Bài giảng: Views trong MongoDB 8

Author: Đặng Kim Thi

---

1. Giới thiệu Views trong MongoDB

1.1. Views là gì?

- **View** trong MongoDB là một tập hợp ảo (chỉ đọc) được tạo dựa trên một truy vấn Aggregation Pipeline.
- Views không lưu trữ dữ liệu thật, mà tự động lược dữ liệu từ collection gốc.

1.2. Đặc điểm

- Views là **read-only** (chỉ đọc).
- Hỗ trợ phép biến đổi dữ liệu sử dụng Aggregation Pipeline.
- Giúp trực quan hóa dữ liệu phức tạp và tăng tính tự động.
- Hạn chế **sự trùng lặp dữ liệu** bằng cách chỉ hiển thị dữ liệu cần thiết mà không cần lưu trữ lại dữ liệu.

1.3. Lưu ý

- Views không tăng tính hiệu quả cho truy vấn nếu pipeline quá phức tạp.
- Chỉ hỗ trợ đọc dữ liệu.

---

2. Cách tạo Views

2.1. Tạo View cơ bản

Cú pháp tạo View:

```
// db.createView(viewName, source, pipeline)
db.createView(
    "view_name",       // Tên View
    "source_collection", // Collection gốc
    [pipeline]         // Aggregation pipeline
)
```

VD: Tạo một View hiển thị những khách hàng có điểm tích lũy trên 1000:

1. Tạo dữ liệu gốc ban đầu:

```
db.customers.insertMany([
    { name: "Alice", loyaltyPoints: 1500 },
    { name: "Bob", loyaltyPoints: 800 },
```

```
    { name: "Charlie", loyaltyPoints: 1200 }
]);
```

2. Tạo View:

```
db.createView(
    "high_value_customers",  // Tên View
    "customers",             // Collection gốc
    [
        { $match: { loyaltyPoints: { $gt: 1000 } } }
    ]
);
```

2.2. Tạo View phức tạp

Pipeline có thể chứa nhiều bước biến đổi:

1. Tạo dữ liệu gốc:

```
db.sales.insertMany([
    { saleDate: new Date("2023-01-15"), amount: 200 },
    { saleDate: new Date("2023-02-20"), amount: 500 },
    { saleDate: new Date("2023-01-25"), amount: 300 }
]);
```

2. Tạo View:

```
db.createView(
    "monthly_sales_summary",
    "sales",
    [
        { $group: {
            _id: { month: { $month: "$saleDate" }, year: { $year:
"$saleDate" } },
            totalSales: { $sum: "$amount" },
            totalOrders: { $sum: 1 }
        } },
        { $sort: { "_id.year": 1, "_id.month": 1 } }
    ]
);
```

---

3. Quản lý Views

3.1. Hiển thị Views

Danh sách Views trong database:

```
db.getCollectionInfos({ type: "view" });
```

3.2. Xóa View

Xóa View bằng lệnh:

```
db.view_name.drop();
```

3.3. Sửa Views

Không thể sửa View trực tiếp. Cách duy nhất:

1. Xóa View cũ.
2. Tạo View mới.

---

4. Sử dụng View để lấy dữ liệu

4.1. Lấy toàn bộ dữ liệu từ View

Giả sử bạn có View tên là `expensive_products`:

```
db.expensive_products.find();
```

4.2. Lọc dữ liệu từ View

Nếu bạn muốn tìm sản phẩm cụ thể trong View `expensive_products`:

```
db.expensive_products.find({ name: "Laptop" });
```

4.3. Chỉ lấy các trường cụ thể với `projection`

Ví dụ: Chỉ lấy tên sản phẩm mà không cần giá:

```
db.expensive_products.find({}, { name: 1, _id: 0 });
```

4.4. Đếm số lượng bản ghi trong View

Dùng `countDocuments()` để đếm số lượng bản ghi trong View:

```
db.expensive_products.countDocuments();
```

4.5. Sắp xếp kết quả từ View

Bạn có thể sắp xếp kết quả khi truy vấn View:

```
db.expensive_products.find().sort({ price: -1 }); // Sắp xếp giá giảm dần
```

4.6. Giới hạn số lượng kết quả

Ví dụ: Lấy 2 sản phẩm đầu tiên trong View:

```
db.expensive_products.find().limit(2);
```

---

5. Ưu điểm và hạn chế

5.1. Ưu điểm

1. **Giảm trùng lặp dữ liệu**:
   - Vì Views chỉ là tập hợp ảo và không lưu trữ dữ liệu thật, bạn không cần phải tạo các bản sao dữ liệu, từ đó giảm việc lưu trữ không cần thiết.
2. **Dễ bảo trì**:
   - Khi thay đổi yêu cầu truy vấn, bạn chỉ cần cập nhật pipeline của View thay vì phải thay đổi hoặc sao lưu dữ liệu thực tế.
3. **Bảo mật dữ liệu**:
   - Views cho phép hiển thị dữ liệu được lọc hoặc chỉ một phần của collection gốc, hạn chế quyền truy cập dữ liệu nhạy cảm.
4. **Phù hợp cho trực quan hóa**:
   - Views rất hữu ích trong việc chuẩn bị dữ liệu đầu vào cho các công cụ phân tích hoặc báo cáo.

