Session 04 Aggregation Pipeline

17 Tháng Mười 2025 7:16 CH

1.Define (định nghĩa)

Aggregation in MongoDB groups multiple documents, processes them, and returns summarized results.

(Aggregation trong MongoDB giúp nhóm nhiều document, xử lý dữ liệu, và trả về kết quả tổng hợp.)

→ Giúp ta thống kê, phân tích dữ liệu (giống như GROUP BY, SUM, COUNT trong SQL).

An aggregation pipeline is a sequence of stages, each transforming the data before passing it to the next stage.

(Một aggregation pipeline là chuỗi các giai đoạn, mỗi giai đoạn biến đổi dữ liệu trước khi chuyển sang giai đoạn kế tiếp.)

Syntax:

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

→ Mỗi **stage (giai đoạn)** bắt đầu bằng ký hiệu `$` và được đặt trong một mảng `[]`.

⚙️ 2. Important Aggregation Stages

(2. Các giai đoạn (stage) quan trọng trong pipeline)

2.1 `$project` — Select or Modify Fields

(Chọn hoặc chỉnh sửa trường dữ liệu)

The `$project` stage specifies which fields to include, exclude, or compute.

(Giai đoạn `$project` cho phép chọn, bỏ hoặc tính toán lại các trường dữ liệu.)

Syntax:

```js
{ $project: { field1: 1, field2: 0, newField: "expression" } }
```

- `1` → bao gồm trường
- `0` → loại bỏ trường
- `"expression"` → tạo trường mới dựa trên công thức hoặc chuỗi

Example:

```js
db.inspections.aggregate([
  { $project: { certificate_number: 1, business_name: 1, address: 1, _id: 0 } }
```

```
])
```

(Chỉ hiển thị 3 trường, bỏ `_id` mặc định.)

2.2 `$limit` — Restrict Output Count

(Giới hạn số lượng kết quả trả về)

The `$limit` stage restricts the number of documents passed to the next stage.

(Giai đoạn `$limit` giới hạn số document được chuyển đến giai đoạn kế tiếp.)

Syntax:

```
js                                                    Sao chép mã

{ $limit: <number> }
```

Example:

```
js                                                    Sao chép mã

db.inspections.aggregate([
  { $limit: 2 },
  { $project: { certificate_number: 1, business_name: 1, address: 1 } }
])
```

(Giới hạn kết quả còn 2 document và chỉ hiển thị các trường cần thiết.)

2.3 $group — Group and Aggregate

(Nhóm và tính toán dữ liệu)

The $group stage groups documents by a field and applies accumulator functions.

(Giai đoạn $group nhóm các document theo một trường và áp dụng các hàm tổng hợp.)

Syntax:

```
js                                                    Sao chép mã

{ $group: { _id: <expression>, <field>: { <accumulator>: <value> } } }
```

Common accumulators:

Function	Description	Example
`$sum`	Tổng giá trị	`{ totalQty: { $sum: "$quantity" } }`
`$avg`	Trung bình	`{ avgPrice: { $avg: "$price" } }`
`$min` / `$max`	Giá trị nhỏ/ lớn nhất	`{ minPrice: { $min: "$price" } }`
`$count`	Đếm số document	`{ count: { $count: {} } }`

Example:

```js
db.inspections.aggregate([
  { $group: { _id: "$address.city", count: { $count: {} } } }
])
```

(Nhóm theo thành phố và đếm số lượng document trong từng nhóm.)

2.4 `$match` — Filter Documents

(Lọc dữ liệu giống WHERE trong SQL)

The `$match` stage filters documents based on a condition.

(Giai đoạn `$match` lọc document theo điều kiện chỉ định.)

Syntax:

```js
{ $match: { <expression> } }
```

Example:

```js
db.inspections.aggregate([
  { $match: { "address.city": "JERSEY CITY" } },
  { $project: { business_name: 1, address: 1, _id: 0 } },
  { $limit: 10 }
])
```

(Lọc chỉ các document có `city = JERSEY CITY` và hiển thị 10 bản ghi đầu tiên.)

