Giải thích demodb3_mongodb

17 Tháng Mười 2025 9:07 CH

1) Các lệnh cơ bản trước khi aggregate

`use("demodb3")`

- **Định nghĩa/Ý nghĩa:** Chọn (switch) database "demodb3" trong `mongosh`. Nếu DB chưa tồn tại, nó sẽ được tạo khi có collection/tài liệu đầu tiên được ghi.
- **Cú pháp:** `use("<dbName>")`
- **Cách dùng:** Gọi trước khi thao tác collection trong DB đó.

`db.createCollection("products")`

- **Định nghĩa/Ý nghĩa:** Tạo collection trống có tên `products`.
- **Cú pháp:** `db.createCollection("<name>", <options?>)`
- **Cách dùng:** Tuỳ chọn; MongoDB sẽ tự tạo collection khi bạn `insert` lần đầu.

`db.products.insertMany([...])`

- **Định nghĩa/Ý nghĩa:** Chèn nhiều document vào collection.
- **Cú pháp:** `db.<coll>.insertMany([ {doc1}, {doc2}, ... ], <options?> )`
- **Cách dùng:** Truyền mảng các object. Ở đây mỗi document có `product`, `category`, `price`, `quantity`.

Lưu ý: Kiểu dữ liệu số (`price`, `quantity`) phải là số để các toán tử như `$multiply`, `$sum`, `$avg` hoạt động chính xác.

2) Kiến thức cốt lõi về Aggregation Pipeline

- **Khái niệm:** Aggregation pipeline là chuỗi các "stage" biến đổi luồng document. Mỗi stage nhận vào stream document từ stage trước và phát ra stream mới.
- **Cú pháp tổng quát:**

`db.<collection>.aggregate([ { <stage1>: { ... } }, { <stage2>: { ... } }, ... ])`

- **Quy tắc:** Thứ tự stage rất quan trọng (ví dụ: `$match` sớm để giảm dữ liệu, rồi mới `$group`/`$sort`).

3) Giải thích từng stage/toán tử bạn dùng

3.1 `$match`

- **Định nghĩa/Ý nghĩa:** Lọc document giống như `find` (lọc theo điều kiện).
- **Cú pháp:**

`{ $match: { <field>: <value>, <field2>: { <operator>: <value> }, ... } }`

- **Ví dụ của bạn:**

```
db.products.aggregate([
    {$match: {
      category:"Stationery"
    }}
])
```

→ chỉ giữ lại sản phẩm thuộc nhóm "Stationery".

- **Lưu ý:** Đặt `$match` càng sớm càng tốt để tối ưu hiệu năng (giảm số doc chuyển qua stage sau). Nếu có index phù hợp, `$match` sẽ tận dụng.

3.2 $group

- **Định nghĩa/Ý nghĩa:** Gom nhóm document theo `_id` nhóm, đồng thời tính toán (sum/avg/min/max/count…) trên mỗi nhóm.
- **Cú pháp:**
  `{ $group: { _id: <expression>, // khoá nhóm (thường là "$<field>") newField1: { <accumulator>: <expression> }, newField2: { <accumulator>: <expression> }, ... } }`
- **Ví dụ của bạn (đúng):**
  `{ $group: { _id: "$category", totalSale: { $sum: { $multiply: ["$price", "$quantity"] } }, totalQty: { $sum: "$quantity" } } }`
- `_id: "$category"`: nhóm theo category.
- `totalSale`: tổng doanh thu = sum(price × quantity).
- `totalQty`: tổng số lượng.
- **Lỗi thường gặp bạn có trong 1 chỗ:**
  `$group: { _id: "category", ... } // SAI`
  Phải là `_id: "$category"` (có $) để lấy **giá trị** của field `category`, không phải chuỗi literal `"category"`.
- **Các accumulator phổ biến:** `$sum`, `$avg`, `$min`, `$max`, `$count` (trong 5.0+), v.v.

3.3 $project

- **Định nghĩa/Ý nghĩa:** Chọn/trả về các trường mong muốn, đồng thời có thể tạo trường tính toán.
- **Cú pháp:**
  `{ $project: { _id: 0/1, fieldA: 1, fieldB: { <expression> }, // ví dụ tính toán ... } }`
- **Ví dụ của bạn:**

```
db.products.aggregate([
    {$project: {
      _id:0,
      product:1,
      total:{$multiply:["$price","$quantity"]}
    }}
])
```

  → ẩn `_id`, giữ `product`, thêm trường tính toán `total`.
- **Lưu ý:** `_id` mặc định là 1 (hiện). Đặt `_id: 0` để ẩn.

