, Nội dung 4

MỤC LỤC

DANH MỤC BẢNG

DANH MỤC HÌNH VẼ

CHƯƠNG 1: GIỚI THIỆU VỀ MONGODB

1.1. Giới thiệu sơ lược về hệ quản trị CSDL

MongoDB là một trong những cơ sở dữ liệu NoSQL phổ biến nhất, là một cơ sở dữ liệu hướng tài liệu có mã nguồn mở. Thuộc loại cơ sở dữ liệu không quan hệ (NoSQL), MongoDB không dựa trên cấu trúc cơ sở dữ liệu quan hệ giống như bảng mà sẽ lưu trữ và truy xuất dữ liệu ở dạng document. Định dạng dữ liệu mới mà MongoDB sử dụng là BSON (khá gần giống với định dạng JSON).

MongoDB được phát hành vào năm 2009 được viết bằng ngôn ngữ C++, C++ là ngôn ngữ gần với ngôn ngữ máy nên dễ dàng hiểu rằng MongoDB có thể tính toán ở tốc độ cao hơn hẳn các hệ quản trị cơ sở dữ liệu khác. Đó cũng là lý do mà MongoDB luôn được các nhà phát triển đánh giá rất cao.

MongoDB hỗ trợ đa nền tảng theo hướng đối tượng dùng lưu trữ dữ liệu có cấu trúc phức tạp. Không như các hệ quản trị cơ sở dữ liệu khác lưu trữ ở dạng bảng, MongoDB lưu dữ liệu vào collection theo hướng tài liệu JSON. Đây là kiểu dữ liệu dạng Key-Value truy xuất nhanh và khả năng mở rộng không bị ràng buộc mới tạo khóa ngoại hay khóa chính.

1.2. Một số khái niệm cơ bản trong MongoDB

Các thành phần cốt lõi tạo nên MongoDB gồm:

- Mongod: tiến trình lõi của MongoDB, nơi lưu trữ và quản lý dữ liệu
- Mongos: controller và query router, dùng cho trường hợp phân tán dữ liệu (sharded cluster)
- Mongosh: giao diện shell để người dùng có thể tương tác thực hiện lệnh với cơ sở dữ liệu. Có thể dùng Mongosh để chạy kiểm tra các lệnh truy vấn và thao tác tùy biến dữ liệu lưu trong cơ sở dữ liệu

Để khởi động MongoDB mặc định trên localhost, chạy lệnh *mongosh* trên command line. Tiến trình mongod khởi động, và mongosh sẽ kết nới với mongod và chạy trên localhost:27017. Lệnh *mongosh* tương đương với câu lệnh *mongosh "mongodb://localhost:27017"*

Document: Trong MongoDB, một bản ghi được gọi là một document. Mỗi document chứa một cặp giá trị gồm trường dữ liệu và giá trị tương ứng của trường đó trong document. Document về hình thức tương tự như dữ liệu kiểu JSON. Giá trị của

một trường có thể là số, văn bản, mảng, một document khác hoặc một mảng các document.

Collection: MongoDB lưu trữ các document trong collection. Collection trong MongoDB tương đương với bảng trong mô hình quan hệ. Để tạo một collection mới trong cơ sở dữ liệu, sử dụng phương thức *db.createCollection()*

BSON: là một định dạng dữ liệu tuần tự hóa dạng nhị phân được sử dụng để lưu trữ document và thực hiện các cuộc gọi thủ tục từ xa trong MongoDB. Có nhiều kiểu dữ liệu BSON, có thể được sử dụng số hoặc chuỗi định danh để quy định kiểu dữ liệu cho một đối tượng.

Database (Cơ sở dữ liệu): Một mongod có thể chứa nhiều cơ sở dữ liệu. Một cơ sở dữ liệu chứa một hoặc nhiều collection. Để chọn một cơ sở dữ liệu và tiến hành các thao tác với dữ liệu với cơ sở dữ liệu đó, trong mongosh dùng lệnh *use <tên cơ sở dữ liệu>*.

Để tạo một cơ sở dữ liệu mới trong mongod, trong trường hợp cơ sở dữ liệu không tồn tại, MongoDB sẽ tự động tạo cơ sở dữ liệu mới khi lưu trữ document lần đầu tiên cho cơ sở dữ liệu đó. Do đó, có thể dùng lệnh *use* để chuyển sang cơ sở dữ liệu không tồn tại và thực hiện thao tác thêm document mới trong collection mới trong mongosh. MongoDB sẽ tạo cơ sở dữ liệu mới kèm theo collection kèm document được chỉ định. Ví dụ:

```
use myNewDB
db.myNewCollection1.insertOne( { x: 1 } )
```

MongoDB Compass: là một giao diện GUI cho phép người dùng tương tác với mongod một cách trực quan thay vì dùng giao diện commandline. MongoDB Compass hiển thị các cơ sở dữ liệu, collection, document một cách trực quan, hỗ trợ truy vấn, thực hiện aggregating function và phân tích dữ liệu MongoDB.

MongoDB Database Tools: là tập hợp các tiện ích dạng command-line được thiết lập sẵn, dùng để thực hiện các thao tác đặc biệt với MongoDB như nhập xuất, sao lưu, khôi phục dữ liệu (mongodump và mongorestore),…

CHƯƠNG 2: CÁC TÍNH NĂNG CƠ BẢN

2.1. Các mô hình dữ liệu.

2.1.1. Giới thiệu về mô hình dữ liệu

Thách thức chính trong mô hình hóa dữ liệu là cân bằng giữa nhu cầu của ứng dụng, đặc điểm về hiệu suất của hệ quản trị cơ sở dữ liệu và các mẫu truy xuất dữ liệu. Khi thiết kế mô hình dữ liệu, cần phải luôn cân nhắc về việc sử dụng dữ liệu trong ứng dụng (bao gồm các truy vấn, cập nhật và xử lý dữ liệu) cũng như cấu trúc vốn có của chính dữ liệu đó.

2.1.2. Tính mềm dẻo

Khác với các cơ sở dữ liệu SQL, yêu cầu phải xác định và khai báo lược đồ của bảng trước khi thêm dữ liệu, các Collection của MongoDB không yêu cầu các document phải có cùng một mô hình dữ liệu: không nhất thiết phải có cùng một nhóm các trường, và kiểu dữ liệu cho một trường cũng có thể khác nhau giữa các document trong bộ dữ liệu. Để thay đổi cấu trúc của document trong Collection, chỉ cần cập nhật document theo cấu trúc dữ liệu mới.

Tính linh hoạt này tạo điều kiện thuận lợi cho việc ánh xạ các document tới một thực thể hoặc một đối tượng. Mỗi document có thể khớp với các trường dữ liệu của thực thể được đại diện, ngay cả khi bản ghi có sự thay đổi đáng kể so với các document khác trong Collection. Tuy nhiên trong thực tế, các document trong cùng một Collection thường có cấu trúc tương tự nhau. Đồng thời, người ta cũng sử dụng các quy tắc xác thực mô hình dữ liệu đối với quá trình thêm và sửa dữ liệu.

2.1.3. Các cấu trúc của mô hình dữ liệu

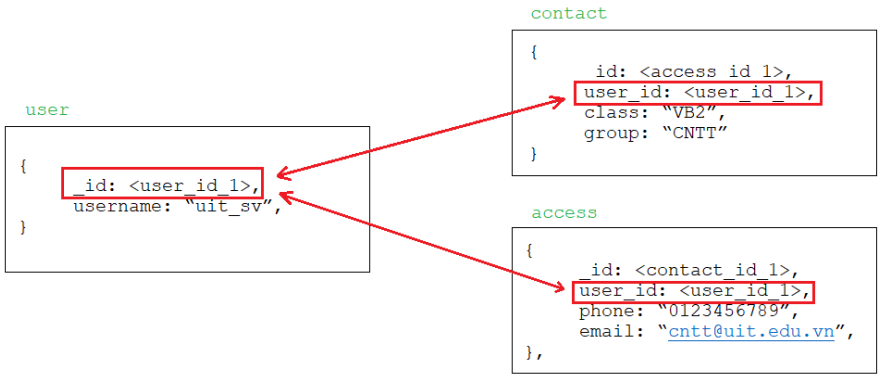
Vấn đề quan trọng trong việc thiết kế mô hình dữ liệu cho các ứng dụng sử dụng MongoDB xoay quanh cấu trúc của các Collection và cách ứng dụng thể hiện mối quan hệ giữa các dữ liệu.

- **Nhúng dữ liệu:** MongoDB cho phép nhúng tất cả dữ liệu liên quan trong một document duy nhất. Dữ liệu được nhúng thể hiện mối quan hệ giữa các dữ liệu bằng cách lưu trữ dữ liệu liên quan trong một cấu trúc tài liệu duy nhất. MongoDB hỗ trợ việc nhúng các cấu trúc dữ liệu vào một trường hoặc mảng trong document. Các mô hình dữ liệu không "chuẩn hóa" này cho phép

các ứng dụng truy xuất và thao tác dữ liệu liên quan trong một thao tác cơ sở dữ liệu duy nhất. Ví dụ:

```
{
        _id: <user_id_1>,
        username: "uit_sv",
        contact: {
                phone: "0123456789",
                email: "cntt@uit.edu.vn",
        },
        access: {
                class: "VB2",
                group: "CNTT"
        }
}
```

- **Tham chiếu:** Còn được gọi là kiến trúc chuẩn hóa của mô hình dữ liệu. Các tham chiếu lưu trữ các mối quan hệ giữa các dữ liệu bằng cách thêm các liên kết hoặc tham chiếu giữa các document. Các ứng dụng có thể thông qua các tham chiếu này để truy cập dữ liệu liên quan. Đây là một dạng mô hình dữ liệu chuẩn hóa. Ở ví dụ dưới đây, trường "user_id" trong bộ dữ liệu "contact" và "access" được dùng để tham chiếu đến user có liên quan.



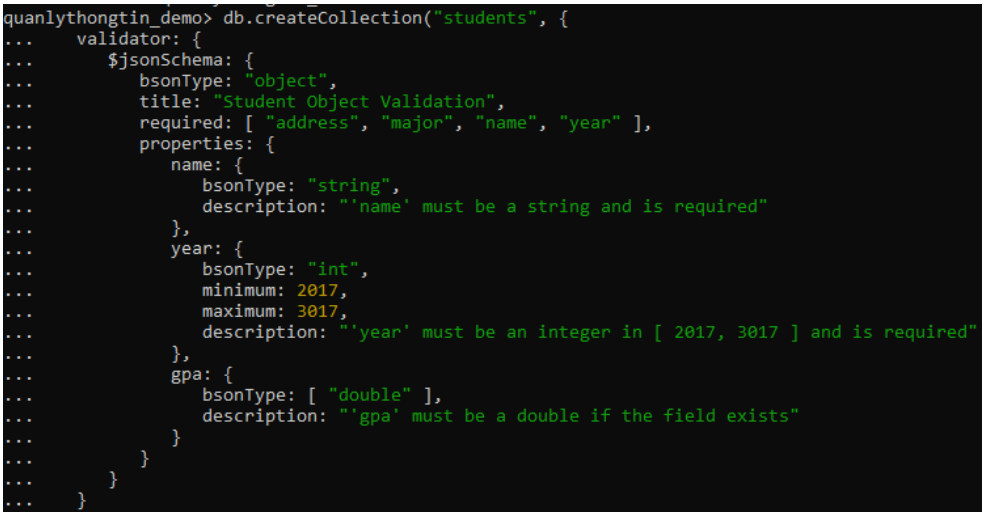
Hình 2.1: Ví dụ về tham chiếu trong mô hình dữ liệu của MongoDB

2.1.4. Xác thực mô hình dữ liệu

2.1.4.1. Thiết lập xác thực bằng JSON schema

Xác thực cấu trúc JSON là một phương pháp cho phép chú thích và xác thực các tài liệu JSON. Thông qua việc sử dụng một lược đồ JSON, để xây dựng các quy tắc xác thực cho các trường của document.

Ví dụ dưới đây sử dụng một toán tử **$jsonSchema** để thiết lập các quy tắc xác thực đối với việc thêm dữ liệu vào Collection "students".

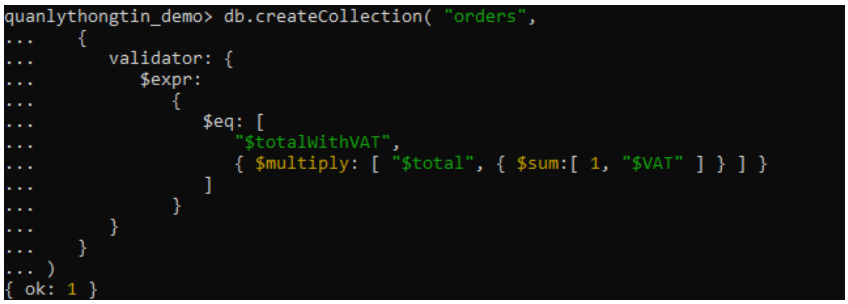


Hình 2.2: Ví dụ thiết lập xác thực bằng JSON schema

Bằng việc sử dụng định dạng JSON để thiết lập các quy tắc xác thực, chúng ta có thể dễ dàng đọc và hiểu được các quy tắc xác thực được đưa ra với các trường dữ liệu như: các trường nào là bắt buộc, kiểu dữ liệu của mỗi trường, giá trị min/max, các thông báo lỗi khi quy tắc xác thực bị vi phạm.

2.1.4.2. Thiết lập xác thực bằng toán tử truy vấn

Thiết lập quy tắc xác thực bằng toán tử truy vấn được sử dụng trong các trường hợp cần tạo ra các quy tắc xác thực động để so sánh giá trị của các trường trong thời gian chạy.



Hình 2.3: Ví dụ thiết lập xác thực bằng toán tử truy vấn

Lệnh ở trong ví dụ trên tạo ra một quy tắc xác thực khi thêm dữ liệu vào Collection “orders”: trường “totalWithVAT” phải bằng tổng của 2 trường “total” và “VAT”. Khi vi phạm sẽ nhận được thông báo lỗi.

2.1.5. Các mẫu thiết kế của mô hình dữ liệu

2.1.5.1. Mô hình quan hệ giữa các document

a) Quan hệ một – một

Xét ví dụ xây dựng sơ đồ mối quan hệ một – một giữa sinh viên và địa chỉ của sinh viên đó. Trong mô hình dữ liệu chuẩn hóa, document "DiaChi" sẽ chứa tham chiếu đến document "SinhVien". Nếu việc sử dụng ứng dụng cần thường xuyên truy xuất dữ liệu địa chỉ cùng với dữ liệu sinh viên, thì với việc dùng tham chiếu sẽ đòi hỏi đưa ra nhiều truy vấn hơn để giải quyết tham chiếu. Mô hình tốt hơn sẽ là nhúng dữ liệu "DiaChi" vào trong dữ liệu "SinhVien" như sau:

```
{
   _id: "sv_uit_1",
   name: "Nguyen Van A",
   address: {
      city: "HCM",
      district: "Thu Duc",
      ward: "Linh Trung"
   }
}
```

b) Quan hệ một – nhiều sử dụng nhúng dữ liệu

Mô hình dữ liệu quan hệ một – nhiều sử dụng phương pháp nhúng dữ liệu được sử dụng trong trường hợp cần truy xuất nhiều thực thể của một dữ liệu nằm trong một dữ liệu khác. Tiếp tục với ví dụ ở trên, ta xét trong trường hợp một sinh viên có thể có nhiều địa chỉ liên hệ. Nếu nhu cầu truy xuất thông tin địa chỉ đồng thời với dữ liệu sinh viên xảy ra thường xuyên, thì phương pháp tối ưu hơn là nhúng các thực thể của dữ liệu địa chỉ vào trong dữ liệu sinh viên.

```
{
   _id: "sv_uit_1",
   name: "Nguyen Van A",
   address: [
      {
         city: "HCM",
         district: "Thu Duc",
         ward: "Linh Trung"
      },
      {
         city: "HCM",
         district: "Phu Nhuan",
         ward: "Phuong 2"
      }
```

```
    ]
}
```

c) Quan hệ một – nhiều sử dụng tham chiếu

Xét ví dụ về mô hình dữ liệu thể hiện quan hệ giữa sinh viên và môn học đã đăng ký. Vì một môn học có rất nhiều sinh viên đăng ký, nên việc nhúng dữ liệu môn học vào trong dữ liệu sinh viên có thể dẫn đến việc lặp lại dữ liệu của môn học, gây lãng phí tài nguyên hệ thống. Để tránh việc lưu trữ lặp lại dữ liệu của môn học, có thể tạo ra một Collection riêng để chứa dữ liệu môn học. Cần lưu ý trong trường hợp này, một môn học sẽ có rất nhiều sinh viên đăng ký, nên việc để tham chiếu từ dữ liệu môn học sang dữ liệu sinh viên sẽ dẫn đến lưu trữ một mảng rất lớn:

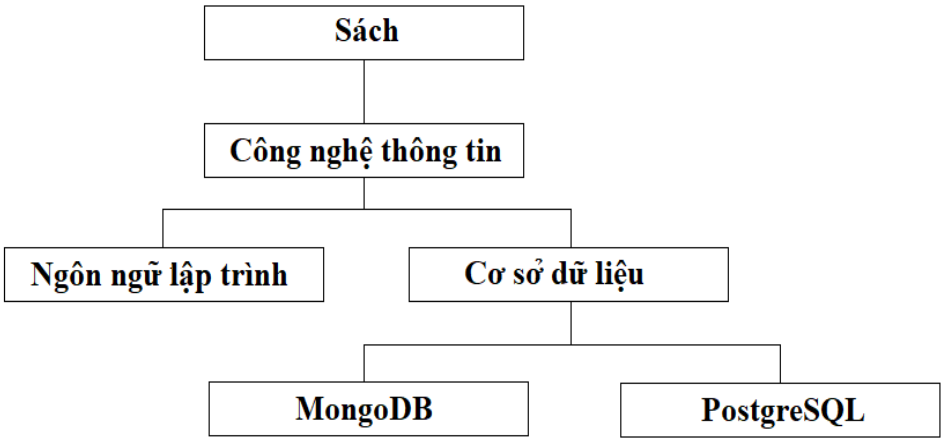
```
// MonHoc
{
   _id: "mon_hoc_1",
   name: "Quan ly thong tin",
   students: [“sv_uit_1”, “sv_uit_2”, “sv_uit_3” …]
}
```

