8. Bài Tập Tuần 08

Bài 1: Sử dụng cơ sở dữ liệu restaurants thực hiện các thao tác sau đây:

- **Kiểm tra truy vấn trước khi tạo chỉ mục**
  1. Liệt kê danh sách restaurant có cuisine là “Hamburger” và sử dụng phương thức explain("executionStats") để kiểm tra hiệu suất truy vấn *(số lần duyệt, thời gian thực thi, cách duyệt,…)*.
- **Làm việc với Index**
  2. Tạo một index đơn trên trường “cuisine” theo thứ tự tăng dần;
  3. Liệt kê lại danh sách restaurant có cuisine là “Hamburger” và sử dụng phương thức explain("executionStats"), so sánh với kết quả câu 1;
  4. Liệt kê tất cả các index hiện có trong collection restaurants, bao gồm cả tên index;
  5. Xóa tất cả index hiện có trong restaurants;
  6. Tạo lại index trên trường “cuisine” theo thứ tự giảm dần, đặt tên là cuisine_desc. Sau đó kiểm tra lại index vừa tạo.
- **Tối ưu hóa truy vấn với Index**
  7. Truy vấn danh sách restaurant có “borough” là “Brooklyn” khi chưa có chỉ mục. Sau đó, tạo chỉ mục trên “borough”, truy vấn lại danh sách và so sánh hiệu suất;
  8. Tạo một compound index trên “cuisine” *(giảm dần)* và “name” *(tăng dần)*, sau đó liệt kê lại các index hiện có;
  9. Liệt kê danh sách restaurants có “cuisine” là "Hamburger" và “name” bắt đầu bằng "Wil"`.
- **Làm việc với Unique Index**
  10. Tạo một Unique Index trên trường “name”;
  11. Chèn một document mới có “name” trùng với một restaurant đã có và nhận xét kết quả;
  12. Ẩn Unique Index vừa tạo, sau đó thử chèn một document có “name” trùng lặp. Nhận xét sự khác biệt so với câu 11.
- **Làm việc với Embedded Index**
  13. Tạo một Embedded Field Index trên trường “address.zipcode” *(tăng dần)*;

14. Truy vấn danh sách restaurant có "zipcode" lớn hơn 11000, kết hợp explain("executionStats") để xem trạng thái thực thi.

- **Xóa và tạo lại Index**

15. Xóa tất cả các Index hiện có trong restaurants;
16. Tạo index trên "name" *(tăng dần)* và "address.zipcode" *(tăng dần)*;
17. Liệt kê danh sách restaurant có "zipcode" lớn hơn 11000, sắp xếp theo tên tăng dần, kết hợp explain("executionStats") để kiểm tra truy vấn.

- **Multi-key Index**

18. Tạo Multi-key Index trên trường "grades" *(tăng dần)*;
19. Liệt kê danh sách restaurant có "grades.grade" là "A", sử dụng explain("executionStats");
20. Tạo Multi-key Index trên trường "grades.score" *(tăng dần)*;
21. Liệt kê danh sách restaurant có "grades.score" từ 5 đến 10, sử dụng explain("executionStats").

Bài 2: Tìm hiểu vai trò và cách sử dụng index trong cơ sở dữ liệu

1. Tạo một cơ sở dữ liệu **highestmountains**, với một collection **peaks** gồm các document chứa các thông tin:
   - **name**: tên của đỉnh núi.
   - **height**: độ cao của đỉnh *(mét)*.
   - **location**: mảng chứa danh sách các quốc gia nơi ngọn núi tọa lạc. Cho phép các ngọn núi nằm ở nhiều quốc gia.
   - **ascents**: embedded document. Chưa thông tin số lần leo núi thành công:
     - ***total***: tổng số lần leo núi thành công.
     - ***first***: là một embedded document chứa trường ***year***, mô tả năm đầu tiên có người chinh phục.
     - ***first_winter:*** là một embbeded document chứa trường ***year***, mô tả năm đầu tiên có người leo lên vào mùa đông thành công.
2. Chèn dữ liệu vào collection **peaks** với 5 documents sau:

```
db.peaks.insertMany([
    {
        "name": "Everest",
```

```
            "height": 8848,
            "location": ["Nepal", "China"],
            "ascents": {
                "first": { "year": 1953 },
                "first_winter": { "year": 1980 },
                "total": 5656
            }},
        {
            "name": "K2",
            "height": 8611,
            "location": ["Pakistan", "China"],
            "ascents": {
                "first": { "year": 1954 },
                "first_winter": { "year": 1921 },
                "total": 306
            }},
        {
            "name": "Kangchenjunga",
            "height": 8586,
            "location": ["Nepal", "India"],
            "ascents": {
                "first": { "year": 1955 },
                "first_winter": { "year": 1986 },
                "total": 283
            }},
        {
            "name": "Lhotse",
            "height": 8516,
            "location": ["Nepal", "China"],
            "ascents": {
                "first": { "year": 1956 },
                "first_winter": { "year": 1988 },
                "total": 461
            }},
        {
            "name": "Makalu",
            "height": 8485,
            "location": ["China", "Nepal"],
            "ascents": {
                "first": { "year": 1955 },
                "first_winter": { "year": 2009 },
                "total": 361
            }
        }
    ])
```