3.4 $sort

- **Định nghĩa/Ý nghĩa:** Sắp xếp document theo một hay nhiều trường.
- **Cú pháp:**
  `{ $sort: { field1: 1/-1, field2: 1/-1, ... } }`
- 1 tăng dần, -1 giảm dần.
- **Ví dụ của bạn:**

```
db.products.aggregate([
    {$sort: {
      price:-1
    }}
])
```

`// giảm dần theo price { $sort: { avgPrice: 1 } } // tăng dần theo avgPrice`

- **Lưu ý hiệu năng:** `$sort` trên lượng dữ liệu lớn nên đi sau `$match` để giảm tập dữ liệu. Có thể cần `allowDiskUse: true` nếu sort trên dữ liệu lớn.

3.5 `$limit` và `$skip`

- **Định nghĩa/Ý nghĩa:** Phân trang/giới hạn kết quả.
- **Cú pháp:**

  `{ $limit: <n> } // lấy n doc đầu { $skip: <n> } // bỏ qua n doc đầu`
- **Ví dụ của bạn:**
- `{$limit: 3}`
- `{$skip: 2}`
- **Lưu ý:** Thường dùng kết hợp `$sort` → `$skip` → `$limit` để phân trang ổn định.

3.6 `$addFields`

- **Định nghĩa/Ý nghĩa:** Thêm (hoặc ghi đè) trường mới cho document ở "dạng phẳng".
- **Cú pháp:**

  `{ $addFields: { newField: <expression>, ... } }`
- **Ví dụ của bạn:**

```
db.products.aggregate([
    {$addFields: {
      total:{
        $multiply:["$price","$quantity"]
      }
    }}
])
```

  → tạo thêm `total`
- **So sánh:** `$addFields` ≈ `$set` (từ MongoDB 4.2, `$set` là alias thân thiện; `$set` thường được ưa dùng hơn gần đây).

3.7 `$merge`

- **Định nghĩa/Ý nghĩa:** Ghi kết quả pipeline vào một collection (upsert/replace/merge theo khoá).
- **Cú pháp cơ bản (ngắn gọn):**

  `{ $merge: "<targetColl>" }`

  hoặc chi tiết:

  `{ $merge: { into: "<targetColl>" or { db: "...", coll: "..." }, on: <field or [fields]>, // khoá khớp (mặc định _id) whenMatched: "replace"|"keepExisting"|"merge"|[pipeline], whenNotMatched: "insert"|"discard"|"fail" } }`
- **Ví dụ của bạn:**

```
//8.merge
use("demodb3")
db.products.aggregate([
    {
        $group: {
          _id: "category",
          totalSale: {
            $sum:{$multiply:["$price","$quantity"]}
          }
        }
    },
    {
```

```
        $merge: "summary"
    }
])
db.summary.find()
```

→ Kết quả ghi vào collection `summary`. Nếu không chỉ định on, mặc định dùng `_id`. Nếu `summary` chưa có, MongoDB sẽ tạo.

- **Khác biệt với `$out`:** `$merge` linh hoạt hơn (upsert/merge). `$out` **thay thế** toàn bộ collection đích bằng kết quả pipeline.

3.8 `$out`

- **Định nghĩa/Ý nghĩa:** Xuất kết quả pipeline thành **một collection mới** (hoặc ghi đè hoàn toàn collection đích).
- **Cú pháp:**

  `{ $out: "<targetCollection>" }`
- **Ví dụ của bạn:**

```
use("demodb3")
db.products.aggregate([
    {$project: {
      product:1,
      total:{$multiply:["$price","$quantity"]}

    }},
    {$out:"saleReport"}
])
use("demodb3")
db.saleReport.find()
```

→ Tạo/ghi đè `saleReport` với đúng **kết quả** của pipeline (các field đúng như sau `$project`).

- **Lưu ý:** `$out` thay thế toàn bộ dữ liệu collection đích; thận trọng khi dùng trên collection đang sử dụng.

3.9 Toán tử biểu thức số học/thống kê

`$multiply`

- **Ý nghĩa:** Nhân các toán hạng (2+ tham số).
- **Cú pháp:** `{ $multiply: [ <expr1>, <expr2>, ... ] }`
- **Ví dụ:** `{ $multiply: ["$price", "$quantity"] }`

`$sum`

- **Ý nghĩa:** Trong `$group`, tính tổng qua các document trong nhóm.
- **Cú pháp:** `{ $sum: <expr> }`
- **Ví dụ:** `{ $sum: { $multiply: ["$price","$quantity"] } }`

`$avg`

- **Ý nghĩa:** Trung bình cộng trên nhóm.
- **Cú pháp:** `{ $avg: <expr> }`
- **Ví dụ:** `{ $avg: "$price" }`

4) Giải thích các pipeline cụ thể của bạn

4.1 Lọc theo category

`db.products.aggregate([{ $match: { category:"Stationery" } }])`

- Lấy tất cả doc có `category = "Stationery"`.

Cảnh báo chính tả: Bạn có chỗ ghi “Stationary” (thiếu ‘e’) – cần thống nhất “Stationery”.

4.2 Tổng doanh thu theo từng loại

```
db.products.aggregate([ { $group: { _id: "$category", totalSale: { $sum: { $multiply: ["$price","$quantity"] } } } } ])
```

- Nhóm theo `category`.
- Tính `totalSale` mỗi nhóm.

