Giải thích shopDB_session06

27 Tháng Mười 2025 7:42 CH

0) Chọn database

Lệnh

```js
use("ShopDB");
```

Tên gọi & ý nghĩa

- `use("<dbName>")`: lệnh của **mongosh** để **chuyển ngữ cảnh** sang database tên ShopDB.
- Database **chỉ được tạo thật** khi có thao tác **ghi** đầu tiên (insert, createCollection…).

Cách hoạt động

- Sau lệnh này, biến đặc biệt db sẽ trỏ đến ShopDB.
- Nếu ShopDB chưa có trên disk, nó sẽ "tồn tại" khi bạn insert dữ liệu đầu tiên.

1) Tạo collection & chèn dữ liệu

1.1 `customers`

Lệnh

```
db.customers.insertMany([
  { _id: 1, name: "Alice",   email: "alice@example.com",   city: "Hanoi" },
  { _id: 2, name: "Bob",     email: "bob@example.com",     city: "HCM" },
  { _id: 3, name: "Charlie", email: "charlie@example.com", city: "Danang" }
]);
```

Tên gọi & ý nghĩa

- `db.customers`: truy cập collection `customers` trong DB hiện tại.
- `insertMany([...])`: **chèn nhiều tài liệu** (documents) cùng lúc.

Cú pháp & hoạt động

- Mỗi phần tử trong mảng là **1 document** JSON.
- `_id` là **khóa chính** (duy nhất). Ở đây bạn tự đặt kiểu **Number** (1,2,3). Nếu không khai báo, MongoDB sẽ tự sinh **ObjectId**.
- Nếu collection chưa tồn tại, **MongoDB tự tạo** khi insert.

1.2 `orders`

Lệnh

```
db.orders.insertMany([
  { _id: 101, customerId: 1, total: 120, status: "shipped", date: ISODate("2025-10-20") },
  { _id: 102, customerId: 1, total: 80,  status: "pending", date: ISODate("2025-10-21") },
  { _id: 103, customerId: 2, total: 200, status: "shipped", date: ISODate("2025-10-22") }
]);
```

Tên gọi & ý nghĩa

- `customerId`: **khóa ngoại logic** (không ràng buộc như SQL), trỏ sang `_id` trong `customers`.
- `ISODate("YYYY-MM-DD")`: tạo đối tượng **Date** chuẩn ISO-8601.

Cách hoạt động

- `orders` là collection giao dịch; mỗi document là một đơn hàng.
- Không có ràng buộc FK cứng, nên **MongoDB không tự kiểm tra** `customerId` có tồn tại trong `customers` hay không—việc đó do ứng dụng/aggregation đảm nhiệm.

2) Bài 1.1 – Ghép đơn hàng với khách hàng (JOIN)

Mục tiêu

Lấy danh sách **orders** kèm tên khách hàng tương ứng.

Lệnh

```
use("ShopDB");
db.orders.aggregate([
  {
    $lookup: {
      from: "customers",
      localField: "customerId",
      foreignField: "_id",
      as: "customerInfo"
    }
  },
  { $unwind: "$customerInfo" },
  {
    $project: {
      _id: 1,
      name: "$customerInfo.name",
      total: 1,
      status: 1,
      date: 1
    }
  }
]);
```

Các stage & ý nghĩa

1. `$lookup` — **Nối (left outer join) 2 collection**

- **`from`**: collection đích để nối (`"customers"`).

- **`localField`**: trường ở pipeline hiện tại (tài liệu `orders`) — `"customerId"`.
- **`foreignField`**: trường ở collection đích để đối sánh — `"_id"`.
- **`as`**: tên mảng chứa kết quả khớp — `"customerInfo"`.

  👉 Sau stage này, mỗi order sẽ có thêm **mảng** `customerInfo` (thường 1 phần tử nếu khớp).

2. `$unwind` — **Bung mảng** thành document đơn

- `$unwind: "$customerInfo"` biến mỗi phần tử của `customerInfo` thành một document riêng.
- Nếu mảng rỗng (không khớp), document sẽ **bị loại**.
- Có thể giữ lại bằng `preserveNullAndEmptyArrays: true` (xem phần "Lưu ý").

3. `$project` — **Chọn/đổi tên trường xuất ra**

- 1 nghĩa là **giữ nguyên** trường.
- `name: "$customerInfo.name"` là **đổi tên** (gán giá trị của trường lồng vào trường mới `name`).
- Kết quả chỉ gồm: `_id` (của order), `name` (của customer), `total`, `status`, `date`.

Kết quả kỳ vọng (ví dụ)

```
[
  { _id: 101, name: "Alice", total: 120, status: "shipped", date: ISODate("2025-10-20") },
  { _id: 102, name: "Alice", total: 80,  status: "pending", date: ISODate("2025-10-21") },
  { _id: 103, name: "Bob",   total: 200, status: "shipped", date: ISODate("2025-10-22") }
]
```

Ghi chú hiệu năng

- Nên tạo index trên **`orders.customerId`** và **`customers._id`** (mặc định `_id` đã có index).

```
js

db.orders.createIndex({ customerId: 1 });
```

3) Bài 1.2 – Tính tổng chi tiêu mỗi khách hàng

Mục tiêu

Tổng hợp số tiền (`totalSpent`) và số đơn (`orderCount`) theo từng `customerId`, rồi **nối ngược** sang `customers` để lấy tên.