Nên cân nhắc để tham chiếu đến dữ liệu môn học từ dữ liệu sinh viên như sau:

```
// MonHoc
{
   _id: "mon_hoc_1",
   name: "Quan ly thong tin",
}
// SinhVien
{
   _id: "sv_uit_1",
   name: "Nguyen Van A",
   class_id: “mon_hoc_1”
}
{
   _id: "sv_uit_2",
   name: "Nguyen Van B",
   class_id: “mon_hoc_1”
}
```

2.1.5.2. Mô hình cây

Xét ví dụ về hệ thống danh mục sách được mô tả trong cấu trúc cây như sau:



Hình 2.4: Ví dụ hệ thống danh mục sách được mô tả theo cấu trúc cây

a) Sử dụng tham chiếu cha

```
{ _id: "MongoDB", parent: "CoSoDuLieu" }
{ _id: "PostgreSQL", parent: "CoSoDuLieu" }
{ _id: "CoSoDuLieu", parent: "CongNgheThongTin" }
{ _id: "NgonNguLapTrinh", parent: "CongNgheThongTin" }
{ _id: "CongNgheThongTin", parent: "Sach" }
{ _id: "Sach", parent: null }
```

b) Sử dụng tham chiếu con

```
{ _id: "MongoDB", child: [] }
{ _id: "PostgreSQL", child: [] }
{ _id: "CoSoDuLieu", child: ["MongoDB", "PostgreSQL"] }
{ _id: "NgonNguLapTrinh", child: [] }
{ _id: "CongNgheThongTin", child: ["CoSoDuLieu", "NgonNguLapTrinh"] }
{ _id: "Sach", child: ["CongNgheThongTin"] }
```

c) Sử dụng danh sách gốc

```
{_id: "MongoDB", ancestors: ["Sach", "CongNgheThongTin", "CoSoDuLieu"],
parent: "CoSoDuLieu"}
{_id: "PostgreSQL", ancestors: ["Sach", "CongNgheThongTin","CoSoDuLieu"],
parent: "CoSoDuLieu"}
{_id: "CoSoDuLieu", ancestors: ["Sach","CongNgheThongTin"],
parent:"CongNgheThongTin"}
{_id:"NgonNguLapTrinh", ancestors: ["Sach","CongNgheThongTin"],
parent:"CongNgheThongTin"}
{_id:"CongNgheThongTin", ancestors: ["Sach"], parent: "Sach"}
{_id: "Sach", ancestors: [], parent: null}
```

d) Sử dụng đường dẫn

```
{_id: "MongoDB", path: ",Sach,CongNgheThongTin,CoSoDuLieu,"}
```

```
{_id: "PostgreSQL", path: ",Sach,CongNgheThongTin,CoSoDuLieu,"}
{_id: "CoSoDuLieu", path: ",Sach,CongNgheThongTin,"}
{_id:"NgonNguLapTrinh", path: ",Sach,CongNgheThongTin"}
{_id:"CongNgheThongTin", path: ",Sach,"}
{_id: "Sach", path: null}
```

2.2. Các phương thức CRUD trong MongoDB

2.2.1. Thêm dữ liệu

Cú pháp ***db.collection.insertOne()*** được sử dụng để thêm một document vào Collection.

```
quanlythongtin_demo> db.students.insertOne({ name: "uit_sv", year: Int32(2020), major: "CNTT", gpa: Double(3
{
  acknowledged: true,
  insertedId: ObjectId("634c01f7a6e48f56aa482685")
}
quanlythongtin_demo>
```

Hình 2.5: Dùng phương thức db.collection.insertOne() để thêm một sinh viên vào collection students

Câu lệnh trong ví dụ trên đã thêm thông tin của một sinh viên mới vào Collection "students". Vì trong document không khai báo trường "_id", MongoDB sẽ tự động gán một giá trị ObjectId cho trường này. Hàm ***insertOne()*** trả về kết quả thực thi thêm dữ liệu và giá trị của trường "_id" của document vừa được thêm vào.

Khi cần thêm nhiều document cùng một lúc, sử dụng cú pháp ***db.collection.insertMany()*** và truyền vào một mảng gồm các document.

```
quanlythongtin_demo> db.students.insertMany([
... { name: "uit_sv_1", year: Int32(2020), major: "CNTT", gpa: Double(3.3), address: { city: "HCM", street: "Duong so 1" } },
... { name: "uit_sv_2", year: Int32(2020), major: "CNTT", gpa: Double(3.7), address: { city: "HCM", street: "Duong so 2" } },
... { name: "uit_sv_3", year: Int32(2020), major: "CNTT", gpa: Double(3.6), address: { city: "HCM", street: "Duong so 3" } }
... ])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("634c0659a6e48f56aa482686"),
    '1': ObjectId("634c0659a6e48f56aa482687"),
    '2': ObjectId("634c0659a6e48f56aa482688")
  }
}
quanlythongtin_demo>
```

Hình 2.6: Dùng phương thức db.collection.insertMany() để thêm các sinh viên vào collection students

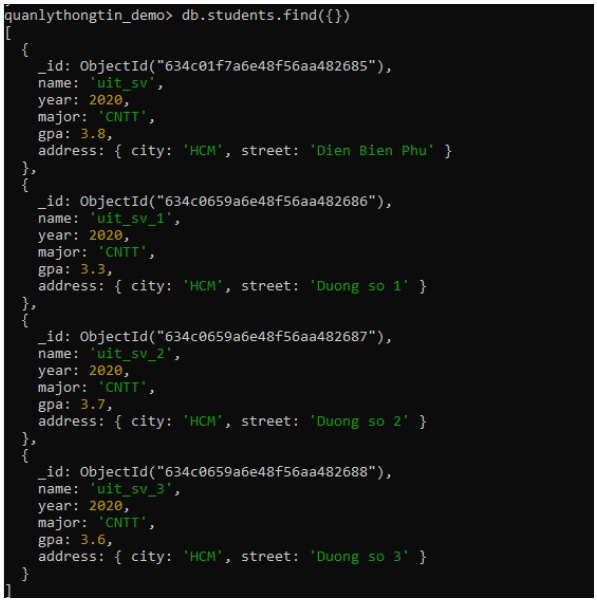
Khi sử dụng câu lệnh insert để thêm dữ liệu, nếu thêm dữ liệu vào một Collection không tồn tại, thao tác thêm dữ liệu sẽ tự động tạo mới Collection. MongoDB cũng yêu cầu mỗi document cần có trường "_id" làm khóa chính. Nếu trong câu lệnh insert chưa khai báo trường này, thao tác thêm dữ liệu sẽ tự động tạo ra một giá trị ObjectId để gán cho "_id", và trả về sau khi thêm dữ liệu thành công.

Ngoài ra, việc thêm dữ liệu cũng có thể được thực hiện bởi hàm ***db.collection.bulkWrite()*** hoặc một số hàm cập nhật, chỉnh sửa dữ liệu với tùy chọn *upsert=true:*

- *db.collection.updateOne()*
- *db.collection.updateMany()*
- *db.collection.findAndModify()*
- *db.collection.findOneAndUpdate()*
- *db.collection.findOneAndReplace()*

2.2.2. Truy vấn dữ liệu

Truy vấn tất cả document trong Collection*:* Sử dụng cú pháp ***db.collection.find()*** và truyền vào tham số là một đối tượng JSON rỗng {}.



Hình 2.7: Dùng phương thức db.collection.find() để lấy toàn bộ thông tin trong collection students

Sử dụng điều kiện bằng*:* Xác định các điều kiện tìm kiếm dưới dạng các cặp *<field>:<value>*, thể hiện dưới định dạng JSON và truyền vào hàm ***db.collection.find()***

Sử dụng toán tử truy vấn*:* MongoDB hỗ trợ tìm kiếm và lọc dữ liệu bằng các toán tử so sánh, logic, điều kiện, ...

Sử dụng điều kiện "AND" hoặc "OR: Ví dụ tìm kiếm tất cả sinh viên khoa *CNTT* có GPA > 3.7 hoặc < 3.6

```
quanlythongtin_demo> db.students.find({ major: "CNTT", $or: [{gpa: {$gt: 3.7}}, {gpa: {$lt: 3.6}}]})
[
  {
    _id: ObjectId("634c01f7a6e48f56aa482685"),
    name: 'uit_sv',
    year: 2020,
    major: 'CNTT',
    gpa: 3.8,
    address: { city: 'HCM', street: 'Dien Bien Phu' }
  },
  {
    _id: ObjectId("634c0659a6e48f56aa482686"),
    name: 'uit_sv_1',
    year: 2020,
    major: 'CNTT',
    gpa: 3.3,
    address: { city: 'HCM', street: 'Duong so 1' }
  }
]
```

Hình 2.8: Dùng phương thức db.collection.find() để tìm sinh viên thỏa điều kiện cho trước

2.2.3. Cập nhật dữ liệu

Cập nhật dữ liệu trên MongoDB được thực hiện thông qua các câu lệnh:

- db.collection.updateOne(<filter>, <update>, <options>)
- db.collection.updateMany(<filter>, <update>, <options>)
- db.collection.replaceOne(<filter>, <update>, <options>)

Trong đó, *<filter>* là điều kiện lọc document để thực hiện cập nhật, *<update>* là cú pháp cập nhật dữ liệu, với *<options>* là một số điều kiện được thêm vào.

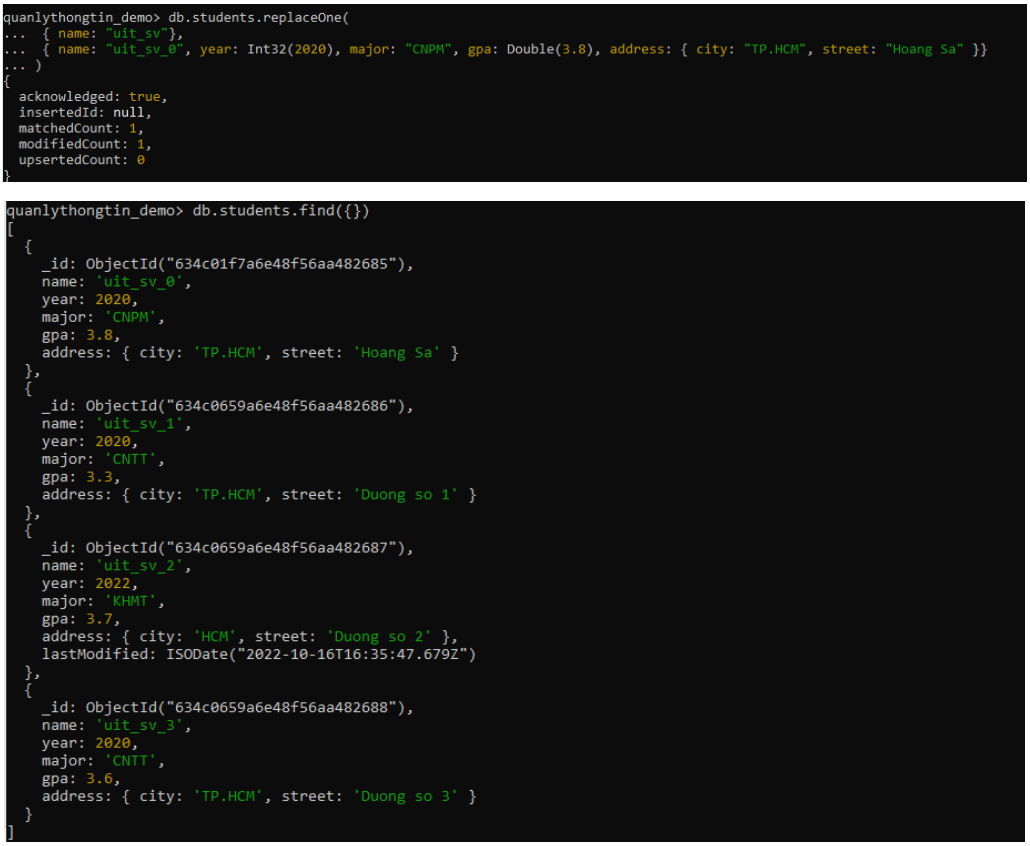
Cập nhật trên một document: Sử dụng cú pháp ***db.collection.updateOne()*** để lấy ra sinh viên đầu tiên có tên *"uit_sv_2"*, cập nhật chuyên ngành thành *"KHMT"* và niên khóa thành 2022. Dùng toán tử *$currentDate* để cập nhật trường *lastModified* thành ngày hiện tại; nếu chưa có sẽ được toán tử *$currentDate* tạo ra.

```
quanlythongtin_demo> db.students.updateOne(
... {name: "uit_sv_2"},
... {
...   $set: {major: "KHMT", year: 2022},
...   $currentDate: { lastModified: true }
... }
... )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
quanlythongtin_demo> db.students.find({name: "uit_sv_2"})
[
  {
    _id: ObjectId("634c0659a6e48f56aa482687"),
    name: 'uit_sv_2',
    year: 2022,
    major: 'KHMT',
    gpa: 3.7,
    address: { city: 'HCM', street: 'Duong so 2' },
    lastModified: ISODate("2022-10-16T16:35:47.679Z")
  }
]
```

Cập nhật trên nhiều document: Sử dụng ***db.collection.updateMany()*** để sửa địa chỉ thành phố của các sinh viên khóa 2020 từ *"HCM"* thành *"TP.HCM"*.

Hình 2.9: Dùng db.collection.updateMany()cập nhật thông tin sinh viên, sau đó kiểm tra kết quả

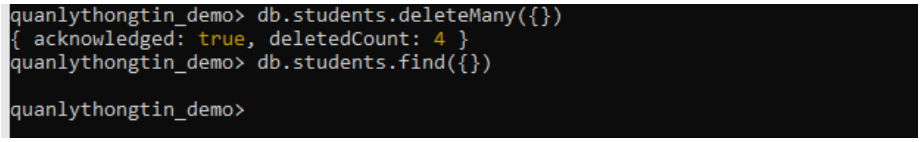
Thay thế giá trị trên một document: Sử dụng **db.collection.replaceOne()** để thay thế thông tin của sinh viên có tên "uit_sv".



Hình 2.10: Dùng db.collection. replaceOne() thay thế thông tin của sinh viên có tên "uit_sv"

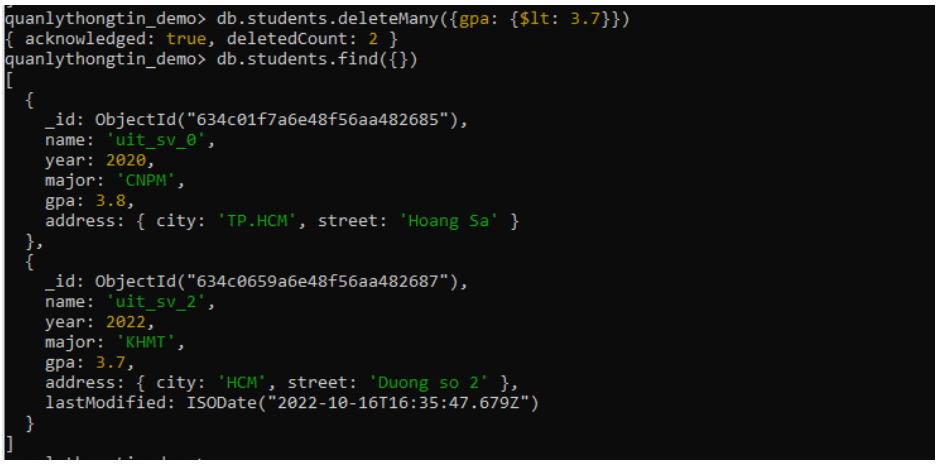
2.2.4. Xóa dữ liệu

Xóa tất cả document trong Collection: Sử dụng **db.collection.deleteMany(),** truyền vào tham số là một đối tượng JSON rỗng.



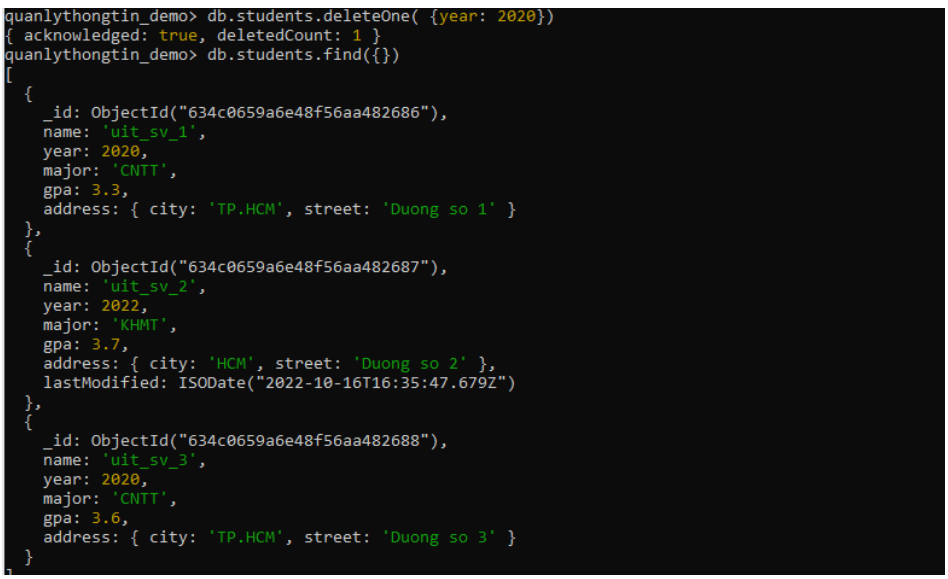
Hình 2.11: Dùng phương thức db.collection.deleteMany() xóa tất cả document trong Collection

Xóa tất cả document thỏa mãn điều kiện**:* Sử dụng ***db.collection.deleteMany(), với tham số truyền vào thể hiện điều kiện để tìm kiếm document thực thi lệnh xóa. Ví dụ xóa dữ liệu các sinh viên có GPA < 3.7 ra khỏi Collection *students.*



Hình 2.12: Dùng phương thức db.collection.deleteMany() xóa các document trong Collection theo điều kiện

Xóa một document thỏa điều kiện**:* Sử dụng cú pháp ***db.collection.deleteOne(), với tham số truyền vào là điều kiện tìm kiếm, lệnh này sẽ xóa document đầu tiên tìm thấy thỏa mãn điều kiện đã cho. Ví dụ, xóa dữ liệu sinh viên đầu tiên niên khóa 2020.