4.3 Chọn field và thêm trường tính toán

```
db.products.aggregate([ { $project: { _id: 0, product: 1, total: { $multiply: ["$price", "$quantity"] } } } ])
```

- Ẩn `_id`, giữ `product`, thêm `total`.

4.4 Sắp xếp giảm dần theo giá

```
db.products.aggregate([{ $sort: { price: -1 } }])
```

- Món nào đắt nhất đứng đầu.

4.5 Giới hạn 3 dòng đầu

```
db.products.aggregate([{ $limit: 3 }])
```

4.6 Bỏ qua 2 dòng đầu

```
db.products.aggregate([{ $skip: 2 }])
```

4.7 Thêm trường `total`

```
db.products.aggregate([ { $addFields: { total: { $multiply: ["$price","$quantity"] } } } ])
```

4.8 Ghi kết quả nhóm vào `summary` (sửa `_id`)

```
db.products.aggregate([ { $group: { _id: "$category", // sửa từ "category" → "$category" totalSale: { $sum: { $multiply: ["$price","$quantity"] } } } }, { $merge: "summary" } ])
```

- Kết quả: `db.summary.find()` sẽ thấy mỗi nhóm 1 doc (key `_id` là category, `totalSale` là tổng).

4.9 Xuất báo cáo ra `saleReport`

```
db.products.aggregate([ { $project: { product: 1, total: { $multiply: ["$price","$quantity"] } } }, { $out: "saleReport" } ]) // Xem: db.saleReport.find()
```

- `saleReport` chứa các field đúng như sau `$project`.

4.10 Lấy sản phẩm rẻ nhất trong Stationery

```
db.products.aggregate([ { $match: { category: "Stationery" } }, { $sort: { price: 1 } }, { $limit: 1 } ])
```

- Lọc nhóm → sort tăng → lấy dòng đầu = rẻ nhất.

4.11 Giá TB theo nhóm, sort tăng

```
db.products.aggregate([ { $group: { _id: "$category", avgPrice: { $avg: "$price" } } }, { $sort: { avgPrice: 1 } } ])
```

4.12 Tổng doanh thu & tổng số lượng theo nhóm, sort theo doanh thu giảm dần

```
db.products.aggregate([ { $group: { _id: "$category", totalSale: { $sum: { $multiply: ["$price", "$quantity"] } }, totalQty: { $sum: "$quantity" } } }, { $sort: { totalSale: -1 } } ])
```

4.13 Bỏ qua sp đắt nhất, lấy 2 sp tiếp, chỉ trả `product` và `total`

```
db.products.aggregate([ { $sort: { price: -1 } }, { $skip: 1 }, { $limit: 2 }, { $project: { _id: 0, product: 1, total: { $multiply: ["$price", "$quantity"] } } } ])
```

4.14 Bài tổng hợp (2.3 + 2.4)

```
db.products.aggregate([ { $group: { _id: "$category", totalSale: { $sum: { $multiply: ["$price", "$quantity"] } } } }, { $sort: { totalSale: -1 } }, { $skip: 1 }, { $limit: 2 }, { $project: { _id: 1, totalSale: 1 } } ])
```

- Tính doanh thu theo nhóm → sort giảm → bỏ top 1 → lấy 2 nhóm tiếp → chỉ trả `_id` và `totalSale`.

4.15 Pipeline "nhiều stage" (Accessories)

```
db.products.aggregate([ { $match: { category: "Accessories" } }, // a) lọc { $addFields: { total: { $multiply: ["$price","$quantity"] } } }, // b) tính total { $addFields: { discountTotal: 205 } }, // c) thêm trường hằng { $match: { total: { $gt: 10 } } }, // d) giữ total > 10 { $sort: { discountTotal: -1 } } // e) sắp xếp theo discountTotal ])
```

- Ở bước (e), vì `discountTotal` là hằng 205 cho mọi doc, sort không thay đổi thứ tự thực tế.
  → Nếu ý định là sort theo `total` (hợp lý hơn), dùng `{ $sort: { total: -1 } }`.

5) Best practices & mẹo tối ưu

- **Đặt `$match` sớm**: Giảm dữ liệu truyền qua các stage nặng như `$group`, `$sort`.
- **Index**: Tạo index trên các field lọc/sort thường xuyên (vd: `{ category: 1, price: 1 }`) để tăng tốc.
- **Chính tả nhất quán**: Dùng "Stationery" thay vì lúc "Stationary".
- **Cẩn thận `$out`**: Nó **ghi đè** toàn bộ collection đích.
- **`$merge` vs `$out`**: `$merge` linh hoạt (on/whenMatched/whenNotMatched); `$out` dùng khi bạn muốn snapshot kết quả thành bảng báo cáo mới.
- **Kiểu dữ liệu số**: Đảm bảo `price`/`quantity` là number, không phải string.