2.5 `$sort` — Sort Documents

(Sắp xếp dữ liệu)

The `$sort` stage sorts documents by specified field(s).

(Giai đoạn `$sort` sắp xếp dữ liệu theo trường chỉ định.)

Syntax:

```js
{ $sort: { field1: 1, field2: -1 } }
```

- `1` = ascending (tăng dần)
- `-1` = descending (giảm dần)

Example:

```js
```

```
db.inspections.aggregate([
  { $sort: { "address.city": -1, "business_name": 1 } },
  { $limit: 10 }
])
```

(Sắp xếp theo thành phố giảm dần, rồi theo tên doanh nghiệp tăng dần.)

2.6 `$skip` — Skip Documents

(Bỏ qua số document đầu tiên)

The `$skip` stage skips a specified number of documents before passing to next stage.

(Giai đoạn `$skip` bỏ qua một số lượng document trước khi chuyển tiếp dữ liệu.)

Syntax:

js Sao chép mã

```
{ $skip: <number> }
```

Example:

js Sao chép mã

```
db.inspections.aggregate([
  { $sort: { "address.city": -1 } },
  { $skip: 10 }
])
```

(Bỏ qua 10 document đầu tiên trong danh sách đã sắp xếp.)

2.7 `$addFields` — Add New Fields

(Thêm trường mới vào document)

The `$addFields` stage adds one or more new fields to each document.

(Giai đoạn `$addFields` thêm một hoặc nhiều trường mới vào mỗi document.)

Syntax:

js Sao chép mã

```
{ $addFields: { newField: <expression> } }
```

Example:

js Sao chép mã

```
db.inspections.aggregate([
  { $addFields: { "address.country": "USA" } },
  { $project: { business_name: 1, "address.city": 1, "address.country": 1, _id: 0
])
```

(Thêm trường `address.country = "USA"` cho mỗi document.)

2.8 $out — Export Result to a Collection

(Ghi kết quả pipeline ra một collection mới)

The $out stage writes the result of the pipeline to a collection (overwrites existing data).

(Giai đoạn $out ghi kết quả pipeline vào một collection — nếu collection tồn tại sẽ ghi đè dữ liệu cũ.)

Syntax:

```js
{ $out: "<collection_name>" }
```

Example:

```js
db.inspections.aggregate([
  { $limit: 8 },
  { $project: { certificate_number: 1, business_name: 1, address: 1 } },
  { $out: "out_inspection1" }
])
```

(Tạo collection mới `out_inspection1` chứa 8 document đầu tiên.)

2.9 $merge — Merge Result into a Collection

(Hợp nhất kết quả với collection khác)

The $merge stage stores pipeline results into an existing or new collection without deleting old data.

(Giai đoạn $merge lưu kết quả pipeline vào collection mới hoặc cũ **mà không xóa dữ liệu cũ**.)

Syntax:

```js
{
  $merge: {
    into: "<collection>",
    on: "<field>",
    whenMatched: "merge",
    whenNotMatched: "insert"
  }
}
```

Example:

```js
db.inspections.aggregate([
  { $match: { "address.city": "HAZLET" } },
```

```
  { $project: { certificate_number: 1, business_name: 1, address: 1 } },
  { $merge: { into: "out_inspection1", on: "_id", whenMatched: "replace", whenNot
])
```

(Hợp dữ liệu từ `inspections` vào `out_insp` ↓ `on1`. Nếu trùng `_id` → ghi đè, nếu không có → thêm mới.)

🔢 3. Aggregation Operators

(3. Các toán tử dùng trong Aggregation Pipeline)

3.1 Arithmetic Operators (Toán tử số học)

Operator	Description	Example	Giải thích
`$add`	Cộng giá trị	`{ $add: ["$limit", 2000] }`	Cộng số hoặc thời gian
`$subtract`	Trừ giá trị	`{ $subtract: ["$a", "$b"] }`	Trừ hai số hoặc ngày
`$multiply`	Nhân	`{ $multiply: ["$price", "$qty"] }`	Tính tổng tiền
`$divide`	Chia	`{ $divide: ["$a", "$b"] }`	Chia giá trị
`$mod`	Lấy dư	`{ $mod: ["$x", 2] }`	Lấy phần dư của phép chia

Example:

js — Sao chép mã

```
db.accounts.aggregate([
  { $project: { account_id: 1, limit: 1, newLimit: { $add: ["$limit", 2000] } } }
])
```

(Tạo cột `newLimit` bằng cách cộng 2000 vào `limit`.)