Hình 2.13: Dùng phương thức db.collection.deleteOne() xóa document đầu tiên trong Collection thỏa điều kiện

2.3. Aggregation Operation

Aggregation Operations xử lý nhiều tài liệu và trả về kết quả đã tính toán. Bạn có thể sử dụng các phép toán tổng hợp để: Nhóm các giá trị từ nhiều document với nhau; Thực hiện các thao tác trên dữ liệu được nhóm để trả về một kết quả duy nhất; Phân tích sự thay đổi của dữ liệu theo thời gian

Aggregation Operations có thể được thực hiện phổ biến bằng các phương thức như Aggregation Pipeline, Map-Reduce

2.3.1. Aggregation Pipeline

Aggregation Pipeline bao gồm nhiều giai đoạn xử lý tài liệu:

- Mỗi giai đoạn thực hiện một thao tác trên các document đầu vào. Ví dụ: một giai đoạn có thể lọc document, nhóm document và tính toán giá trị.
- Các document được trả ra từ một giai đoạn xử lý sẽ được chuyển vào cho giai đoạn xử lý tiếp theo
- Một Aggregation Pipeline có thể trả về kết quả cho các nhóm tài liệu. Ví dụ: trả về giá trị tổng, trung bình, lớn nhất, nhỏ nhất.

Ví dụ về Aggregation Pipeline dưới đây gồm có hai giai đoạn và trả về tổng số lượng đặt hàng của các mặt hàng có cỡ *medium* và được nhóm theo tên mặt hàng:

```
db.orders.aggregate( [
// Giai đoạn 1: Lọc ra các đơn hàng có cỡ là "medium"
   {
      $match: { size: "medium" }
   },
// Giai đoạn 2: Từ các đơn hàng đã lọc, nhóm theo tên mặt hàng và tính tổng
   {
      $group: { _id: "$name", totalQuantity: { $sum: "$quantity" } }
   }] )
```

Ví dụ trên minh họa một Aggregation Pipeline với 2 giai đoạn:

- Giai đoạn *$match*: Lọc ra các document đơn hàng có kích cỡ là *medium,* sau đó *c*huyển các document vừa lọc được đến giai đoạn *$group*
- Giai đoạn *$group*: Nhóm các document nhận được theo tên mặt hàng. Sau đó sử dụng *$sum* để tính toán tổng đơn hàng từ *$quantity* của mỗi đơn hàng. Tổng vừa tính được lưu trữ trong *totalQuantity* và được trả về bởi Aggregation Pipeline.

2.3.2. Map-Reduce

Map-Reduce là một mô hình xử lý dữ liệu để nhóm và mô hình hóa khối lượng lớn dữ liệu thô thành các kết quả tổng hợp hữu ích. Để thực hiện các thao tác map-reduce, MongoDB cung cấp lệnh *mapReduce*.

```
db.orders.mapReduce(
   function() { emit( this.cust_id, this.amount ); },
   function(key, value) { return Array.sum( values) },
   {
      query: { status: "A" },
      out: "order_totals"
   }
)
```

Trong thao tác *map-reduce* ở ví dụ trên, MongoDB áp dụng giai đoạn *map* cho mỗi document đầu vào (nghĩa là các document trong Collection thỏa mãn điều kiện truy vấn). Quá trình *map* tạo ra ánh xạ các cặp khóa-giá trị (*key-value*). Đối với những khóa có nhiều giá trị, MongoDB áp dụng *reduce*, phương thức này thu thập và rút gọn dữ liệu tổng hợp. MongoDB sau đó lưu trữ các kết quả trong một Collection. Theo tùy chọn, đầu ra của quá trình *reduce* có thể chuyển qua một hàm cuối cùng để rút gọn thêm hoặc xử lý kết quả của Aggregation Operation.

Tất cả các hàm phục vụ *map-reduce* trong MongoDB đều dùng JavaScript và chạy trong tiến trình *mongod*. Thao tác *map-reduce* lấy các tài liệu của một Collection duy nhất làm đầu vào và có thể thực hiện bất kỳ thao tác phân loại và lọc nào trước khi bắt đầu giai đoạn *map*. *mapReduce* có thể trả về kết quả của thao tác dưới dạng document hoặc có thể ghi kết quả vào Collection.

CHƯƠNG 3: CÁC TÍNH NĂNG NÂNG CAO

3.1. Phân quyên, xác thực, bảo mật trong MongoDB

Để đảm bảo an toàn bảo mật cho cơ sở dữ liệu, MongoDB cung cấp các tính năng giúp người sử dụng thực hiện các biện pháp an ninh cần có như sau:

- Xác thực các kết nối của người dùng (Authentication)
- Phân quyền (Authorization)/ kiểm soát người dùng truy cập thông qua các role (Role-Based access control.)
- Mã hóa mạng (Network Encryption)/ mã hóa các thông tin được truyền đi (Transport Encryption)
- Mã hóa kho dữ liệu (Storage Encryption)/ Mã hóa dữ liệu lúc hệ thống đang ở trạng thái nghỉ (Encryption-at-rest)
- Kiểm tra (Auditing)

3.1.1. Xác thực các kết nối của người dùng (Authentication)

Tính năng xác thực yêu cầu máy khách cung cấp bằng chứng rằng tài khoản là hợp lệ để có thể truy cập cơ sở dữ liệu. MongoDB cung cấp 3 loại cơ chế xác thực:

- Xác thực dựa trên mật khẩu – máy khách xác minh danh tính bằng cách chứng minh việc có sở hữu một giá trị bí mật được xác định trước. Cơ chế này ở MongoDB sử dụng SCRAM-SHA-1 và SCRAM-SHA-256
- Xác thực dựa trên chứng chỉ - ứng dụng khách xác minh danh tính bằng chứng chỉ x.509 được ký bởi Tổ chức Phát hành Chứng chỉ (Certificate Authority) đáng tin cậy.
- Xác thực bên ngoài - máy khách xác minh danh tính bằng dịch vụ bên ngoài, chẳng hạn như Kerberos hoặc LDAP.

MongoDB Community hỗ trợ xác thực kiểu SCRAM, x.509, còn với phiên bản MongoDB Atlas và MongoDB Enterprise sẽ hỗ trợ thêm các cơ chế xác thực bên ngoài: xác thực dùng LDAP proxy, xác thực Kerberos.

3.1.1.1. Sử dụng SCRAM để xác thực các kết nối của người dùng

a) Giới thiệu về SCRAM

Salted Challenge Response Authentication Mechanism (SCRAM) là cơ chế xác thực người dùng mặc định cho MongoDB. Khi một người dùng xác thực vào server,

MongoDB sẽ dùng SCRAM để xác thực các thông tin được người dùng cung cấp về tên đăng nhập, mật khẩu và cơ sở dữ liệu muốn được xác thực để truy cập

b) Tính năng đặc trưng khi triển khai bảo mật kiểu SCRAM

- Các hệ số công việc có thể điều chỉnh. Ví dụ như số lần lặp (iteration count) trong phương thức dẫn xuất khóa (Key derivation function).
- Mật mã Salt ngẫu nhiên cho mỗi người dùng.
- Xác thực danh tính hai chiều giữa máy chủ và máy khách

c) Cách bật tính năng kiểm soát người dùng và tạo tài khoản với mongod

- **Bước 1**: Khởi động MongoDB ở chế độ không kiểm soát người truy cập
- **Bước 2**: Kết nối mongosh tới mongod cần thiết lập
- **Bước 3**: Tạo user quản trị (administrator) bằng cách dùng collection admin và chạy phương thức db.createUser(). Ví dụ

db.createUser(

{

user: "vunguyen",

pwd: "vunguyen",

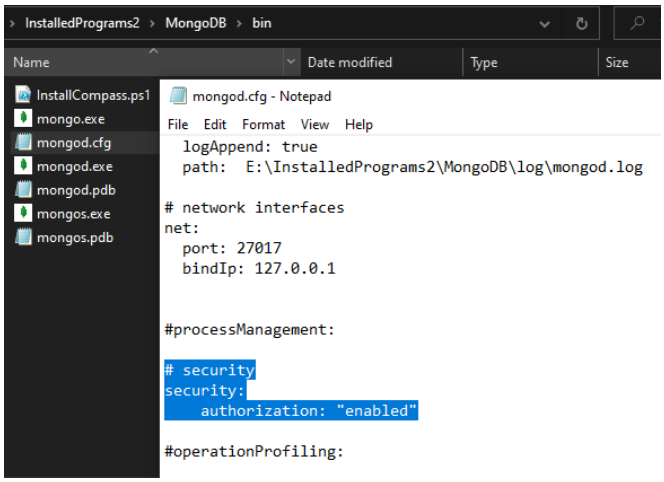
roles: [

{ role: "userAdminAnyDatabase", db: "admin" },

{ role: "readWriteAnyDatabase", db: "admin" }

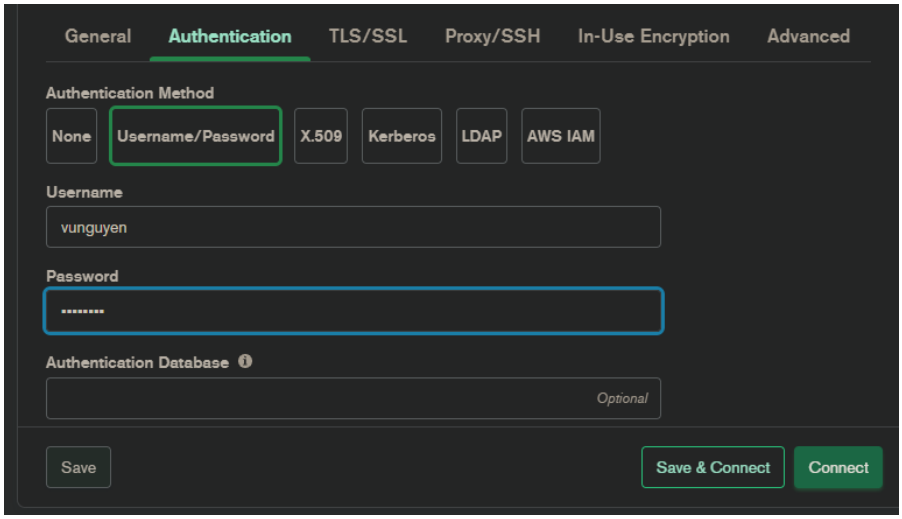
]})

- **Bước 4**: Bật chế độ kiểm soát người truy cập của mongod. Mở file config mongod.conf và chỉnh sửa phần security thêm authorization: enabled



Hình 3.1: Bật chế độ kiểm soát người truy cập của Mongo bằng cách sửa file mongod.conf

- **Bước 5**: Kết nối lại MongoDB và xác thực dùng tài khoản admin vừa tạo



Hình 3.2: Sử dụng MongoDB Compass và tiến hành đăng nhập

- **Bước 6**: Tạo một người dùng mới bằng tài khoản admin

```
db.createUser(
 {
   user: "user1",
   pwd:  "user1",
   roles: [ { role: "readWrite", db: "quan_ly_sach_2" }]})
```

- **Bước 7**: Kết nối lại MongoDB và xác thực bằng tài khoản mới tạo

3.1.2. Phân quyền và kiểm soát người dùng truy cập thông qua các role

MongoDB sử dụng Kiểm soát truy cập dựa trên các role (Role-Based Access Control - RBAC) để quản lý quyền truy cập vào hệ thống MongoDB. Người dùng được cấp một hoặc nhiều role xác định quyền truy cập của người dùng vào các hoạt động và tài nguyên cơ sở dữ liệu. người dùng không có quyền truy cập vào hệ thống ngoài các role được gán

3.1.2.1. Một số định nghĩa

Resource: Một resource (tài nguyên) trong MongoDB có thể là một cơ sở dữ liệu, một collection, một bộ các collection hoặc một cluster. Nếu tài nguyên là cluster, các hành động liên kết ảnh hưởng đến trạng thái của hệ thống hơn là một cơ sở dữ liệu hoặc collection cụ thể.

Privilege: Privilege (Đặc quyền) bao gồm một resource cụ thể và các hành động được phép trên resource. Một role cấp privilege (đặc quyền) để thực hiện các

hành động được chỉ định trên một resource. Mỗi privilege hoặc được chỉ định rõ ràng trong role hoặc được kế thừa từ một role khác hoặc cả hai.

3.1.2.2. Role được thiết lập sẵn và Role do người dùng định nghĩa

a) Các role được thiết lập sẵn

- Các role cho người dùng cơ sở dữ liệu: read, readWrite
- Các role cho người quản trị cơ sở dữ liệu: dbAdmin, dbOwner, userAdmin
- Các role cho người quản trị Cluster: Cơ sở dữ liệu admin có định nghĩa thêm các role để quản trị toàn bộ hệ thống thay vì một cơ sở dữ liệu riêng lẻ: clusterAdmin, clusterManager, clusterMonitor, và hostManager
- Các role cho sao lưu và phục hồi: Cơ sở dữ liệu admin định nghĩa sẵn các role backup, restore
- Các role áp dụng lên toàn bộ các cơ sở dữ liệu trừ local và config: readAnyDatabase, readWriteAnyDatabase, userAdminAnyDatabase, dbAdminAnyDatabase
- Các role Superuser: Một số role trực tiếp hay gián tiếp có quyền truy cập tới toàn bộ hệ thống. Các role sau đây có thể gán bất kì privilge nào ở bất kì cơ sở dữ liệu nào: dbOwner, userAdmin, userAdminAnyDatabase, root
- Role nội bộ hệ thống: __system

b) Role do người dùng định nghĩa

- Để tạo một role mới trong MongoDB, sử dụng phương thức db.createRole().
- MongoDB cũng cung cấp các phương thức để tiến hành chỉnh sửa các role người dùng đã định nghĩa trước đó

3.1.2.3. Các thao tác trên role

a) Hiển thị role của người dùng: Sử dụng phương thức db.getUser()

b) Hiển thị privilege của một role

- Để xem thông tin của một role, người dùng phải được cấp role đó, hoặc được phép thực thi viewRole trên cơ sở dữ liệu hiện tại
- Sử dụng phương thức db.getRole() hoặc lệnh rolesInfo, với thiết lập showPrivileges: true

c) Xóa role khỏi người dùng: Dùng phương thức db.revokeRolesFromUser().

d) Gán role cho người dùng: Dùng phương thức db.grantRolesToUser().

3.1.2.4. Thay đổi mật khẩu và dữ liệu tùy chỉnh

Để sửa đổi mật khẩu và dữ liệu tùy chỉnh phải có privilege được cấp cho các hành động changeOwnPassword và changeOwnCustomData tương ứng trên cơ sở dữ liệu của người dùng. Có 2 cách thay đổi mật khẩu và dữ liệu tùy chỉnh:

- Dùng phương thức db.updateUser() để cập nhật mật khẩu và dữ liệu tùy chỉnh.
- Dùng phương thức passwordPrompt () kết hợp với các phương thức người dùng khác nhau để nhắc nhập mật khẩu thay vì chỉ định mật khẩu trực tiếp trong phương thức

3.1.2.5. Kiểm soát quyền truy cập mức độ Collection

Bằng cách chỉ định cơ sở dữ liệu và collection trong phần định nghĩa tài nguyên của một privilege, quản trị viên có thể giới hạn các hành động chỉ trong một collection cụ thể trong một cơ sở dữ liệu cụ thể. Mỗi hành động privilege trong một role có thể được xác định phạm vi cho một collection khác. Ví dụ:

privileges: [

{ resource: { db: "products", collection: "inventory" }, actions: ["find", "update", "insert"] },

{ resource: { db: "products", collection: "orders" }, actions: ["find"] }]

3.1.2.6. Các bước tiến hành thiết lập các role hỗ trợ phân quyền và kiểm soát truy cập người dùng

- **Bước 1**: Tạo tài khoản user administartor
- **Bước 2**: Từ tài khoản admin, tạo thêm những tài khoản duy nhất cho từng người /ứng dụng truy cập vào server
- **Bước 3**: Tạo các role định nghĩa các quyền truy cập cần có của một nhóm người dùng
- **Bước 4**: Tạo các tài khoản người dùng và gán cho họ những role mà họ cần để thực hiện chức năng vai trò của họ

3.1.3. Mã hóa truyền dữ liệu (TLS/SSL)

Mã hóa truyền dữ liệu có tác dụng ngăn chặn người lạ không cho sao chép dữ liệu trong khi dữ liệu đang được truyền đi MongoDB hỗ trợ TLS / SSL (Transport Layer Security/Secure Sockets Layer) để mã hóa tất cả lưu lượng mạng của

MongoDB. TLS / SSL đảm bảo rằng lưu lượng mạng MongoDB chỉ có thể đọc được bởi đúng máy khách được nhắm tới

3.1.4. Mã hóa và bảo vệ dữ liệu

Mã hóa dữ liệu lưu trữ là công việc cần thiết để phòng trường hợp các tệp chứa cơ sở dữ liệu bị sao chép bất hợp pháp. Việc sao chép các tệp cơ sở dữ liệu có thể xảy ra khi có người đột nhập vào trung tâm dữ liệu (data center) và đánh cắp đĩa cứng của máy chủ. Tiến hành mã hóa dữ liệu trong lớp lưu trữ với tính năng Encryption at Rest (mã hóa ở trạng thái nghỉ) của của công cụ lưu trữ WiredTiger, hoặc sử dụng mã hóa hệ thống tệp, mã hóa thiết bị hoặc mã hóa vật lý (ví dụ: dm-crypt). Ngoài ra cũng nên bảo vệ dữ liệu MongoDB bằng cách sử dụng các quyền tệp hệ thống. Có thể sử dụng Mã hóa có thể truy vấn (Queryable Encryption) hoặc Mã hóa các trường dữ liệu phía máy khách (Client-Side Field Level Encryption) để mã hóa các trường dữ liệu trong document phía ứng dụng trước khi truyền dữ liệu đến máy chủ.

