

Київський національний університет ім. Тараса Шевченка

Кафедра мережевих та інтернет технологій

Лабораторна робота №4

Дисципліна: Базы 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

Пась Олександра

Хід виконання

Створення колекцій orders, customers та products:

```
db.orders.insertOne({
  "orderId": "ORD001",
  "customerId": ObjectId("65f1a3d5a123456789abcd01"),
  "date": ISODate("2024-01-12"),
  "items": [
    { "product": "Laptop", "quantity": 1, "price": 1200 },
    { "product": "Mouse", "quantity": 2, "price": 50 }
  ],
  "status": "Completed"
})

db.customers.insertOne({
  "_id": ObjectId("65f1a3d5a123456789abcd01"),
  "name": "John Doe",
  "email": "john.doe@example.com",
  "city": "New York",
  "registeredAt": ISODate("2021-03-15")
})

db.products.insertOne({
  "name": "Laptop",
  "category": "Electronics",
```

```
"price": 1200,  
"stock": 15  
})
```

Частина 1: Базові агрегаційні операції

1. Відфільтрувати замовлення за останні 3 місяці

```
db.orders.aggregate([  
  
  {  
  
    $match: {  
  
      date: { $gte: new Date(new Date().setMonth(new Date().getMonth() - 3)) }  
  
    }  
  
  }  
  
])
```

2. Групування замовлень за місяцем

```
db.orders.aggregate([  
  
  {  
  
    $group: {  
  
      _id: { $month: "$date" },  
  
      totalOrders: { $sum: 1 }  
  
    }  
  
  }  
  
])
```

Результат:

```
< {  
  _id: 1,  
  totalOrders: 1  
}
```

3. Сортування за сумою замовлення

```
db.orders.aggregate([  
  
  {
```

```

    $addFields: {
      totalAmount: {
        $sum: {
          $map: {
            input: "$items",
            as: "item",
            in: { $multiply: ["$$item.quantity", "$$item.price"] }
          }
        }
      }
    }
  },
  { $sort: { totalAmount: -1 } }
)

```

Результат:

```

< {
  _id: ObjectId('67e4266ba24557a1ea9e1c4d'),
  orderId: 'ORD001',
  customerId: ObjectId('65f1a3d5a123456789abcd01'),
  date: 2024-01-12T00:00:00.000Z,
  items: [
    {
      product: 'Laptop',
      quantity: 1,
      price: 1200
    },
    {
      product: 'Mouse',
      quantity: 2,
      price: 50
    }
  ],
  status: 'Completed',
  totalAmount: 1300
}

```

У частині 1 ми застосували базові агрегаційні операції: за допомогою оператора \$match відфільтрували замовлення за останні 3 місяці, використали \$group для групування замовлень за місяцем, а також застосували \$addFields та \$sort для обчислення та сортування замовлень за сумою, що дозволило нам отримати загальну вартість кожного замовлення.

Частина 2: Робота з масивами

4. Розгорніть масив items у замовленнях

```
db.orders.aggregate([  
  { $unwind: "$items" }  
])
```

Результат:



```
< {  
  _id: ObjectId('67e4266ba24557alea9e1c4d'),  
  orderId: 'ORD001',  
  customerId: ObjectId('65f1a3d5a123456789abcd01'),  
  date: 2024-01-12T00:00:00.000Z,  
  items: {  
    product: 'Laptop',  
    quantity: 1,  
    price: 1200  
  },  
  status: 'Completed'  
}  
{  
  _id: ObjectId('67e4266ba24557alea9e1c4d'),  
  orderId: 'ORD001',  
  customerId: ObjectId('65f1a3d5a123456789abcd01'),  
  date: 2024-01-12T00:00:00.000Z,  
  items: {  
    product: 'Mouse',  
    quantity: 2,  
    price: 50  
  },  
  status: 'Completed'  
}
```

5. Підрахуйте кількість проданих одиниць товарів

```
db.orders.aggregate([  
  { $unwind: "$items" },  
  {  
    $group: {  
      _id: "$items.product",  
      totalSold: { $sum: "$items.quantity" }  
    }  
  }  
])
```

Результат:

```
< {  
  _id: 'Laptop',  
  totalSold: 1  
}  
{  
  _id: 'Mouse',  
  totalSold: 2  
}
```

У частині 2 ми зосередилися на роботі з масивами: оператор \$unwind розгорнув масив items для кожного замовлення, а потім за допомогою \$group підраховували кількість проданих одиниць товарів, що є корисним для аналізу продажів.