5.2. Hạn chế

1. **Hiệu suất truy vấn thấp hơn**:
   - Nếu pipeline của View quá phức tạp, hiệu suất sẽ bị ảnh hưởng do MongoDB phải xử lý lại pipeline mỗi khi View được truy vấn.
2. **Không hỗ trợ ghi dữ liệu**:
   - Views chỉ hỗ trợ đọc dữ liệu, không thể sử dụng để chèn (`insert`), cập nhật (`update`) hoặc xóa (`delete`).
3. **Không sử dụng được index trực tiếp**:
   - Index chỉ áp dụng trên collection gốc, không áp dụng trực tiếp trên View, điều này có thể làm giảm hiệu suất nếu View xử lý lượng dữ liệu lớn.

---

6. Khi nào nên dùng và không nên dùng Views

6.1. Khi nào nên dùng

1. **Đơn giản hóa truy vấn lặp lại**:
   - Nếu bạn thường xuyên cần chạy cùng một pipeline Aggregation, Views giúp bạn giảm mã lệnh và chuẩn hóa truy vấn.
   - **Ví dụ**: Hiển thị doanh thu theo tháng:

```
db.createView(
    "monthly_sales_summary",
    "sales",
    [
        { $group: {
            _id: { month: { $month: "$saleDate" }, year: { $year:
"$saleDate" } },
            totalRevenue: { $sum: "$amount" }
        } }
    ]
);
```

2. **Ẩn dữ liệu nhạy cảm**:
   - Chỉ hiển thị các trường cụ thể thay vì toàn bộ dữ liệu.
   - **Ví dụ**: Hiển thị tên và email khách hàng:

```
db.createView(
    "public_customers",
    "customers",
    [
        { $project: { name: 1, email: 1, _id: 0 } }
    ]
);
```

3. **Chuẩn bị dữ liệu cho báo cáo**:
   - Dùng Views làm nguồn dữ liệu đầu vào cho các công cụ BI hoặc dashboard.

4. **Dữ liệu tạm thời hoặc kiểm tra nhanh**:
   - Tạo View để phân tích dữ liệu trong thời gian ngắn mà không thay đổi collection gốc.

6.2. Khi nào không nên dùng

1. **Pipeline quá phức tạp**:
   - Nếu View có nhiều bước phức tạp như `$lookup`, `$unwind`, hiệu suất sẽ giảm vì MongoDB phải xử lý lại mỗi lần truy vấn.

- **Giải pháp**: Lưu kết quả pipeline vào một collection mới để truy vấn nhanh hơn.

2. **Truy vấn thường xuyên với dữ liệu lớn**:
   - Nếu View được truy vấn thường xuyên, việc xử lý pipeline lặp lại sẽ làm giảm hiệu suất.
   - **Giải pháp**: Sử dụng cron job để cập nhật Materialized View trong một collection thực.
3. **Cần index trên dữ liệu kết quả**:
   - View không hỗ trợ index trực tiếp. Nếu cần index, hãy lưu kết quả của View vào collection thực tế.

6.3. So sánh hiệu quả giữa dùng và không dùng Views

Tiêu chí	Dùng Views	Không dùng Views
Hiệu suất	Tốt với pipeline đơn giản	Tốt hơn với pipeline phức tạp
Tính bảo trì	Dễ bảo trì, không cần thay đổi dữ liệu gốc	Cần lưu và quản lý collection thủ công
Tính bảo mật	Ẩn dữ liệu nhạy cảm	Phải viết thêm logic để lọc dữ liệu
Trường hợp phức tạp	Giảm hiệu suất	Tối ưu nếu lưu pipeline vào collection

7. Thực hành: Hướng dẫn chi tiết

7.1. Bài tập 1: Tạo View hiển thị những sản phẩm có giá trên 500

1. Mở **mongosh**:

```
mongosh
```

2. Kết nối với database:

```
use your_database_name;
```

3. Tạo dữ liệu gốc:

```
db.products.insertMany([
    { name: "Laptop", price: 1000 },
    { name: "Mouse", price: 20 },
    { name: "Monitor", price: 300 }
]);
```

4. Tạo View:

```
db.createView(
    "expensive_products",  // Tên View
    "products",            // Collection gốc
    [
        { $match: { price: { $gt: 500 } } },
        { $sort: { name: 1 } }
    ]
);
```

7.2. Bài tập 2: Tạo View tóm tắt doanh thu theo tháng

1. Pipeline:

```
db.createView(
    "monthly_revenue_summary",
    "sales",
    [
        { $group: {
            _id: { month: { $month: "$saleDate" }, year: { $year:
"$saleDate" } },
            totalRevenue: { $sum: "$amount" }
        } },
        { $sort: { "_id.year": 1, "_id.month": 1 } }
    ]
);
```

7.3. Bài tập 3: Hiển thị Views

1. Danh sách Views:

```
db.getCollectionInfos({ type: "view})
```