3.1.5. Kiểm tra (Auditing)

Kiểm tra giúp theo dõi người dùng cơ sở dữ liệu, ngay cả khi người đó xóa dấu vết của họ sau khi thay đổi hoặc thay đổi dữ liệu cơ sở dữ liệu, cho phép theo dõi quyền truy cập và thay đổi cấu hình cơ sở dữ liệu và dữ liệu. MongoDB Enterprise có một cơ sở hệ thống kiểm tra có thể ghi lại các sự kiện hệ thống như hoạt động của người dùng và các sự kiện kết nối trên một phiên bản MongoDB.

MongoDB tập hợp các log nhật ký vào một chỗ trung tâm. Các nhật ký này chứa các nỗ lực xác thực cơ sở dữ liệu, bao gồm địa chỉ IP nguồn. Các hồ sơ kiểm tra này giúp cho việc phân tích điều tra và cho phép quản trị viên xác minh các biện pháp kiểm soát thích hợp.

3.2. Index

3.2.1. Giới thiệu về Index

Index là một cấu trúc dữ liệu đặc biệt, ứng dụng cây nhị phân để lưu trữ một phần của dữ liệu theo cách dễ duyệt và tìm kiếm. Index chứa dữ liệu của một trường hoặc một nhóm các trường cụ thể, và sắp xếp theo giá trị trong các trường đó. Việc sắp xếp các giá trị trong index giúp cho các truy vấn liên quan đến khoảng giá trị hoặc tìm giá trị khớp được hiệu quả hơn. Ngoài ra, MongoDB có thể trả về kết quả được

sắp xếp sẵn khi dùng trình tự sắp xếp trong index. MongoDB có thể tận dụng index để giới hạn số lượng document cần kiểm tra

Các loại index: MongoDB cung cấp các loại index khác nhau để phục vụ cho từng loại dữ liệu và câu truy vấn: Single Field, Compound Index, Multikey Index, Geospatial Index, Text Index, Hashed Index, Clustered Index

Các thuộc tính của Index: Unique Index, Partial Index, Sparse Index, TTL Index, Hidden Index

3.2.2. Tạo một index

Để tạo một index trong Mongo shell, dùng lệnh **db.collection.createIndex()** theo cú pháp sau:

db.collection.createIndex(<tên trường dữ liệu và kiểu index kèm thông số>, <các tùy chọn khác>)

Ví dụ: tạo ra một index trên trường name, sắp xếp giá trị theo thứ tự giảm dần: db.collection.createIndex({ name: -1 })

3.2.3. Sử dụng Index để hỗ trợ truy vấn

Index có thể giúp cải thiện tốc độ truy vấn trong cơ sở dữ liệu. Khi truy vấn dữ liệu kèm theo chỉ định vùng cần tìm kiếm mà tương ứng với một index đã thiết lập, MongoDB sẽ trả về kết quả từ các giá trị trong index thay vì quét từng document hay đưa document vào vùng nhớ

Ngoải ra, MongoDB có thể thực hiện Index Intersection bằng cách kết hợp các index đã thiết lập khác nhau để xử lý truy vấn nếu cho kết quả phù hợp với yêu cầu

3.3. Replica Set

3.3.1. Giới thiệu về replica set

Replica set là một nhóm các trình mongod cùng quản lý một bộ dữ liệu. Replica set giúp dữ liệu tồn tại ở nhiều nơi và luôn sẵn sàng cung cấp thông tin cho các ứng dụng, góp phần tránh gián đoạn dịch vụ khi có sự cố server; giúp tăng khả năng máy khách đọc được dữ liệu, dữ liệu phân bố cục bộ và tập trung hơn theo khu vực. Ngoài ra, replica set còn hỗ trợ việc sao lưu, khôi phục dữ liệu, báo cáo, …

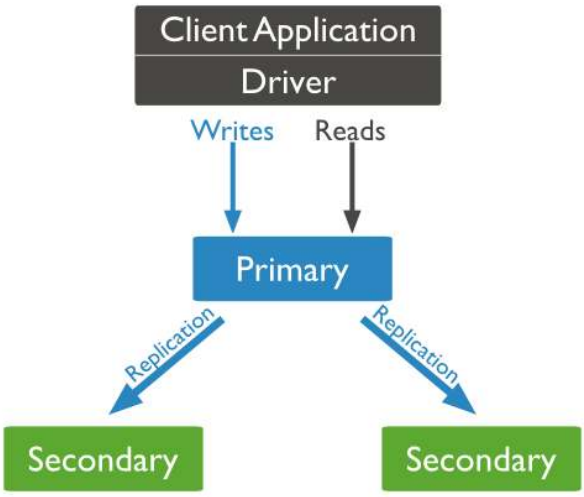
3.3.2. Cấu trúc của một replica set

Một replica set bao gồm nhiều node (tức server), trong đó có một Primary node duy nhất, các node còn lại là các Secondary node, ngoài ra có thể chứa thêm arbiter

node để hỗ trợ chọn primary node khi một server gặp sự cố. Primary node và Secondary node đều phụ trách chứa dữ liệu, còn Arbiter node thì không

Primary node phụ trách toàn bộ việc ghi dữ liệu, lưu các thay đổi và chỉnh sửa trong dữ liệu vào collection tên oplog. Các secondary node sẽ sao chép collection oplog của primary node, dựa vào đó thực hiện các thao tác trên dữ liệu lưu ở server của chính nó, làm sao để dữ liệu trên server giống dữ liệu trên primary node. Việc sao chép collection oplog và thực hiện lại thao tác ghi trong đó diễn ra đồng thời với nhau, đảm bảo dữ liệu trên các secondary node giống như dữ liệu trên primary node, giúp replica set có thể hoạt động và cung cấp dữ liệu ngay cả khi có node không hoạt động

Dưới đây là sơ đồ đặc trưng cho Replication trong MongoDB, trong đó ứng dụng ở Client luôn luôn tương tác với primary node và primary node này sau đó tái tạo dữ liệu cho secondary node



Hình 3.3: Sơ đồ mối quan hệ các thành phần trong Replica Set

Mỗi node trong trong replica set sẽ gửi tín hiệu ping (heartbeat) tới các node khác mỗi 2 giây. Nếu một node không có tín hiệu ping trả về trong vòng 10 giây, các node khác sẽ đánh dấu node đó là không truy cập được. Trong trường hợp đó, một secondary node bất kì sẽ tiến hành tổ chức cuộc bình chọn để các node có thể chọn ra một secondary node trở thành primary node mới.

Cấu hình của một replica set khuyến nghị tối thiểu nên có 3 node phụ trách chứa dữ liệu: một primary node và hai secondary node. Trong một số trường hợp, chẳng hạn replica set có hai (02) node, một primary node và một secondary node, có

thể thêm một mongod đóng vai trò arbiter node. Arbiter node này không chứa dữ liệu mà chỉ tham gia bình chọn để phá thế hòa giúp chọn ra node nào là primary node.

3.3.3. Các bước thiết lập một replica set

Các bước sau đây sẽ mô tả cách tạo ra replica set gồm 3 node từ 3 mongod có sẵn, trường hợp không cần xác thực

Bước 1: Khởi động từng server mongod của replica set theo cú pháp sau:

mongod --replSet "rs0" --bind_ip localhost,<hostname|địa chỉ ip >

Trong đó mục <hostname|địa chỉ ip > chỉ định tên miền và/hoặc địa chỉ IP của mongod mà các máy khách từ xa (bao gồm cả các mongod khác trong cùng replica set) có thể kết nối và sử dụng.

Bước 2: Kết nối mongosh tới một trong các node mongod

Bước 3: Khởi tạo replica set. Trong mongosh chạy phương thức rs.initiate(). Thiết lập đối số của phương thức theo mẫu như sau:

rs.initiate({
_id : "rs0",
members: [
{ _id: 0, host: "mongodb0.example.net:27017" },
{ _id: 1, host: "mongodb1.example.net:27017" },
{ _id: 2, host: "mongodb2.example.net:27017" }
]})

Khi đó MongoDB sẽ khởi tạo ra một replica set dùng các giá trị đã thiết lập

Bước 4: Xem các thông số cài đặt của replica set thông qua lệnh **rs.conf()**

Bước 5: Kiểm tra trạng thái replica set hiện tại thông qua lệnh **rs.status()**

3.4. Phân tán dữ liệu

3.4.1. Giới thiệu về Sharding

Sharding là một phương thức dùng để phân phối dữ liệu trên nhiều máy khác nhau. MongoDB sử dụng sharding để hỗ trợ triển khai các ứng dụng có các tập dữ liệu rất lớn và có các hoạt động trên dữ liệu mà cần thông lượng cao, hỗ trợ horizontal scaling (mở rộng theo chiều ngang) giúp tài nguyên hệ thống bắt kịp yêu cầu khả năng xử lý bằng cách phân chia tập dữ liệu hệ thống và tải trên nhiều máy chủ khác nhau, thêm các máy chủ bổ sung để tăng dung lượng theo yêu cầu.

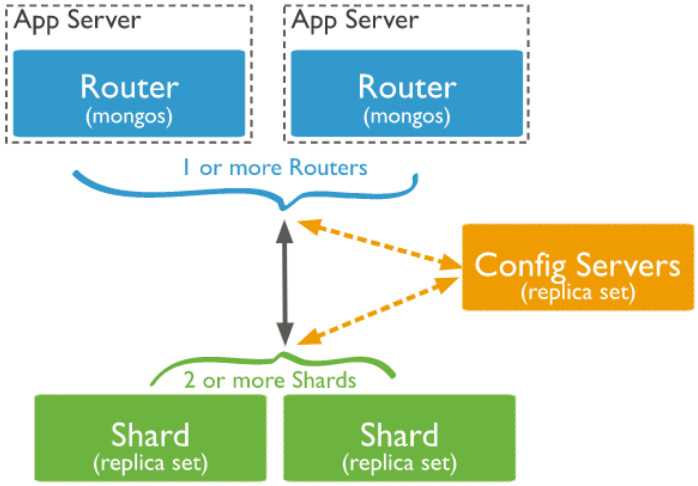
3.4.2. Một số định nghĩa

3.4.2.1. Sharded Cluster

Một sharded cluster trong MongoDB chứa các thành phần sau:

- **Shard**: mỗi shard chứa một tập hợp con của dữ liệu được chia nhỏ. Mỗi shard có thể được triển khai cài đặt như một tập bản sao của dữ liệu gốc (replica set)
- **Mongos**: đóng vai trò là query router, là lớp interface trung gian giữa các ứng dụng bên máy khách và sharded cluster.
- **Config servers**: thiết lập cấu hình máy chủ lưu trữ các thông tin metadata và các cài đặt cho cluster

MongoDB phân đoạn dữ liệu tới mức độ collection, phân phối dữ liệu của collection trên các shard khác nhau trong cluster.



Hình 3.4: Sơ đồ mối quan hệ các thành phần trong Sharded Cluster

3.4.2.2. Shard Key và Chunk

Shard Key: MongoDB sử dụng shard key để phân phối document của collection trên các shard. Giá trị của shard key bao gồm giá trị của một trường hoặc nhiều trường dữ liệu trong các document. Khi phân mảnh một collection ra nhiều shard cần tiến hành chọn các shard key. Giá trị shard key của một document sẽ quyết định cách nó sẽ được phân đoạn như thế nào trên các shard khác nhau

Chỉ số Shard Key: Để phân đoạn một collection lên các shard khác nhau, collection đó phải có một chỉ số (index) bắt đầu bằng giá trị shard key. Khi phân đoạn một collection trống, MongoDB sẽ hỗ trợ tạo chỉ số nếu collection đó chưa có chỉ số

phù hợp cho shard key. Việc lựa chọn khóa shard key ảnh hưởng đến hiệu suất, hiệu quả và khả năng mở rộng của một sharded cluster

Chunk: MongoDB phân dữ liệu trong shard thành các các chunk. Mỗi chunk có khoảng giới hạn giá trị dựa trên shard key

3.4.2.3. Collection được chia nhỏ và không được chia nhỏ

Một cơ sở dữ liệu có thể có một hỗn hợp các collection được phân đoạn và collection không được phân đoạn. Các collection đã được phân đoạn sẽ được chia vùng và phân phối trên các shard trong cluster. collection không được phân đoạn được lưu trữ trên shard chính. Mỗi cơ sở dữ liệu có một shard chính của riêng nó.

3.4.3. Phương pháp Sharding

3.4.3.1. Sharding theo giá trị băm (Hashed Sharding)

Hashed Sharding liên quan đến việc tính toán một hàm băm dựa trên giá trị của trường shard key. Sau khi tính toán các giá trị của shard key và có được một tập hợp các giá trị băm, mỗi chunk được gán một khoảng giá trị dựa trên các giá trị băm đó. Khi đó các shard key có thể có giá trị sát nhau, nhưng các giá trị băm sau khi tính toán của chúng sẽ cách nhau dẫn đến shard key sẽ không nằm trên cùng một chunk. Phân phối dữ liệu dựa trên các giá trị băm tạo điều kiện phân phối dữ liệu đồng đều hơn, đặc biệt là trong các tập dữ liệu mà shard key thay đổi theo một hàm đơn điệu.

Tuy nhiên phân phối dữ liệu kiểu hàm băm khiến cho việc truy xuất và thao tác dữ liệu dựa theo các khoảng giá trị của shard key có khả năng sẽ nhắm tới nhiều shard thay vì một shard nhất định, do đó việc xử lý có thể diễn ra trên quy mô cluster liên quan đến nhiều shard thay vì chỉ một shard nhất định.

3.4.3.2. Sharding theo khoảng giá trị (Ranged Sharding)

Sharding theo khoảng giá trị là phân chia dữ liệu thành các khoảng giá trị dựa trên chính các giá trị của shard key. Mỗi chunk được gán một phạm vi chứa giá trị dựa trên các giá trị của shard key. Khi đó, các shard key có giá trị sát nhau có nhiều khả năng nằm trên cùng một chunk, cho phép việc truy xuất và thao tác dữ liệu dựa theo các khoảng giá trị của shard key sẽ chỉ nhắm tới một shard nhất định

Hiệu quả của Ranged Sharding phụ thuộc vào giá trị shard key được chọn. Hệ thông các shard key không được tổ chức cẩn thận có thể khiến việc phân phối dữ liệu

không đồng đều, dẫn đến không tận dụng được một số lợi ích của việc sharding hoặc có thể gây ra tắc nghẽn khi xử lý dữ liệu

3.4.4. Zone trong Sharded Cluster

Zone có thể hỗ trợ việc khoanh vùng dữ liệu cho các shared cluster mà các phân đoạn đó tồn tại trên nhiều trung tâm dữ liệu khác nhau. Trong các cluster phân đoạn, có thể tạo các vùng (zone) chứa các phân đoạn dữ liệu dựa trên shard key. Mỗi zone bao gồm một hoặc nhiều khoảng giá trị shard key. Đồng thời, zone có thể liên kết với một hoặc nhiều shard trong cluster. Một shard có thể liên kết với một hay nhiều zone. Trong một cluster, MongoDB chỉ di chuyển các chunk chứa trong một zone đến các shard được liên kết với zone đó.

3.4.5. Cách triển khai một Sharded Cluster

Mỗi thành phần trong sharded cluster phải đảm bảo có thể kết nối với tất cả các thành phần khác trong cluster, bao gồm tất cả các shard và config server. Quy trình tổng quát để triển khai một Sharded Cluster gồm các bước sau: Tạo config server replica set, Tạo các shard replica set, Khởi động mongos để khởi động Sharded Cluster, Kết nối với Sharded Cluster, và Phân mảnh một Collection

3.4.5.1. Tạo config server replica

Bước 1: Với mỗi một mongod, tiến hành thiết lập các giá trị cài đặt mongod thông qua file configuaration hoặc qua câu lệnh command line

mongod --configsvr --replSet <replica set name> --dbpath <path> --bind_ip localhost,<hostname(s)|ip address(es)>

Bước 2: Kết nối mongosh tới một trong các config server

mongosh --host <hostname> --port <port>

Bước 3: Khởi chạy replica set. Trong mongosh, chạy phương thức **rs.initiate()**

rs.initiate() nhận đối số là các thông số cài đặt replica set. Các thông số gồm:

- *_id*: tên định danh của replica set (lấy từ replication.replSetName hoặc –replSet)
- *configsvr*: để giá trị true
- *members*: mảng chứa các thành phần mongod của replica set

3.4.5.2. Tạo các shard replica

Khi triển khai sharded cluster trong thực tế, dùng replica set với ít nhất 3 thành phần mongod

Bước 1: Khởi động từng shard. Thiết lập các thông số khởi động của mongod thông qua file config hoặc câu lệnh command line

mongod --shardsvr --replSet <replSetname> --dbpath <path> --bind_ip localhost,<hostname(s)|ip address(es)>

Bước 2: Kết nối mongosh tới một trong các shard đã khởi động

mongosh --host <hostname> --port <port>

Bước 3: Khởi tạo replica set. Trong mongosh, chạy phương thức **rs.initiate()**

rs.initiate() nhận đối số là các thông số cài đặt replica set. Các thông số gồm:

- *_id*: tên định danh replica set (lấy từ replication.replSetName hoặc –replSet)
- *members*: mảng chứa các thành phần mongod của replica set

3.4.5.3. Khởi động mongos để khởi động Sharded Cluster

Khởi động mongos kèm theo thiết lập các thông số khởi động của mongod thông qua file config hoặc câu lệnh command line

mongos --configdb <configReplSetName>/cfg1.example.net:27019,cfg2.example.net:27019,cfg3.example.net:27019 --bind_ip localhost,<hostname(s)|ip address(es)>

3.4.5.4. Kết nối với Sharded Cluster

Kết nối mongosh tới mongos vừa tạo, chỉ rõ mongos chạy trên host và port nào

mongosh --host <hostname> --port <port>

3.4.5.5. Thêm shard vào cluster

Trong một phiên mongosh đang kết nối tới mongos, dùng phương thức **sh.addShard()** để thêm shard vào cluster

sh.addShard("<replSetName>/s1-mongo1.example.net:27018,s1-mongo2.example.net:27018,s1-mongo3.example.net:27018")

3.4.5.6. Phân mảnh một Collection

Để phân mảnh một collection, kết nối mongosh tới mongos và dùng phương thức **sh.shardCollection().**Có 2 phương pháp phân mảnh một collection:

- Sharding theo giá trị băm (hashed Sharding):

sh.shardCollection("<database>.<collection>", { <shard key field> : "hashed" })

- Sharding theo khoảng giá trị (Ranged Sharding): *sh.shardCollection("<database>.<collection>", { <shard key field> : 1, ... })*

3.5. Sao lưu và khôi phục

Các mongodump và mongorestore tận dụng việc xuất dữ liệu BSON để tạo bản sao lưu các triển khai cơ sở dữ liệu nhỏ. Do mongodump và mongorestore hoạt động bằng cách tương tác với một phiên bản mongod đang chạy, chúng có thể ảnh hưởng đến hiệu suất của cơ sở dữ liệu đang chạy. Sử dụng sao lưu thay thế như Filesystem Snapshots hoặc MongoDB Cloud Manager nếu tác động hiệu suất của mongodump và mongorestore là không thể chấp nhận.