3.2 Array Operators (Toán tử mảng)

Operator	Description
`$first`, `$last`	Lấy phần tử đầu/cuối
`$size`	Đếm phần tử trong mảng
`$filter`	Lọc phần tử thỏa điều kiện
`$sortArray`	Sắp xếp mảng
`$concatArrays`	Gộp mảng
`$in`	Kiểm tra giá trị có trong mảng không

Example:

js — Sao chép mã

```
db.grades.aggregate([
  { $addFields: { exam_score: { $first: "$scores" } } },
```

```
  { $project: { student_id: 1, class_id: 1, exam_score: 1 } }
])
```

(Lấy điểm thi đầu tiên trong mảng `scores`.)

3.3 Boolean Operators (Toán tử logic)

Operator	Description	Example
`$and`	Cả hai điều kiện đúng	`{ $and: [cond1, cond2] }`
`$or`	Một trong hai đúng	`{ $or: [cond1, cond2] }`
`$not`	Phủ định điều kiện	`{ $not: [cond] }`

Example:

js — Sao chép mã

```
db.trips.aggregate([
  { $addFields: {
      op_status: { $and: [
        { $eq: ["$tripduration", 379] },
        { $eq: ["$usertype", "Subscriber"] }
      ]}
  }}
])
```

(Chỉ đặt `op_status: true` nếu `tripduration=379` và `usertype="Subscriber"`.)

3.4 Comparison Operators (So sánh)

Operator	Description	Giải thích
`$eq`	Equal	bằng
`$ne`	Not equal	khác
`$gt`	Greater than	lớn hơn
`$gte`	Greater or equal	lớn hơn hoặc bằng
`$lt`	Less than	nhỏ hơn
`$lte`	Less or equal	nhỏ hơn hoặc bằng

Example:

js — Sao chép mã

```
db.accounts.aggregate([
  { $project: {
      limit: 1,
      limit_status: { $and: [
        { $gt: ["$limit", 9000] },
        { $lt: ["$limit", 12000] }
      ]}
```

```
  }}
])
```

(Lọc các tài khoản có `limit` nằm trong khoảng 9000–12000.)

3.5 String Operators (Toán tử chuỗi)

Operator	Description	Example
`$concat`	Nối chuỗi	`{ $concat: ["A", " - ", "B"] }`
`$toUpper` / `$toLower`	Viết hoa / thường	`{ $toUpper: "$name" }`
`$split`	Tách chuỗi thành mảng	`{ $split: ["$text", " "] }`
`$trim` / `$rtrim` / `$ltrim`	Xóa khoảng trắng	`{ $trim: {input:"$name"} }`

Example:

js — Sao chép mã

```
db.trips.aggregate([
  { $project: {
      source_destination: {
        $concat: ["$start station name", " - ", "$end station name"]
      }
  }}
])
```

(Gộp tên trạm bắt đầu và trạm kết thúc thành 1 chuỗi mới.)

✅ 4. Summary (Tóm tắt cho người mới học)

- **Aggregation pipeline = multi-step data processing system.**
  (Aggregation pipeline = hệ thống xử lý dữ liệu nhiều bước.)
- **Main stages:** `$project`, `$match`, `$group`, `$sort`, `$limit`, `$skip`, `$addFields`, `$out`, `$merge`.
  (Các giai đoạn chính để lọc, nhóm, sắp xếp, và xuất dữ liệu.)
- **Operators:** arithmetic, array, boolean, comparison, string.
  (Các toán tử hỗ trợ phép tính, mảng, logic, so sánh và chuỗi.)
- **Practical uses:** data summarization, reports, dashboards.
  (Ứng dụng thực tế: tổng hợp dữ liệu, báo cáo, dashboard.)