Лабораторна робота № 6 Бази даних та інформаційні системи

Тема: Практичне використання Aggregation Framework у MongoDB

Мета: Закріпити знання про основні стадії Aggregation Framework. Навчитися будувати ефективні агрегаційні запити. Освоїти методи фільтрації, групування, сортування та обробки масивів у MongoDB. Практично працювати з $match, $group, $sort, $unwind, $lookup, $project. Аналізувати продуктивність агрегацій та оптимізувати запити.

Виконала: студентка групи МІТ-31, Панченко Владислава

Вихідні дані:

Маємо колекції orders, customers, products:

```
myDatabase> db.orders.find().pretty()
[
  {
    _id: ObjectId('6810e671a8feed8730b5f899'),
    orderId: 'ORD001',
    status: 'Completed'
  },
  {
    _id: ObjectId('6810e920a8feed8730b5f89c'),
    orderId: 'ORD001',
    customerId: ObjectId('6810e855a8feed8730b5f89a'),
    date: ISODate('2024-01-12T00:00:00.000Z'),
    items: [
      { product: 'Laptop', quantity: 1, price: 1200 },
      { product: 'Mouse', quantity: 2, price: 50 }
    ],
    status: 'Completed'
  }
]
```

```
myDatabase> db.customers.find().pretty()
[
  {
    _id: ObjectId('6810e855a8feed8730b5f89a'),
    name: 'John Doe',
    email: 'john.doe@example.com',
    city: 'New York',
    registeredAt: ISODate('2021-03-15T00:00:00.000Z')
  }
]
```

```
myDatabase> db.products.find().pretty()
[
  {
    _id: ObjectId('6810e8c4a8feed8730b5f89b'),
    name: 'Laptop',
    category: 'Electronics',
    price: 1200,
    stock: 15
  }
]
```

Завдання:

Частина 1: Базові агрегаційні операції

1. Відфільтруйте замовлення за останні 3 місяці

Використовуємо $match для фільтрування замовлень.

```
myDatabase>   db.orders.aggregate([
...     {
...       $match: {
...         date: { $gte: new ISODate("2024-01-01T00:00:00Z") }
...       }
...     }
...   ]);
[
  {
    _id: ObjectId('6810e920a8feed8730b5f89c'),
    orderId: 'ORD001',
    customerId: ObjectId('6810e855a8feed8730b5f89a'),
    date: ISODate('2024-01-12T00:00:00.000Z'),
    items: [
      { product: 'Laptop', quantity: 1, price: 1200 },
      { product: 'Mouse', quantity: 2, price: 50 }
    ],
    status: 'Completed'
  }
]
```

2. Групування замовлень за місяцем

Використовуємо оператор $group та функцію $month.

```
myDatabase> db.orders.aggregate([
...     {
...       $project: {
...         month: { $month: "$date" },
...         orderId: 1
...       }
...     },
...     {
...       $group: {
...         _id: "$month",
...         totalOrders: { $sum: 1 }
...       }
...     }
...   ]);
[ { _id: 1, totalOrders: 1 }, { _id: null, totalOrders: 1 } ]
```

3. Сортування за сумою замовлення

Спочатку обчислюємо суму замовлень, потім сортуємо. Використовуємо $project і $sum.

```
myDatabase> db.orders.aggregate([
...   {
...     $project: {
...       orderId: 1,
...       totalAmount: {
...         $sum: { $map: {
...           input: "$items",
...           as: "item",
...           in: { $multiply: ["$$item.price", "$$item.quantity"] }
...         }}
...       }
...     }
...   },
...   { $sort: { totalAmount: -1 } }
... ]);
...
[
  {
    _id: ObjectId('6810e920a8feed8730b5f89c'),
    orderId: 'ORD001',
    totalAmount: 1300
  },
  {
    _id: ObjectId('6810e671a8feed8730b5f899'),
    orderId: 'ORD001',
    totalAmount: 0
  }
]
```