3.5.1. Sao lưu cơ sở dữ liệu với mongdump

3.5.1.1. Tạo bản sao lưu cho mongod chạy trên local

Loại trừ cơ sở dữ liệu cục bộ: mongodump loại trừ nội dung của cơ sở dữ liệu cục bộ trong đầu ra của nó.

Quyền truy cập bắt buộc: phải có đặc quyền cấp hành động tìm kiếm cho mỗi cơ sở dữ liệu để sao lưu. Vai trò sao lưu tích hợp cung cấp các đặc quyền cần thiết để thực hiện sao lưu bất kỳ và tất cả các cơ sở dữ liệu.

Các mongodump sao lưu dữ liệu bằng cách kết nối với mongod đang chạy. Tiện ích này có thể tạo bản sao lưu cho toàn bộ máy chủ, cơ sở dữ liệu hoặc collection hoặc có thể sử dụng truy vấn để sao lưu một phần của collection

Các thao tác mongodump cơ bản:

- Khi chạy *mongodump* mà không có bất kỳ đối số nào, lệnh sẽ tiến hành kết nối với MongoDB trên hệ thống cục bộ (ví dụ: localhost) trên cổng 27017 và tạo bản sao lưu cơ sở dữ liệu có tên là dump / trong thư mục hiện tại.
- Có thể dùng đối số *--host* và *--port* để chỉ ra MongoDB nên kết nối tới.
- Để tùy chỉnh thư mục đầu ra, sử dụng đối số *--out* hoặc *-o*
- Nếu chỉ muốn sao lưu một phần cơ sở dữ liệu, sử dụng các đối số *--db và --collection* để chỉ ra cơ sở dữ liệu hay collection nào muốn sao lưu

Ví dụ: *mongodump --username=uit --password=uit1 --authenticationDatabase=admin --db=rangerShop --out=D:\backup*

- Tạo bản sao lưu bằng Oplog:

Các *--oplog* tùy chọn với *mongodump* thu thập các mục oplog và cho phép thực hiện sao lưu trên cơ sở dữ liệu trực tiếp. Nếu sau đó khôi phục cơ sở dữ liệu từ bản sao lưu này, cơ sở dữ liệu sẽ giống như khi quá trình sao lưu hoàn tất. Với --oplog, mongodump sao chép tất cả dữ liệu từ cơ sở dữ liệu nguồn cũng như tất cả các mục oplog từ đầu đến cuối quy trình sao lưu. Hoạt động này kết hợp với mongorestore --oplogReplay cho phép khôi phục bản sao lưu phản ánh thời điểm cụ thể tương ứng với thời điểm mongodump hoàn thành tạo tệp xuất.

3.5.1.2. Tạo bản sao lưu từ các phiên bản Non-Local mongod

Các --host và --port cho mongodump cho phép bạn kết nối và sao lưu host từ xa, như ví dụ sau: "*mongodump --host=mongodb1.example.net --port=3017 --username=user --password="pass" --out=/opt/backup/mongodump-2013-10-24"*

Bất cứ lệnh mongodump sử dụng sẽ liên quan đến phân quyền username password trên database

3.5.2. Khôi phục cơ sở dữ liệu với mongorestore

Để khôi phục dữ liệu một triển khai MongoDB có kiểm soát truy cập, cần có role cung cấp các đặc quyền cần thiết để khôi phục dữ liệu từ các bản sao lưu nếu dữ liệu không bao gồm dữ liệu thu thập system.profile và chạy mongorestore không có –oplogReplay.

Nếu dữ liệu sao lưu bao gồm dữ liệu thu thập system.profile hoặc chạy với –oplogReplay và cơ sở dữ liệu đích không chứa tập hợp system.profile thì cần phải có các đặc quyền sau:

- Mongorestore cố gắng tạo collection mặc dù chương trình không thực sự khôi phục tài liệu system.profile. Do đó, người dùng cần các đặc quyền bổ sung để thực hiện các hành động createCollection và convertToCapped trên system.profile cho cơ sở dữ liệu.
- Cả hai vai trò tích hợp dbAdmin và dbAdminAnyDatabase đều cung cấp các đặc quyền bổ sung.

3.5.2.1. Các thao tác cơ bản với mongorestore

Các mongorestore khôi phục bản sao lưu nhị phân được tạo bởi mongodump. Theo mặc định, mongorestore tìm kiếm một bản sao lưu cơ sở dữ liệu trong thư mục

dump / directory. Các mongorestore tiện ích khôi phục dữ liệu bằng cách kết nối trực tiếp với mongod đang chạy, mongorestore có thể khôi phục toàn bộ bản sao lưu cơ sở dữ liệu hoặc một tập hợp con của bản sao lưu.

Để sử dụng "*mongorestore*" giao tiếp với "*mongod*" sử dụng lệnh dưới đây *"mongorestore --port=<port number> <path to the backup>"*

3.6. Nhập và xuất dữ liệu

MongoDB Compass hỗ trợ nhập và xuất dữ liệu JSON lẫn CSV

3.6.1. Nhập dữ liệu vào Collection

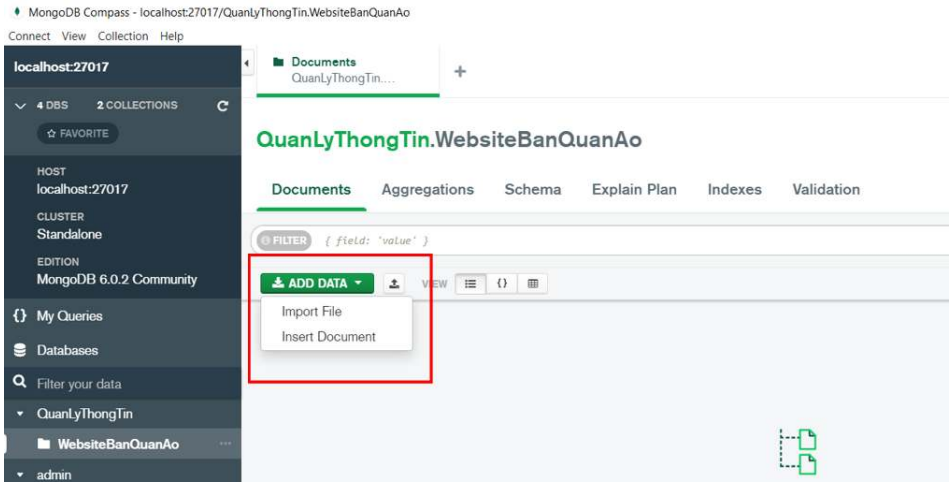
Một số hạn chế khi nhập dữ liệu thông qua MongoDB Compass bao gồm chức năng nhập dữ liệu không được hỗ trợ ở phiên bản Read Only của MongoDB Compass, và chức năng này sẽ không hoạt động nếu đang kết nối đến Data Lake

Trước khi nhập data vào MongoDB Compass phải chắc rằng dữ liệu đã được định dạng đúng đến Compass hiểu được

- Với định dạng file JSON: Mỗi dòng là một dữ liệu thông tin. Dấu phẩy ngăn cách dữ liệu trong một mảng (không ngắt dòng giữa các phần tử trong mảng)
- Đối với định dạng file CSV: Dòng đầu tiên của file là tên cột phải được ngăn cách bằng các dấu phẩy, những dòng tiếp theo là các giá trị của cột theo thứ tự.

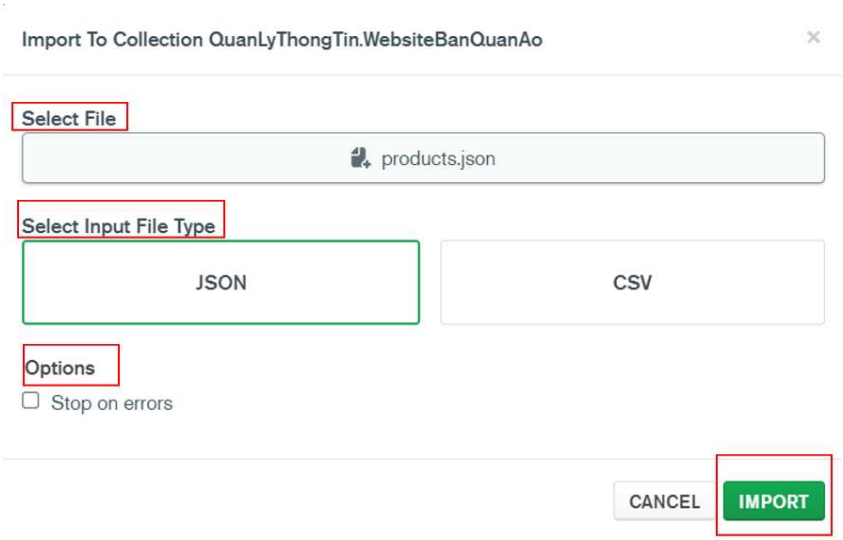
Quy trình:

- **Bước 1**: Kết nối với bộ tài liệu mà bạn muốn nhập dữ liệu
- **Bước 2**: Điều hướng đến bộ tài liệu muốn nhập
- **Bước 3**: Chọn "Add Data" và chọn "Import File"



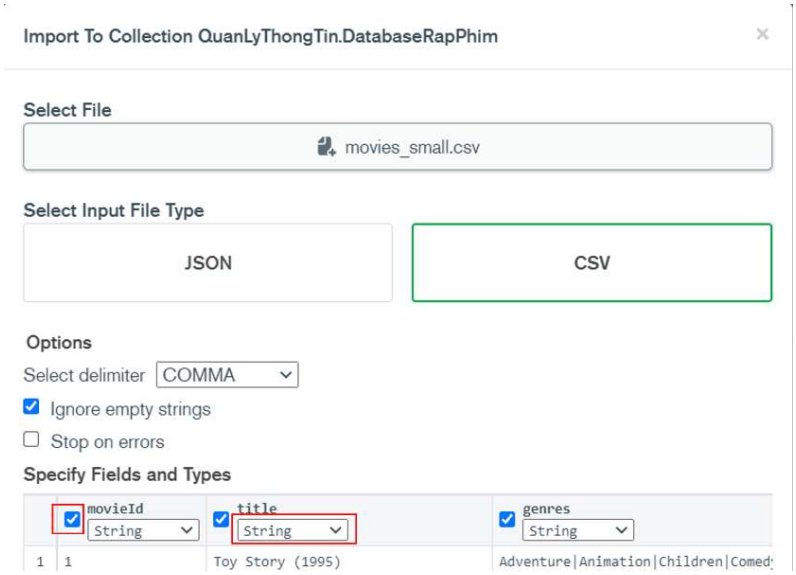
Hình 3.5: Click "Add Data" chọn Import File

- **Bước 4:** Chọn vị trí của tệp trong "Import File", chọn loại tệp thích hợp với tệp muốn thêm vào bao gồm JSON hoặc CSV



Hình 3.6: Chọn vị trí tệp và loại tệp cho loại tệp JSON

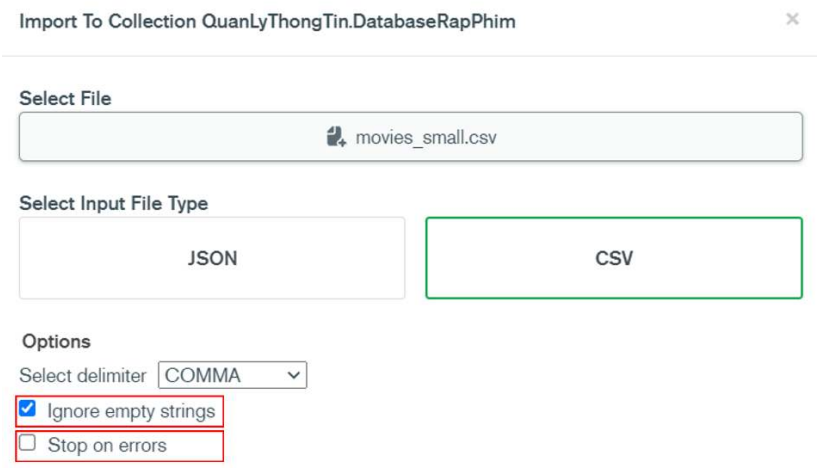
Giả sử nếu như chọn loại tệp là CSV thì sẽ có thể chỉ định các loại trường để nhập và loại dữ liệu của các trường đó. Kiểu dữ liệu cho tất cả các trường mặc định là kiểu chuỗi. Để loại trừ một trường khỏi tệp CSV thì hãy bỏ chọn hợp checkbox bên cạnh. Để chọn loại cho một trường thì sử dụng drop-down menu phía dưới trường đó.



Hình 3.7: Chọn trường muốn thêm và chọn loại dữ liệu cho trường cho loại tệp CSV

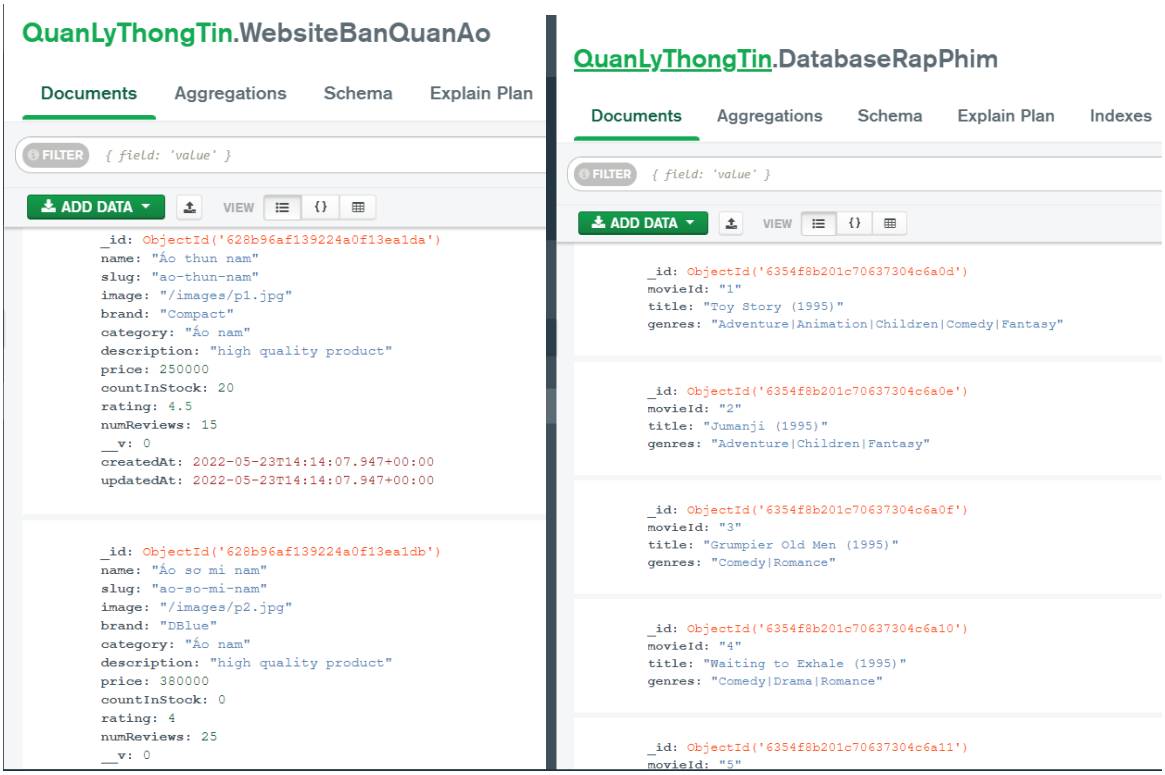
- **Bước 5:** Định cấu hình các tùy chọn nhập trong "Options"

Nếu đang nhập tệp CSV có thể chọn cách phân tách dữ liệu của mình. Đối với cả nhập tệp JSON và CSV, có thể chuyển đổi "bỏ qua chuỗi trống" (Ignore empty Strings) và "dừng khi có lỗi" (Stop on errors)



Hình 3.8: Chọn các options nâng cao cho loại tệp CSV

- **Bước 6:** Chọn Import



Hình 3.9: Màn hình hiển thị các dữ liệu file JSON (bên trái), CSV (bên phải) đã nhập vào

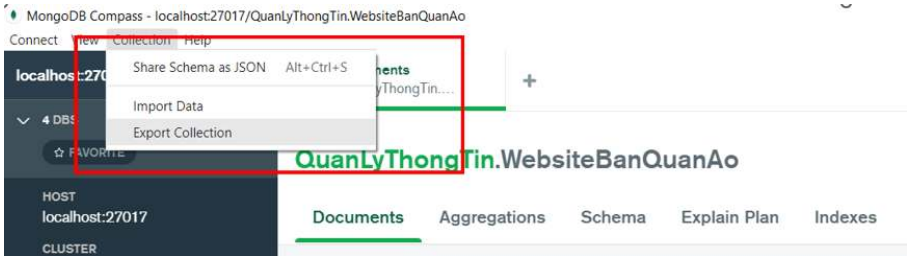
3.6.2. Xuất dữ liệu từ một Collection

MongoDB Compass có thể xuất dữ liệu dưới dạng tệp JSON hoặc CSV. Chỉ định bộ lọc hoặc đường dẫn tổng hợp cho nơi chứa dữ liệu của mình, Compass sẽ xuất các tài liệu phù hợp với kết quả truy vấn hoặc đường dẫn đã chỉ định. Cần lưu ý là không thể định hình lại tài liệu đã xuất bằng tài liệu dự án.