3. Thực hiện truy vấn tìm ngọn núi có độ cao > 8700m.
4. Kiểm tra hiệu suất truy vấn bằng explain(). Dùng phương thức explain("executionStats") để xem cách MongoDB thực thi truy vấn.

```
db.peaks.find( {"height": {$gt: 8700}}).explain("executionStats")
```

5. Tạo một index *(Single Field Index)* trên trường "height"**.** Kiểm tra số lượng chỉ mục trong collection **peaks** trước và sau khi tạo.

6. Thực hiện lại truy vấn ở câu số 3, dùng phương thức **explain(executionStats**) để xem thông tin về cách thực hiện truy vấn, so sánh hiệu suất.
7. Tạo một index *(Unique Indexes)* đảm bảo tên các đỉnh núi "name" là duy nhất. Viết câu lệnh xem kết quả của Index vừa tạo.

```
db.peaks.createIndex( { "name": 1 }, { "unique": true } )
Output
{
        "createdCollectionAutomatically": false,
        "numIndexesBefore" : 2,
        "numIndexesAfter" : 3,
        "ok": 1
}
```

8. Thực hiện câu lệnh truy vấn tìm đỉnh **Everest** và kiểm tra hiệu xuất của câu truy vấn.
9. Kiểm tra tính Unique của chỉ mục: chèn thêm một document có name là "Everest" để kiểm tra tính ràng buộc của chỉ mục Unique và nhận xét kết quả.
10. Tạo một index *(Embedded Field)* trên trường mảng
    a. Thực hiện truy vấn tìm những ngọn núi có "total" > 300, sắp xếp giảm dần;

    ```
    db.peaks.find({"ascents.total": {$gt:300}})
    .sort({"ascents.total": -1}).explain("executionStats")
    ```

    b. Thực hiện lại truy vấn trên nhưng kết hợp với phương thức explain("executionStats") để xem tình trạng thực thi của câu truy vấn (*giá trị COLLSCAN trong phần này của đầu ra cho biết, MongoDB đã sử dụng quét toàn bộ bộ sưu tập và duyệt qua tất cả các tài liệu từ bộ sưu tập đỉnh để so sánh chúng với các điều kiện truy vấn*);
    c. Tạo một index trên field "ascents.total";
    d. Chạy lại truy vấn ở mục "a" và phân tích hiệu suất (*IXSCAN được sử dụng đối với chỉ mục ascents.total_-1 mới được tạo và chỉ có bốn tài liệu đã được kiểm tra. Đây là cùng một số lượng tài liệu được trả lại và kiểm tra trong chỉ mục, vì vậy không có tài liệu bổ sung nào được truy xuất để hoàn thành truy vấn*).
11. **Compound Index**: sử dụng chỉ mục khi thực hiện các truy vấn trên nhiều trường.
    a. Tìm những ngọn núi có "height" < 8600m và "ascents.first_winter.year" > 1990, sắp xếp theo "height" *(khi chưa tạo index)*. Xem cách mongodb thực thi truy vấn;
    b. Tạo compound index trên field "height" và "ascents.first_winter.year";

c. Kiểm tra lại hiệu suất khi thực hiện lại truy vấn ở mục "a" sau khi tạo index.

12. **Multi-key Index:** khi trường được sử dụng để tạo chỉ mục là trường có kiểu dữ liệu mảng.

a. Viết câu lệnh tìm tất cả các ngọn núi ở **Nepal**;

Nhận xét: Mỗi đỉnh bao trùm nhiều quốc gia như được biểu thị bằng các trường location, là một mảng gồm nhiều giá trị. Do không có sẵn chỉ mục mở rộng trường ***location****, MongoDB hiện thực hiện quét toàn bộ bộ collection để thực hiện truy vấn.*

b. Tạo một index mới cho trường **location**;

c. Kiểm tra lại hiệu suất sau khi tạo index.

```
db.peaks.find( {"location": "Nepal"} ).
explain("executionStats")
```

Nhận xét kết quả

- *MongoDB đã sử dụng quét chỉ mục ("stage" : "IXSCAN",) làm chiến lược, đề cập đến chỉ mục location_1 mới được tạo*
- *Thuộc tính* ***isMultiKey****: true. MongoDB tự động tạo một multi-key index cho trường location.*
- *Đối với document có trường* ***location*** *lưu trữ array ["China", "Nepal"], hai index entries riêng biệt sẽ xuất hiện cho cùng một tài liệu, một cho China và một cho Nepal. Bằng cách này, MongoDB có thể sử dụng chỉ mục một cách hiệu quả ngay cả khi truy vấn yêu cầu khớp một phần với nội dung mảng.*