Частина 2: Робота з масивами

4. Розгорніть масив items у замовленнях

Використовуємо $unwind щоб розгорнути масив items і вивести кожен товар в окремий документ.

```
myDatabase> db.orders.aggregate([
...   { $unwind: "$items" }
... ]);
...
[
  {
    _id: ObjectId('6810e920a8feed8730b5f89c'),
    orderId: 'ORD001',
    customerId: ObjectId('6810e855a8feed8730b5f89a'),
    date: ISODate('2024-01-12T00:00:00.000Z'),
    items: { product: 'Laptop', quantity: 1, price: 1200 },
    status: 'Completed'
  },
  {
    _id: ObjectId('6810e920a8feed8730b5f89c'),
    orderId: 'ORD001',
    customerId: ObjectId('6810e855a8feed8730b5f89a'),
    date: ISODate('2024-01-12T00:00:00.000Z'),
    items: { product: 'Mouse', quantity: 2, price: 50 },
    status: 'Completed'
  }
]
```

5. Підрахуйте кількість проданих одиниць товарів

Використовуємо $group.

```
myDatabase>   db.orders.aggregate([
...     { $unwind: "$items" },
...     { $group: {
...       _id: "$items.product",
...       totalQuantitySold: { $sum: "$items.quantity" }
...     }}
...   ]);
[
  { _id: 'Laptop', totalQuantitySold: 1 },
  { _id: 'Mouse', totalQuantitySold: 2 }
]
```

Частина 3: З’єднання колекцій ($lookup)

6. Отримання інформації про клієнтів у замовленнях

```
myDatabase> db.orders.aggregate([
...   {
...     $lookup: {
...       from: "customers",
...       localField: "customerId",
...       foreignField: "_id",
...       as: "customerDetails"
...     }
...   }
... ]);
...
[
  {
    _id: ObjectId('6810e671a8feed8730b5f899'),
    orderId: 'ORD001',
    status: 'Completed',
    customerDetails: []
  },
  {
    _id: ObjectId('6810e920a8feed8730b5f89c'),
    orderId: 'ORD001',
    customerId: ObjectId('6810e855a8feed8730b5f89a'),
    date: ISODate('2024-01-12T00:00:00.000Z'),
    items: [
      { product: 'Laptop', quantity: 1, price: 1200 },
      { product: 'Mouse', quantity: 2, price: 50 }
    ],
    status: 'Completed',
    customerDetails: [
      {
        _id: ObjectId('6810e855a8feed8730b5f89a'),
        name: 'John Doe',
        email: 'john.doe@example.com',
        city: 'New York',
        registeredAt: ISODate('2021-03-15T00:00:00.000Z')
      }
    ]
  }
]
```

7. Визначте найбільш активних клієнтів

Для цього групуємо замовлення за customerId і рахуємо кількість замовлень для кожного клієнта, виводимо 5 найактивніших клієнтів.

```
myDatabase> db.orders.aggregate([
...     { $group: { _id: "$customerId", totalOrders: { $sum: 1 } } },
...     { $sort: { totalOrders: -1 } },
...     { $limit: 5 }
...   ]);
...
[
  { _id: null, totalOrders: 1 },
  { _id: ObjectId('6810e855a8feed8730b5f89a'), totalOrders: 1 }
]
```

Частина 4: Оптимізація запитів

8. Перевірте продуктивність запиту

Використовуємо команду .explain()

```
myDatabase> db.orders.aggregate([
...   {
...     $lookup: {
...       from: "customers",
...       localField: "customerId",
...       foreignField: "_id",
...       as: "customerDetails"
...     }
...   }
... ]).explain("executionStats");
...
{
  explainVersion: '2',
  queryPlanner: {
    namespace: 'myDatabase.orders',
    parsedQuery: {},
    indexFilterSet: false,
    queryHash: 'B3956D95',
    planCacheShapeHash: 'B3956D95',
    planCacheKey: 'E66D7B71',
    optimizationTimeMillis: 0,
    optimizedPipeline: true,
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    prunedSimilarIndexes: false,
    winningPlan: {
      isCached: false,
      queryPlan: {
        stage: 'EQ_LOOKUP',
        planNodeId: 2,
```

9. Оптимізуйте агрегаційний запит

Використовуємо проекції ($project) для обмеження кількості полів, які проходять через pipeline

```
myDatabase> db.orders.aggregate([
...   { $project: { orderId: 1, totalAmount: { $sum: "$items.price" } } },
...   { $sort: { totalAmount: -1 } }
... ]);
...
[
  {
    _id: ObjectId('6810e920a8feed8730b5f89c'),
    orderId: 'ORD001',
    totalAmount: 1250
  },
  {
    _id: ObjectId('6810e671a8feed8730b5f899'),
    orderId: 'ORD001',
    totalAmount: 0
  }
]
```

Додаткові завдання

10. Визначте категорії товарів із найбільшою кількістю продажів. Використайте $group для підрахунку загальної кількості проданих товарів за категоріями. Відсортуйте результат за спаданням.