Để xuất toàn bộ bộ sưu tập sang một tệp:

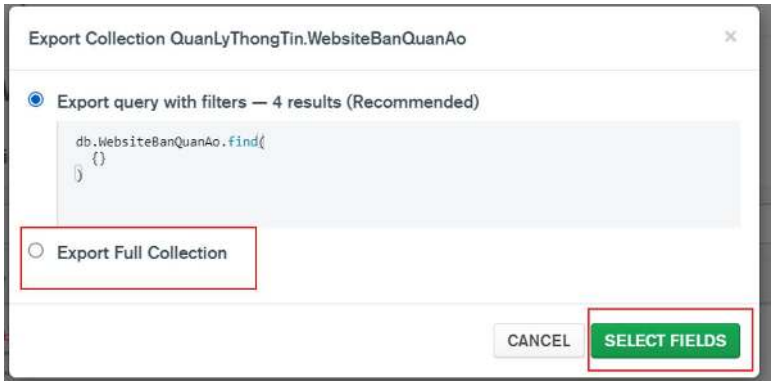
- **Bước 1**: Kết nối với triển khai có chứa Collection muốn truy xuất

- **Bước 2**: Nhấp vào Collection trên thanh menu và chọn "Export Collection"



Hình 3.10: Chọn Export Collection

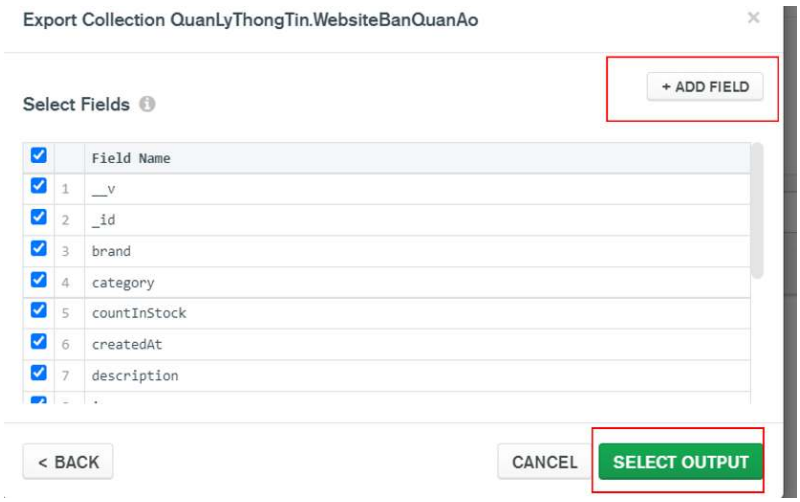
Compass sẽ xuất hiện hộp thoại như sau:



Hình 3.11: Hộp thoại chọn lựa cách xuất dữ liệu bằng cách query filter hay xuất toàn bộ

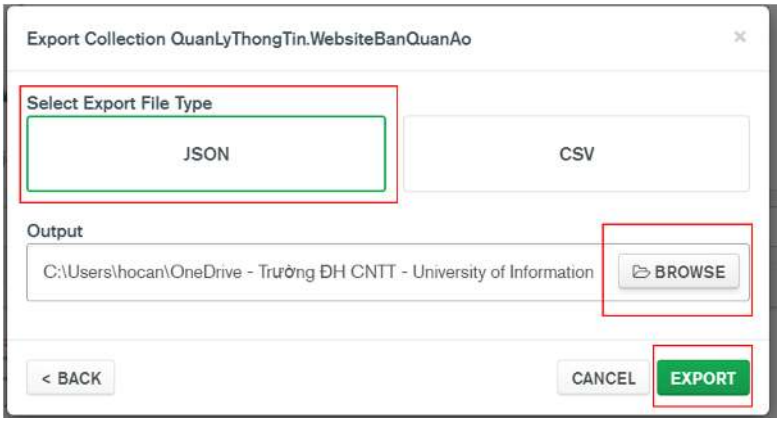
Hộp thoại hiển thị truy vấn mặc định sẽ được sử dụng để xuất dữ liệu. Nếu không có chỉnh sửa câu truy vấn, thao tác sẽ trả về tất cả các tài liệu trong collection. Nếu muốn bỏ qua bộ lọc và xuất toàn bộ dữ liệu thì chọn "Export Full Collection" và nhấp vào "Select Fields"

- **Bước 3**: Chọn các trường tài liệu đưa vào tệp sẽ xuất. Thêm các trường khác bằng nút "Add Field" nếu trường đó không được chương trình tự động phát hiện.



Hình 3.12: Chọn thêm trường để xuất hoặc hủy bỏ những trường không muốn xuất ra

- **Bước 5**: Chọn loại tệp và vị trí xuất: Trong Chọn loại tệp xuất, chọn JSON hoặc CSV. Nếu chọn JSON, dữ liệu sẽ được xuất sang tệp đích dưới dạng một mảng. Sau đó chọn "Select Output", chọn nơi xuất tệp sang.



Hình 3.13: Chọn Browse để chọn vị trí đích muốn xuất file ra, sau đó nhấn Export để tiến hành xuất

- **Bước 6**: Nhấp vào Export. Thanh tiến trình hiển thị trạng thái của quá trình xuất. Nếu xảy ra lỗi, thanh tiến trình sẽ chuyển sang màu đỏ và thông báo lỗi xuất hiện trong hộp thoại. Sau khi xuất thành công, hộp thoại sẽ đóng.

3.7. Một số tính năng khác trên MongoDB

Change stream: Change stream cho phép các ứng dụng tiếp cận thông tin về các thay đổi trong dữ liệu theo thời gian thực thay vì phải theo dõi oplog. Các ứng dụng kết nối với cơ sở dữ liệu MongoDB có thể dùng change strams để có được các thay đổi về dữ liệu lưu trên một collection đơn lẻ, một cơ sở dữ liệu hay nguyên cả dự án, và dựa trên đó có thể ngay lập tức có các thao tác phản ứng lại với các thay đổi đó.

Time Series: Time series data (dữ liệu theo thời gian) là một chuỗi các dữ liệu thay đổi theo thời gian, từ sự thay đổi của dữ liệu có thể phân tích, nghiên cứu, đưa ra các nhận xét về dữ liệu, ngoài ra còn giúp cải thiện việc truy vấn và tối ưu vùng nhớ lưu trữ dữ liệu và các index. Collection time series có thể thực hiện các thao tác như thêm và truy vấn như các collection bình thường

Thực hiện một transaction trên nhiều document: Trong MongoDB, một thao tác lên một document có tính nguyên tử. Tuy nhiên, đối với các trường hợp cần tính nguyên tử trên thao tác đọc và ghi cùng lúc trên nhiều document (có thể trên một hoặc nhiều collection), MongoDB hỗ trợ thực hiện transaction trên nhiều document.

CHƯƠNG 4: DEMO TÍNH NĂNG TRÊN MONGODB

4.1. Kịch bản demo

Tạo ứng dụng website bán quần áo rangerShop bằng cách tạo và lưu dữ liệu trên Sharded Cluster của MongoDB, cung cấp các tính năng cơ bản truy xuất, thêm, xóa, sửa người dùng, sản phẩm, đơn hàng. Backend sử dụng NodeJS và framework ExpressJS, truy vấn cơ sở dữ liệu sử dụng thư viện Mongoose. Frontend sử dụng ReactJS. Tiến hành sao lưu, khôi phục, xuất và nhập dữ liệu từ file JSON

Bảng 4.1: Thiết lập sharded cluster MongoDB trên localhost

Các thành phần	**Tên replica set**	**Hostname, port mongod của replica set**
Config Server Replica Set	config_repl	localhost:28041, localhost:28042, localhost:28043
Shard Replica Set (1)	shard_repl	localhost:28081, localhost:28082, localhost:28083
Shard Replica Set (2)	shard2_repl	localhost:29081, localhost:29082, localhost:29083
Mongos		localhost:27017

Bảng 4.2: Tạo người dùng và phân quyền

Host	**Username**	**Role trong database admin**
localhost:28081	shard_repl_admin	userAdminAnyDatabase, readWriteAnyDatabase, clusterAdmin
	adminRangerShopShard1	readWriteAnyDatabase
localhost:29081	shard2_repl_admin	userAdminAnyDatabase, readWriteAnyDatabase, clusterAdmin
	adminRangerShopShard2	readWriteAnyDatabase
localhost:27017	vunguyen	userAdminAnyDatabase, readWriteAnyDatabase, clusterAdmin
	nguoidung	readWriteAnyDatabase

Database tạo thêm trong Sharded Cluster: rangerShop

Bảng 4.3: Các collection kèm các thông tin trong database rangerShop

**Trường dữ liệu in nghiêng không bắt buộc phải có trong collection*

Collection	**Shard Keys**	**Kiểu dữ liệu trong collection**	
		Trường dữ liệu	**Kiểu BSON và yêu cầu dữ liệu**
users	{_id:1}	name	string
		email	string
		password	string
		isAdmin	bool
products	{_id:"hashed"}	name	string
		slug	string
		image	string
		brand	string
		category	string
		description	string
		price	number, minimum: 0
		countInStock	number, minimum: 0
		rating	number, minimum: 0
		numReviews	number, minimum: 0
		reviews	array, chứa các object Object trong mảng có các thuộc tính sau: name (string), comment (string), rating (number)
orders	{_id:"hashed"}	orderItems	array, chứa các object Object trong mảng có thuộc tính: name (string), slug (string), image (string), price (number, minimum:0), quantity (number, minimum:0), product (objectId)

		shippingAddress	object; có các thuộc tính sau: fullName (string), address (string), city (string), postalCode (string), country (string), *location (object)* location chứa các thuộc tính: *lat (number), lng (number), address (string), name (string), vicinity (string), googleAddressId (string)*
		paymentMethod	string
		paymentResult	object; có các thuộc tính sau: id(string), status(string), update_time(string)
		itemsPrice	number, minimum: 0
		shippingPrice	number, minimum: 0
		taxPrice	number, minimum: 0
		totalPrice	number, minimum: 0
		user	objectId
		isDelivered	bool
		deliveredAt	date

4.2. Các chức năng sẽ demo

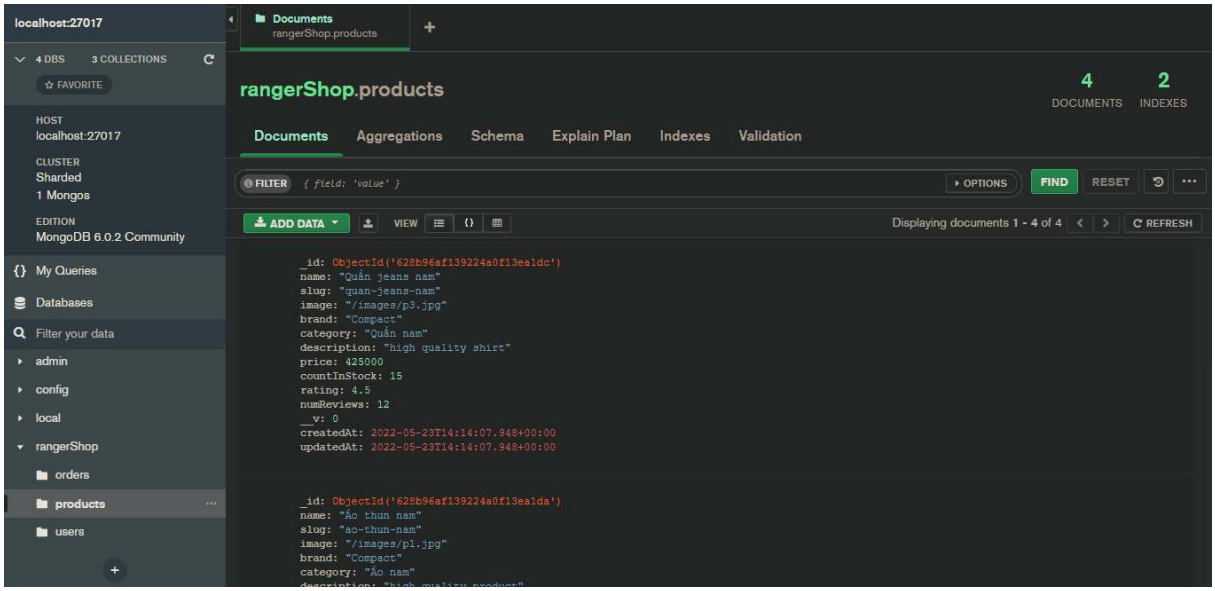
- Tạo sharded cluster và thiết lập xác thực, tạo tài khoản, phân quyền
- Tạo cơ sở dữ liệu, thêm xác thực dữ liệu
- Nhập dữ liệu từ file JSON
- Thiết kế ứng dụng (truy xuất, thêm, xóa, sửa các collection users, products, orders)
- Xuất và backup dữ liệu

4.3. Kết quả demo

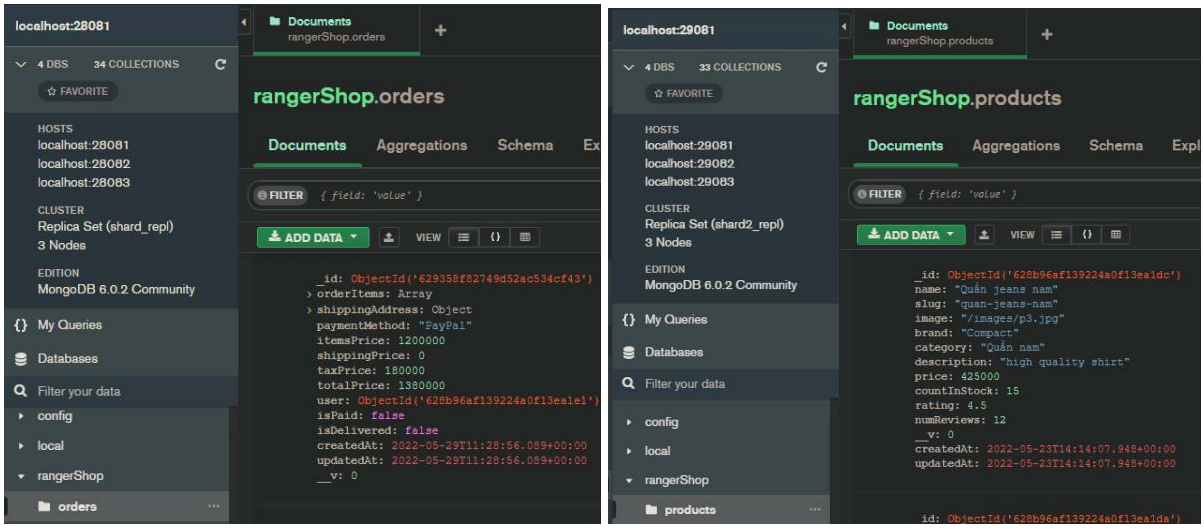
Link Demo:

https://drive.google.com/file/d/1PM7t07qmx4JeJPenqGWfoaYh6uFThohR/view?usp=sharing

Chi tiết các collection và document của cơ sở dữ liệu rangerShop trong sharded cluster:

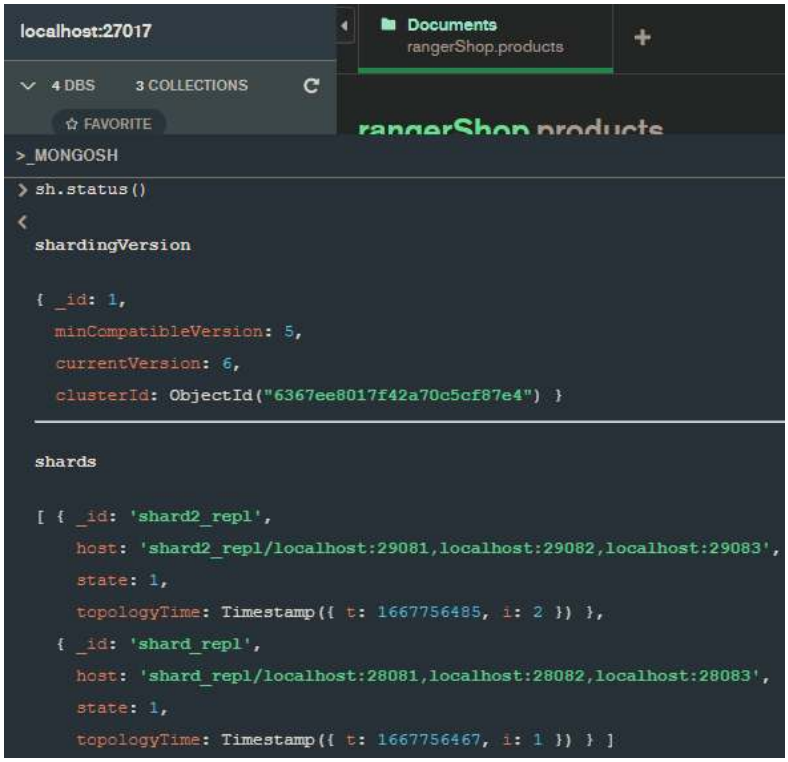


Hình 4.1: Collection products trong cơ sở dữ liệu rangerShop

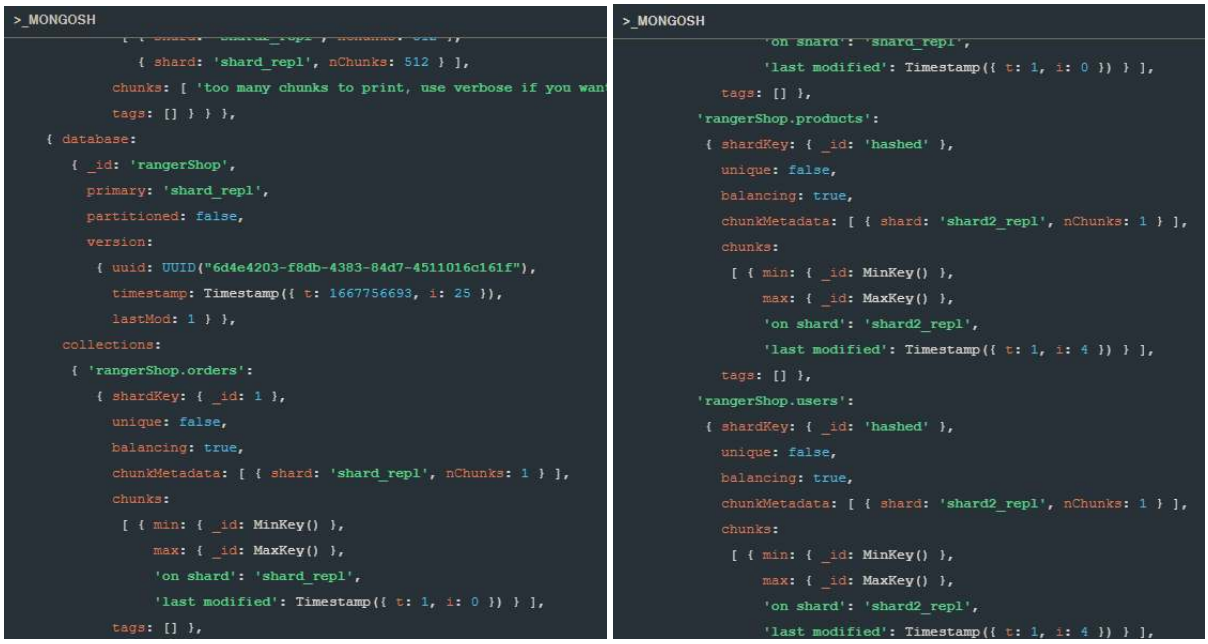


Hình 4.2: Collection orders và users trong cơ sở dữ liệu rangerShop

Sử dụng phương thức rs.status() để xem thông tin sharded cluster và các shard được kết nối trong sharded cluster