Lệnh

```
use("ShopDB");
db.orders.aggregate([
  {
    $group: {
      _id: "$customerId",
      totalSpent: { $sum: "$total" },
      orderCount: { $sum: 1 } // đếm số document
    }
  },
  {
    $lookup: {
      from: "customers",
      localField: "_id",
      foreignField: "_id",
      as: "customerInfo"
    }
  },
```

```
  { $unwind: "$customerInfo" },
  {
    $project: {
      _id: 1,
      name: "$customerInfo.name",
      totalSpent: 1,
      orderCount: 1
    }
  }
]);
```

Các stage & ý nghĩa

4. $group — **Nhóm & tổng hợp**

- `_id: "$customerId"`: khóa nhóm là `customerId`.
- `totalSpent: { $sum: "$total" }`: cộng dồn trường `total`.
- `orderCount: { $sum: 1 }`: đếm số đơn (mỗi document cộng 1).

  👉 Kết quả tạm thời sẽ như:

  `[ { _id: 1, totalSpent: 200, orderCount: 2 },`
  `{ _id: 2, totalSpent: 200, orderCount: 1 } ]`

5. $lookup — **Nối với customers lấy tên**

- `localField: "_id"` (chính là `customerId` sau khi group).
- `foreignField: "_id"` của `customers`.
- Kết quả vào mảng `customerInfo`.

6. $unwind — **Bung** `customerInfo` thành object.
7. $project — **Chọn trường đầu ra**

- Giữ `_id` (tức customerId), `totalSpent`, `orderCount`.
- Thêm name từ `customerInfo.name`.

Kết quả kỳ vọng (ví dụ)

```
[
  { _id: 1, name: "Alice", totalSpent: 200, orderCount: 2 },
  { _id: 2, name: "Bob",   totalSpent: 200, orderCount: 1 }
]
// Charlie (_id:3) không có đơn -> không xuất hiện (vì $group chạy trên orders)
```

4) Lưu ý quan trọng & biến thể hay dùng

- **$lookup là LEFT OUTER JOIN** theo tài liệu gốc:

  Document bên trái (nguồn pipeline) **luôn giữ**, mảng as có thể rỗng nếu không khớp.

- **$unwind có thể làm mất document** nếu mảng rỗng.

  Dùng tùy chọn để **giữ lại**:

  `{ $unwind: { path: "$customerInfo", preserveNullAndEmptyArrays: true } }`

  Khi đó `name: "$customerInfo.name"` có thể `null`.

- **Muốn liệt kê TẤT CẢ khách hàng (kể cả không có đơn)** kèm *tổng chi tiêu = 0*, nên bắt đầu từ `customers` và dùng `$lookup` dạng pipeline:

```
db.customers.aggregate([
  {
    $lookup: {
      from: "orders",
      let: { cid: "$_id" },
      pipeline: [
        { $match: { $expr: { $eq: ["$customerId", "$$cid"] } } },
        { $group: { _id: null, totalSpent: { $sum: "$total" }, orderCount: { $sum: 1 } } }
      ],
      as: "stats"
    }
  },
  { $unwind: { path: "$stats", preserveNullAndEmptyArrays: true } },
  {
    $project: {
      _id: 1,
      name: 1,
      totalSpent: { $ifNull: ["$stats.totalSpent", 0] },
      orderCount: { $ifNull: ["$stats.orderCount", 0] }
    }
  }
]);
```

Sao chép mã

- **Kiểu dữ liệu `date`**:

  `ISODate("2025-10-20")` tạo **Date** thật, hỗ trợ so sánh thời gian, sắp xếp, filter theo range. Tránh lưu ngày ở dạng string nếu cần truy vấn theo thời gian.
- **$project – chọn/đổi tên**:
- `field: 1` → giữ nguyên trường từ input.
- `newName: "$path.to.field"` → tạo trường mới từ giá trị ở đường dẫn.
- Mặc định các trường **không liệt kê** sẽ bị loại trừ (trừ `_id`, phải set 0 nếu muốn bỏ).
- **$group – vai trò `_id`**:
- Mọi nhóm đều cần `_id` (khóa nhóm).
- Có thể group theo nhiều trường: `_id: { cid: "$customerId", status: "$status" }`.
- **Chỉ mục (Index) khuyến nghị**:
- `db.orders.createIndex({ customerId: 1 })` — tăng tốc `$lookup`/filter theo `customerId`.
- `db.orders.createIndex({ date: 1 })` — tăng tốc truy vấn theo thời gian.
- `_id` đã có index mặc định.

5) Tóm tắt nhanh

- `use("ShopDB")`: chuyển sang DB ShopDB (tạo thật khi ghi).
- `insertMany`: chèn nhiều document, tự tạo collection nếu chưa có.
- `ISODate(...)`: tạo đối tượng ngày chuẩn.
- `aggregate([...])`: chạy **pipeline** gồm nhiều stage nối tiếp.
- `$lookup`: nối (left join) 2 collection → trả mảng.
- `$unwind`: bung mảng thành nhiều documents (có thể giữ doc rỗng bằng `preserveNullAndEmptyArrays`).
- `$project`: chọn/đổi tên trường đầu ra.
- `$group`: nhóm & tổng hợp; `_id` là khóa nhóm; `$sum` cộng giá trị, `$sum:1` để đếm.