Спочатку розгортаємо масив товарів в кожному замовленні за допомогою $unwind, потім групуємо за категорією, підсумовуємо і сортуємо.

```
myDatabase>   db.orders.aggregate([
...     { $unwind: "$items" },
...     { $lookup: {
...         from: "products",
...         localField: "items.product",
...         foreignField: "name",
...         as: "productDetails"
...       }
...     },
...     { $unwind: "$productDetails" },
...     { $group: {
...         _id: "$productDetails.category",
...         totalSales: { $sum: "$items.quantity" }
...       }
...     },
...     { $sort: { totalSales: -1 } }
...   ]);
[ { _id: 'Electronics', totalSales: 1 } ]
```

11. Розрахуйте середню ціну товарів у кожній категорії. Використайте $group для підрахунку середньої ціни товарів у кожній категорії. Знайдіть користувачів, які зробили більше одного замовлення.

Використовуємо $unwind для розгортання масиву товарів, а потім групуємо за категорією, обчислюючи середню ціну за допомогою $avg

```
myDatabase>   db.orders.aggregate([
...     { $unwind: "$items" },
...     { $lookup: {
...         from: "products",
...         localField: "items.product",
...         foreignField: "name",
...         as: "productDetails"
...       }
...     },
...     { $unwind: "$productDetails" },
...     { $group: {
...         _id: "$productDetails.category",
...         avgPrice: { $avg: "$items.price" }
...       }
...     },
...     { $sort: { avgPrice: -1 } }
...   ]);
...
[ { _id: 'Electronics', avgPrice: 1200 } ]
```

12. Використайте $group і $match, щоб знайти клієнтів, які мали більше одного замовлення.

Використовуємо агрегацію, щоб згрупувати замовлення за customerId і підраховуємо кількість замовлень для кожного клієнта. Потім застосовуємо $match, щоб вибрати лише тих користувачів, у яких більше одного замовлення.

```
myDatabase>   db.orders.aggregate([
...     { $group: {
...         _id: "$customerId",
...         orderCount: { $sum: 1 }
...       }
...     },
...     { $match: { orderCount: { $gt: 1 } } }
...   ]);
...

myDatabase>
```

Запитання для самоперевірки

1. Що таке Aggregation Framework у MongoDB?

Aggregation Framework у MongoDB — це набір операцій, який дозволяє обробляти і перетворювати дані з колекцій на більш складні результати.

2. Які основні стадії агрегаційного пайплайну ви використовували?

$match — для фільтрації даних на основі певних умов

$unwind — для розгортання масивів

$group — для групування даних за певними полями

$sort — для сортування даних

$project — для вибору конкретних полів, які потрібно повернути

$lookup — для об'єднання даних з іншими колекціями

3. У чому різниця між $push та $addToSet?

$push — додає елемент до масиву, не перевіряючи наявність такого елемента в масиві

$addToSet — додає елемент до масиву, але тільки якщо такого елемента немає в масиві

4. Як працює $lookup і для чого його використовують?

$lookup дозволяє об'єднувати документи з іншої колекції.

5. Як можна оптимізувати агрегаційні запити?

Індекси на полях, які часто фільтруються або сортуються ($match, $sort).

$match на початку — чим менше даних далі, тим швидше запит.

$project — обирати тільки потрібні поля.

Обмежити використання $unwind, бо він сильно збільшує кількість документів.

Висновок: у цій роботі закріпили знання про основні стадії Aggregation Framework. Навчилися будувати ефективні агрегаційні запити. Освоїли методи фільтрації, групування, сортування та обробки масивів у MongoDB. Практично працювали з $match, $group, $sort, $unwind, $lookup, $project. Аналізуватл продуктивність агрегацій та оптимізували запити.