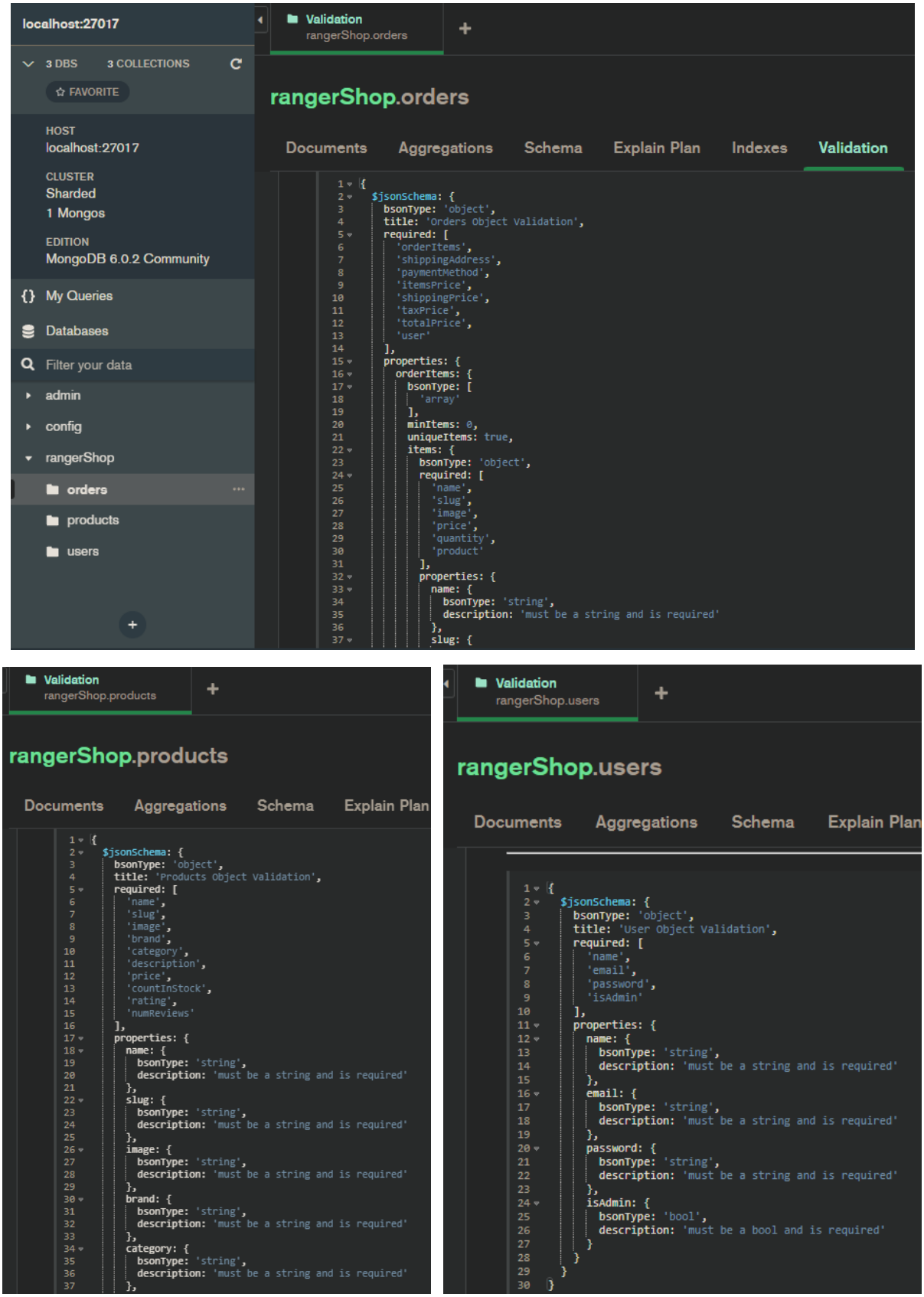


Hình 4.3: Sharded Cluster gồm 2 shard là 2 replica set có tên định danh shard_repl và shard2_repl



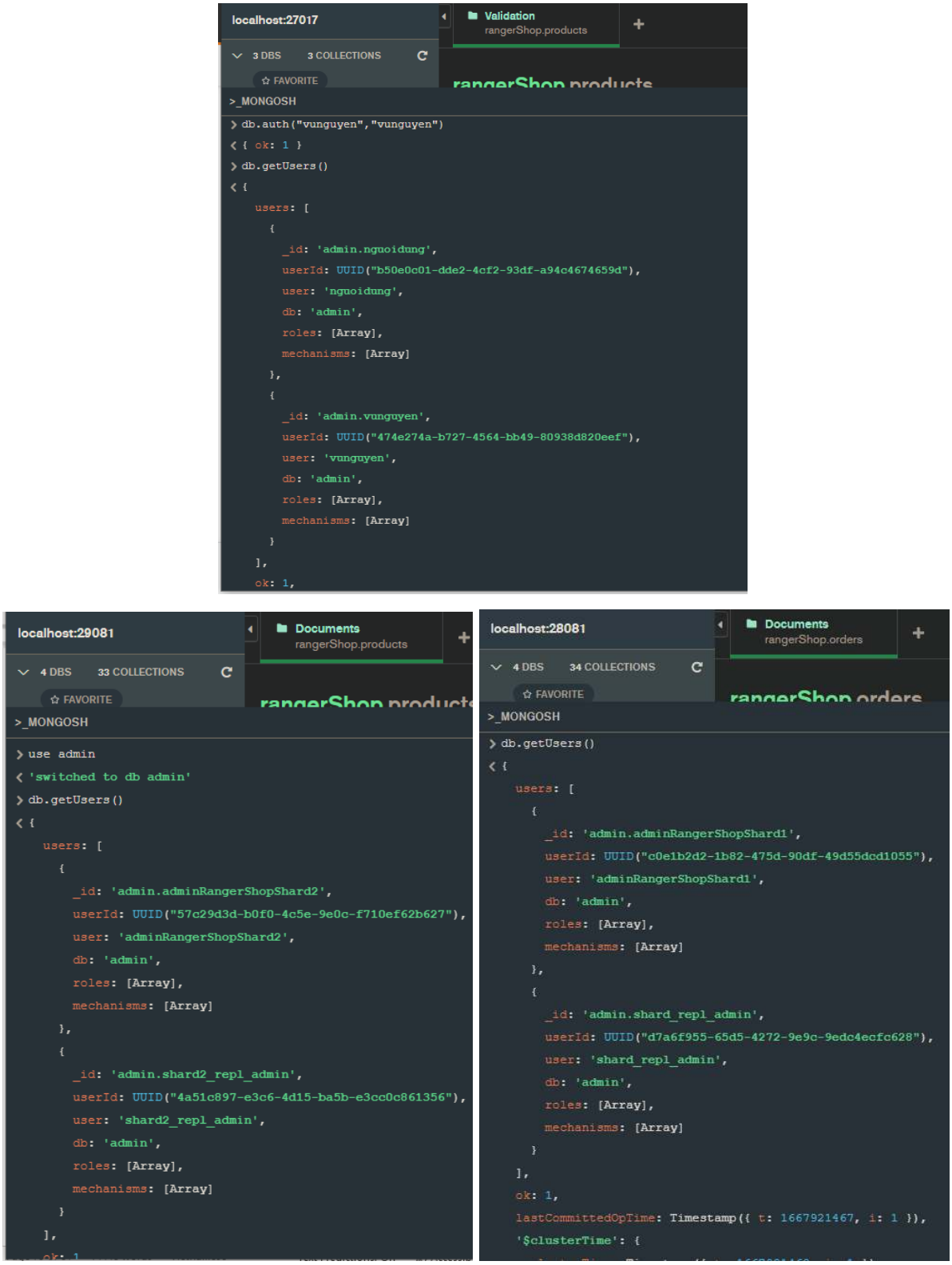
Hình 4.4: Sharded Cluster chứa các collection đã được phân tán theo index là orders, products, users

Dùng tính năng xác thực dữ liệu (validation) trên các collection bằng JSON schema để giúp cấu trúc dữ liệu được thống nhất. Kiểm tra thông tin xác thực mô hình dữ liệu hông qua tab Validation của từng collection trong MonggoDB Compass



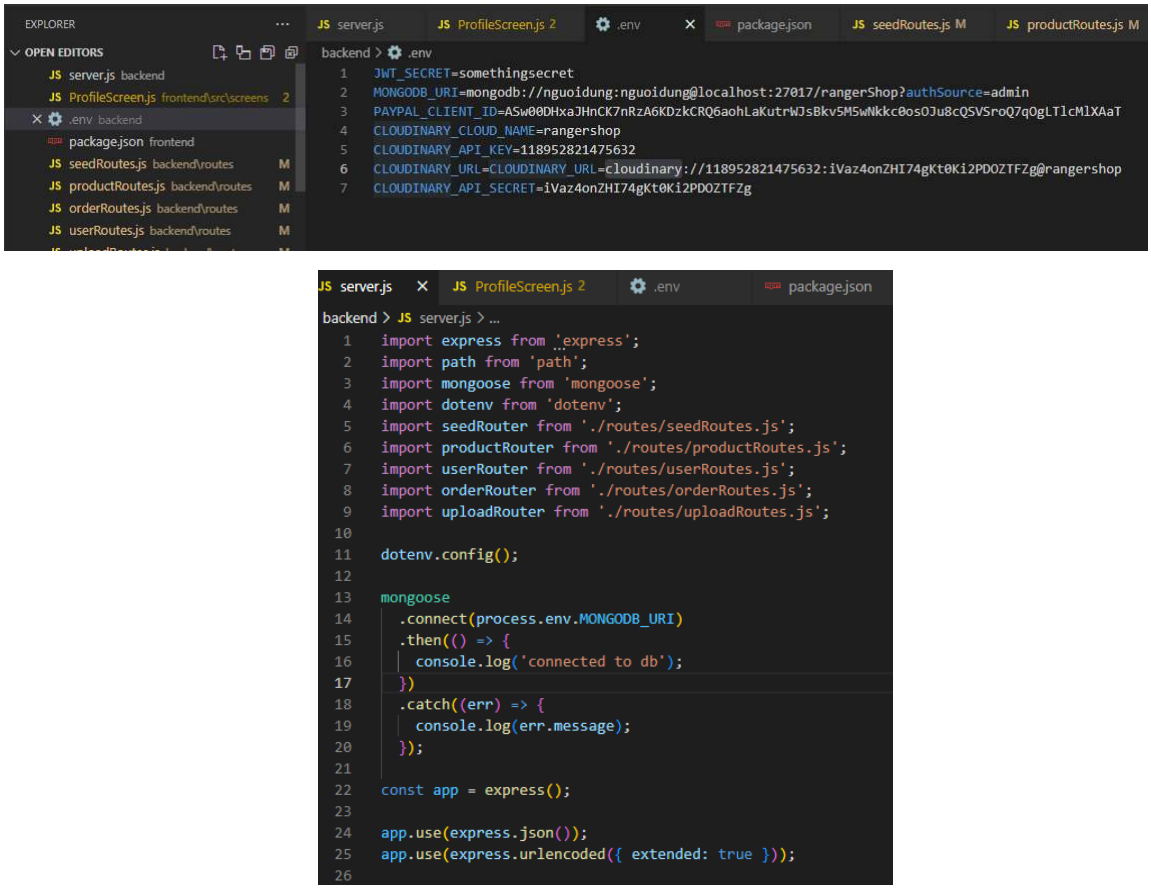
Hình 4.5: Quy định xác thực dữ liệu trong các collection trong cơ sở dữ liệu rangerShop

Dùng phương thức db.getUSers() để lấy thông tin người dùng và phân quyền trên mongos và trên các shard thành phần



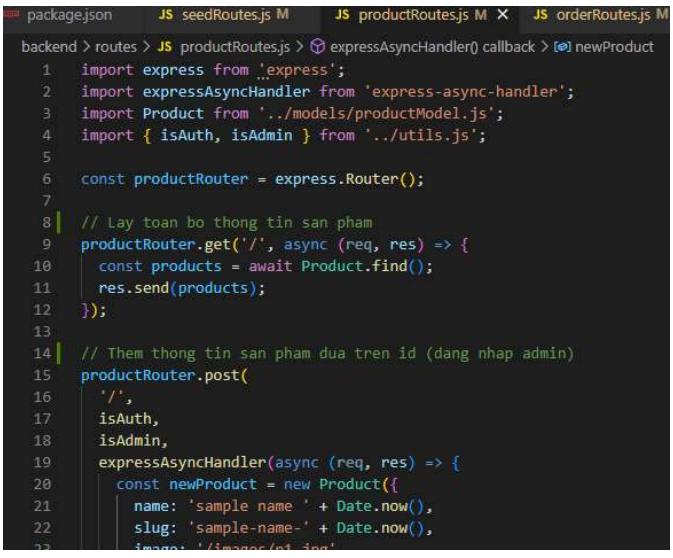
Hình 4.6: Kiểm tra thông tin người dùng và phân quyền trên mongos và trên các shard thành phần

Xây dựng backend ứng dụng website rangerShop kết nối với cơ sở dữ liệu. Kết nối backend với shard cluster thông qua URI kèm thông tin xác thực

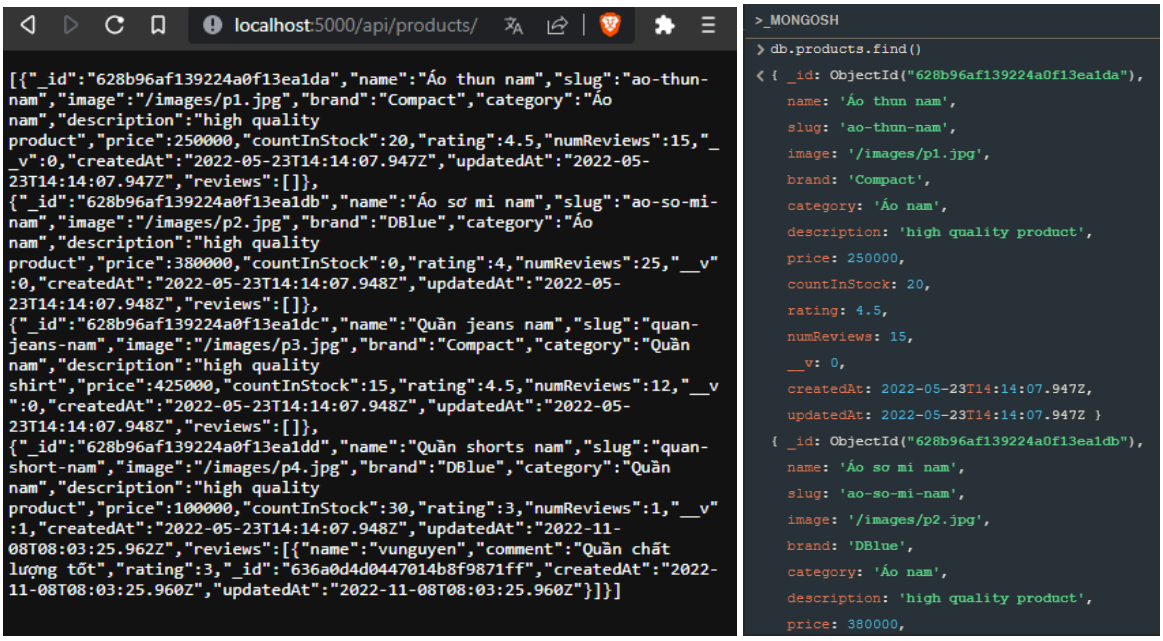


Hình 4.7: Dùng mongoose.connect() kết nối backend với cơ sở dữ liệu rangerShop kèm thông tin xác thực

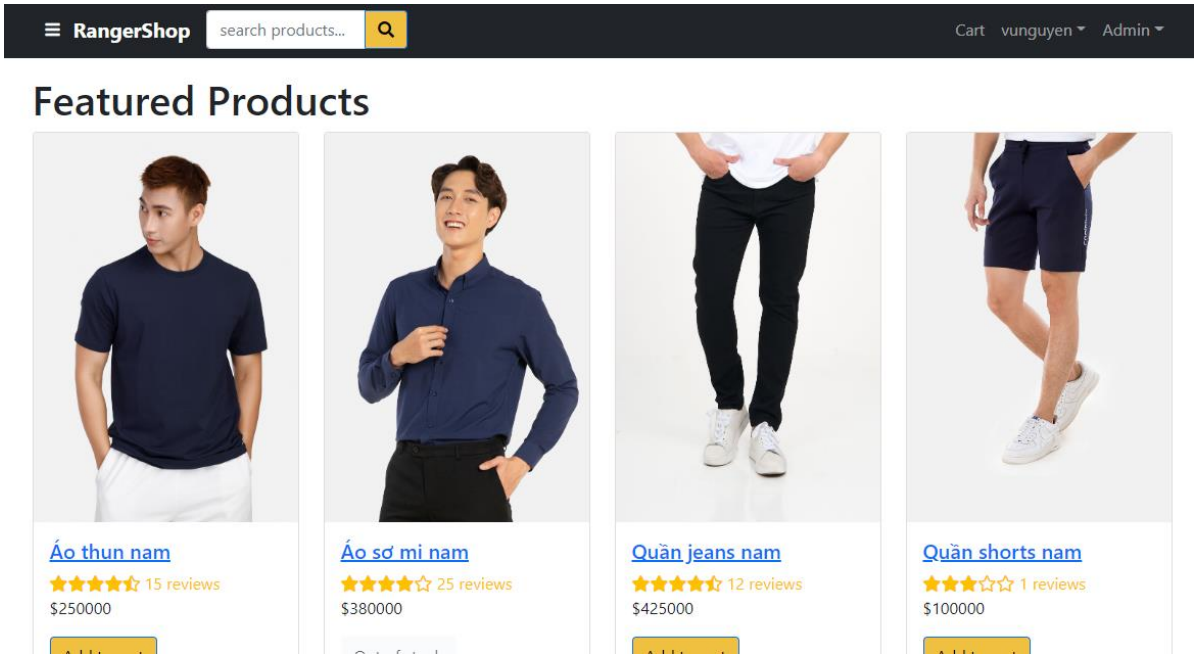
Lập trình backend (productRoutes.js) lấy thông tin sản phẩm từ MongoDB cho ra kết quả giống như khi chạy lệnh truy vấn trong MongoShell. Frontend lấy dữ liệu đó hiển thị thông tin lên website



Hình 4.8: Thiết kế phương thức GET trong productRouter lấy toàn bộ thông tin sản phẩm trong cơ sở dữ liệu

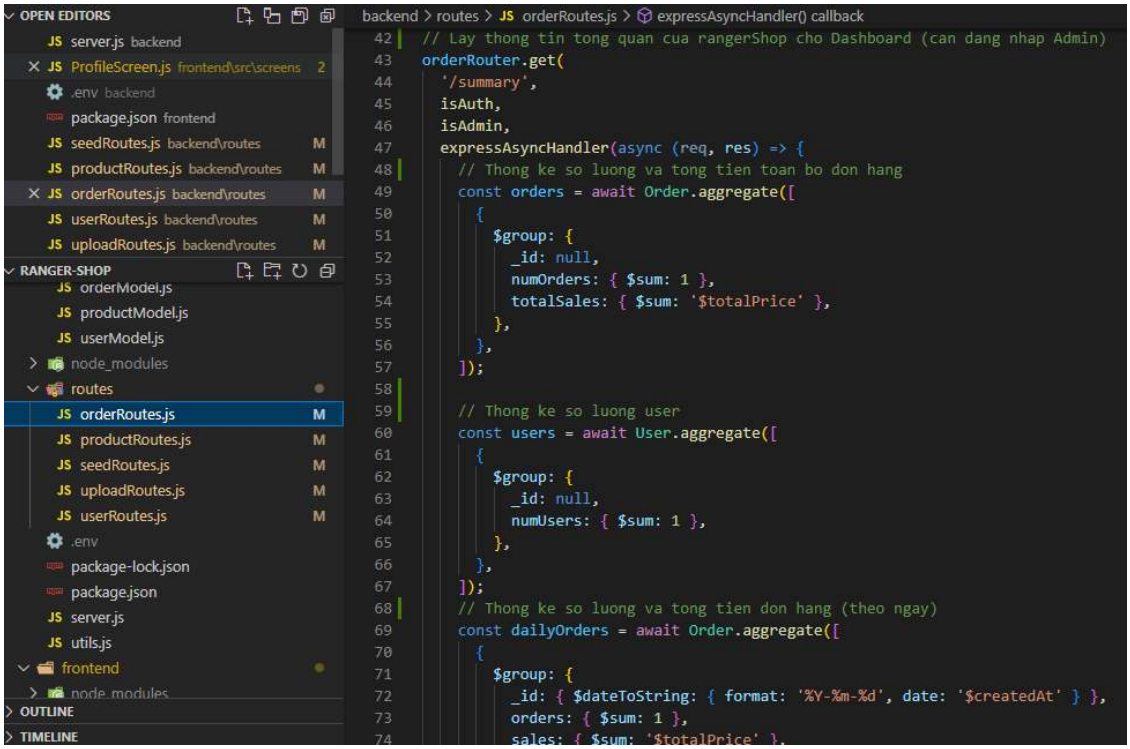


Hình 4.9: Kết quả phản hồi của backend khi yêu cầu đưa tất thông tin sản phẩm trong cơ sở dữ liệu, so sánh với kết quả truy vấn khi dùng phương thức trong Mongosh



Hình 4.10: Hiển thị sản phẩm lên frontend

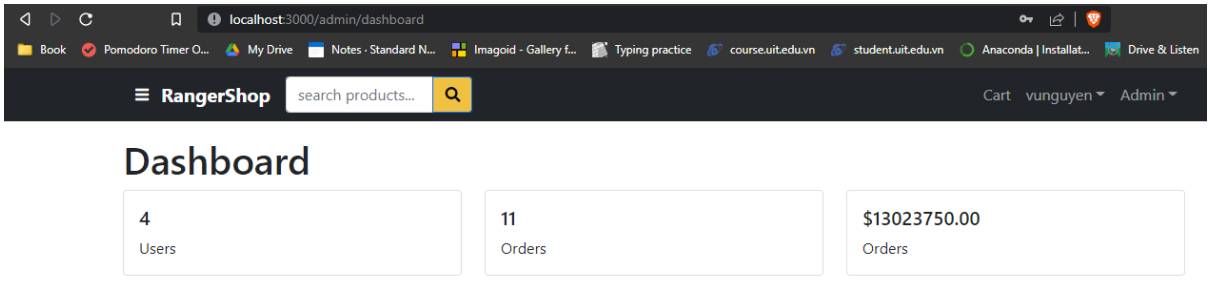
Sử dụng tính năng Aggregation Operations của MongoDB: Lập trình backend (file productRoutes.js) lấy thông tin tổng quan về số lượng và tổng tiền từ các đơn hàng bằng cách dùng Aggregation Operation của MongoDB



Hình 4.11: Thiết kế phương thức GET trong orderRouter lấy thông tin tổng quan dữ liệu trong collection orders

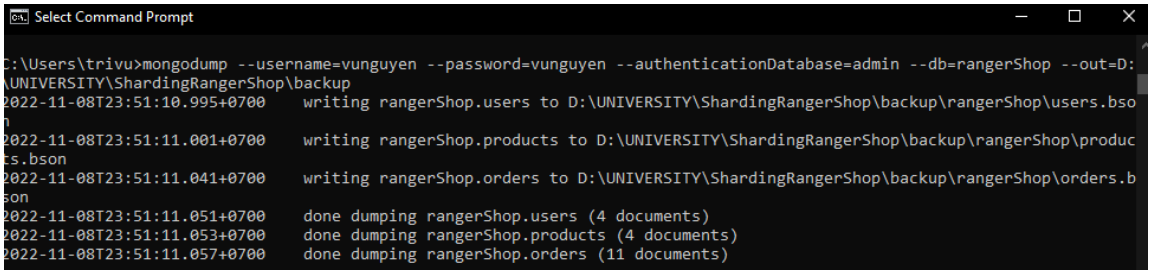
```
>_MONGOSH
> db.orders.aggregate( {$group: {
            _id: null,
            numOrders: { $sum: 1 },
            totalSales: { $sum: '$totalPrice' }
          }})
< { _id: null, numOrders: 11, totalSales: 13023750 }
[mongos] rangerShop>
```

Hình 4.12: Sử dụng phương thức db.collections.aggregate() để xuất thông tin số lượng đơn hàng và tổng giá trị tính được trong collection orders

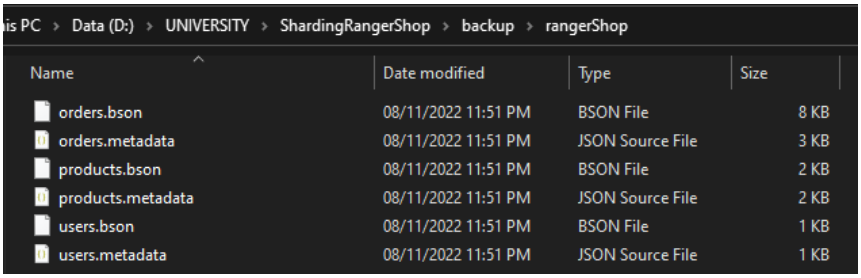


Hình 4.13: Hiển thị các thông số lên giao diện website của admin

Sao lưu và khôi phục dữ liệu của cơ sở dữ liệu rangerShop

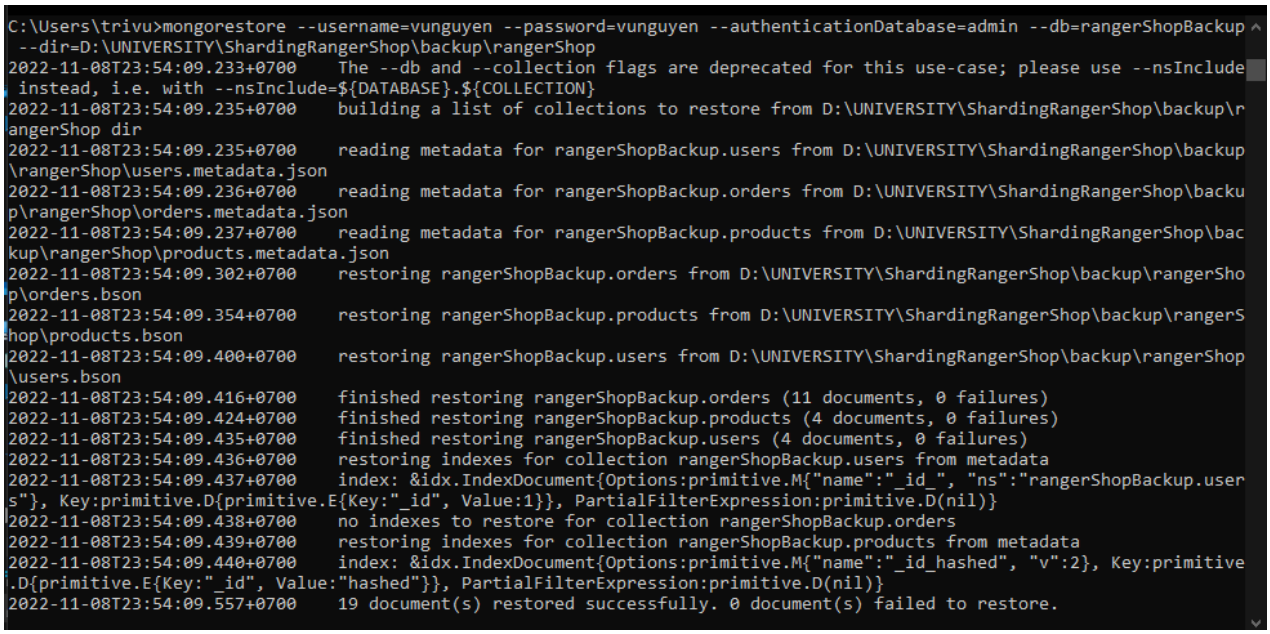


Hình 4.14: Dùng mongodump sao lưu dữ liệu của cơ sở dữ liệu rangerShop

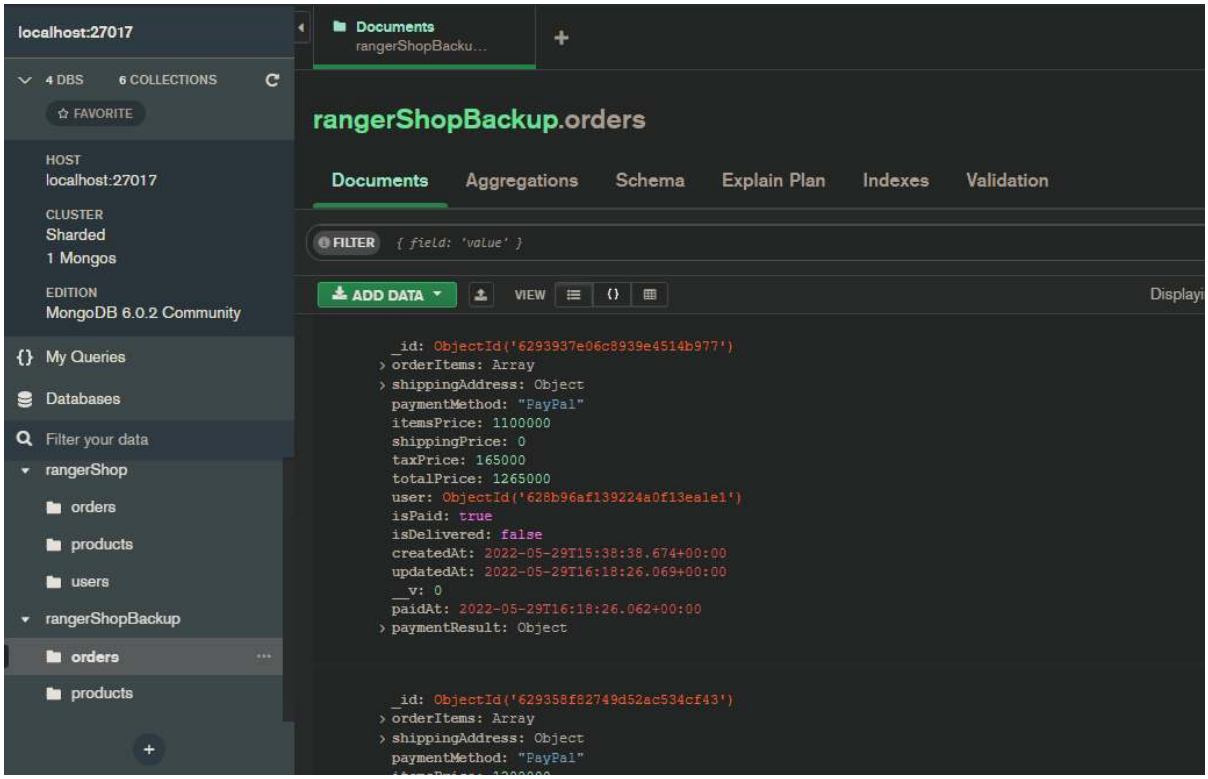


Hình 4.15: Các file sao lưu cơ sở dữ liệu rangerShop được tạo ra

Khôi phục lại cơ sở dữ liệu rangerShop, đặt tên mới là rangerShopBackup. Toàn bộ dữ liệu và các xác thực dữ liệu được khôi phục.

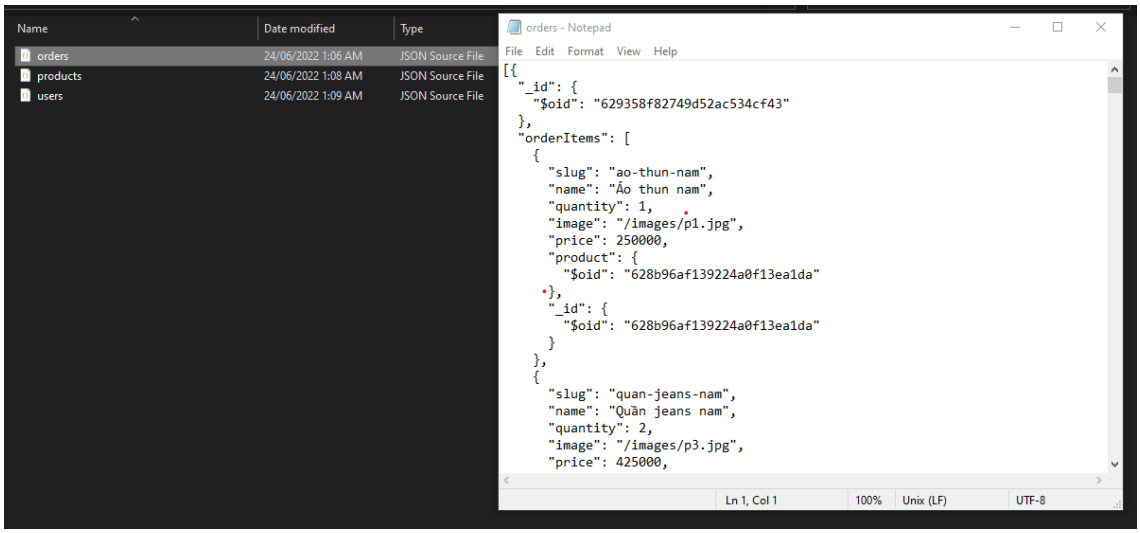


Hình 4.16: Dùng mongorestore khôi phục lại cơ sở dữ liệu rangerShop từ bản sao lưu



Hình 4.17: Kết quả khôi phục lại cơ sở dữ liệu bằng mongorestore

Xuất dữ liệu của cơ sở dữ liệu rangerShop ra file JSON



Hình 4.18: File JSON của các collection trong cơ sở dữ liệu rangerShop

CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1. Kết luận

5.1.1. Ưu điểm

- Đã nêu các thông tin cơ bản về MongoDB
- Đề cập các tính năng cơ bản: mô hình và cấu trúc dữ liệu, các phương thức CRUD trong MongoDB, tính năng Aggregation Operation
- Tìm hiểu được một số tính năng nâng cao: Phân quyền xác thực, bảo mật, Index, Replica Set, Sharded Cluster, sao lưu và khôi phục, nhập xuất dữ liệu
- Sử dụng các tính năng tìm hiểu được trên một ứng dụng trên môi trường localhost

5.1.2. Nhược điểm

- Chưa tìm hiểu các phương thức bảo mật khác như x509, TLS/SSL. Chưa đi sâu vào một số tính năng trên MongoDB như change streams, time series, transactions
- Demo cần chỉnh sửa về tổ chức dữ liệu và hoàn thiện hơn về giao diện
- Chưa đi sâu vào các phương pháp sao lưu như Filesystem Snapshots hoặc MongoDB Cloud Manager

5.2. Hướng phát triển

- Khắc phục các nhược điểm đề cập ở trên trên
- Tiến hành triển khai sharded cluster trên các máy chủ khác nhau thay vì tổ chức trên local
- Nghiên cứu cài đặt index để tối ưu xử lý khi phân tán dữ liệu cho các shard trong sharded cluster.

TÀI LIỆU THAM KHẢO

[1] MongoDB Manual: https://www.mongodb.com/docs/manual/

[2] Demo: https://drive.google.com/file/d/1PM7t07qmx4JeJPenqGWfoaYh6uFThohR/view?usp=sharing