Частина 3: З'єднання колекцій (\$lookup)

6. Отримання інформації про клієнтів у замовленнях

```
db.orders.aggregate([  
  {  
    $lookup: {  
      from: "customers",  
      localField: "customerId",  
      foreignField: "_id",  
      as: "customer_info"  
    }  
  }  
])
```

Результат:

```
< {
  _id: ObjectId('67e4266ba24557a1ea9e1c4d'),
  orderId: 'ORD001',
  customerId: ObjectId('65f1a3d5a123456789abcd01'),
  date: 2024-01-12T00:00:00.000Z,
  items: [
    {
      product: 'Laptop',
      quantity: 1,
      price: 1200
    },
    {
      product: 'Mouse',
      quantity: 2,
      price: 50
    }
  ],
  status: 'Completed',
  customer_info: [
    {
      _id: ObjectId('65f1a3d5a123456789abcd01'),
      name: 'John Doe',
      email: 'john.doe@example.com',
      city: 'New York',
      registeredAt: 2021-03-15T00:00:00.000Z
    }
  ]
}
```

7. Визначте найбільш активних клієнтів

```
Код: db.orders.aggregate([
  {
    $group: {
      _id: "$customerId",
      totalOrders: { $sum: 1 }
    }
  },
  { $sort: { totalOrders: -1 } },
  { $limit: 5 }
])
```

Результат:

```
< {
  _id: ObjectId('65f1a3d5a123456789abcd01'),
  totalOrders: 1
}
```

У частині 3 застосовано \$lookup для об'єднання даних з колекції customers з даними замовлень, що дозволило отримати додаткову інформацію про клієнтів, а також було виконано групування та сортування для визначення найбільш активних клієнтів.

Частина 4: Оптимізація запитів

8. Перевірте продуктивність запиту

```
db.orders.explain("executionStats").aggregate([
  { $match: { status: "Completed" } }
])
```

Результат:

```
< {
  explainVersion: '1',
  queryPlanner: {
    namespace: 'Lab6.orders',
    parsedQuery: {
      status: {
        '$eq': 'Completed'
      }
    },
    indexFilterSet: false,
    queryHash: '5D6543D9',
    planCacheShapeHash: '5D6543D9',
    planCacheKey: '485CB45D',
    optimizationTimeMillis: 0,
    optimizedPipeline: true,
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    prunedSimilarIndexes: false,
    winningPlan: {
      isCached: false,
      stage: 'COLLSCAN',
      filter: {
        status: {
          '$eq': 'Completed'
        }
      },
      direction: 'forward'
    },
    rejectedPlans: []
  },

```

```
    executionStats: {
      executionSuccess: true,
      nReturned: 1,
      executionTimeMillis: 1,
      totalKeysExamined: 0,
      totalDocsExamined: 1,
      executionStages: {
        isCached: false,
        stage: 'COLLSCAN',
        filter: {
          status: {
            '$eq': 'Completed'
          }
        },
        nReturned: 1,
        executionTimeMillisEstimate: 0,
        works: 2,
        advanced: 1,
        needTime: 0,
        needYield: 0,
        saveState: 0,
        restoreState: 0,
        isEOF: 1,
        direction: 'forward',
        docsExamined: 1
      }
    },
    queryShapeHash: '1DB714845DB57A135C73C4BE447B26F5A67C0E2D8453FB7AB21F6496CA1ECE24',
    command: {
      aggregate: 'orders',
      pipeline: [
        {
          '$match': {
            status: 'Completed'
          }
        }
      ]
    },

```

```

serverInfo: {
  host: 'DESKTOP-F3ENC70',
  port: 27017,
  version: '8.0.5',
  gitVersion: 'cb9e2e5e552ee39deale39d7859336456d0c9820'
},
serverParameters: {
  internalQueryFacetBufferSizeBytes: 104857600,
  internalQueryFacetMaxOutputDocSizeBytes: 104857600,
  internalLookupStageIntermediateDocumentMaxSizeBytes: 104857600,
  internalDocumentSourceGroupMaxMemoryBytes: 104857600,
  internalQueryMaxBlockingSortMemoryUsageBytes: 104857600,
  internalQueryProhibitBlockingMergeOnMongoS: 0,
  internalQueryMaxAddToSetBytes: 104857600,
  internalDocumentSourceSetWindowFieldsMaxMemoryBytes: 104857600,
  internalQueryFrameworkControl: 'trySbeRestricted',
  internalQueryPlannerIgnoreIndexWithCollationForRegex: 1
},
ok: 1
}

```

9. Оптимізуйте агрегаційний запит

```
db.orders.createIndex({ date: 1 })
```

Результат:

```
< date_1
```

Частина 4 була присвячена оптимізації запитів: ми скористалися методом `explain("executionStats")` для аналізу продуктивності запитів і виявили, що без індексації MongoDB виконує повне сканування колекції (COLLSCAN), тому було рекомендовано створити індекс для покращення швидкодії.

10. Визначте категорії товарів із найбільшою кількістю продажів

```

db.orders.aggregate([
  { $unwind: "$items" },
  {
    $lookup: {
      from: "products",
      localField: "items.product",
      foreignField: "name",
      as: "product_info"
    }
  },
  { $unwind: "$product_info" },

```



```
{
  $group: {
    _id: "$product_info.category",
    totalSold: { $sum: "$items.quantity" }
  }
},
{ $sort: { totalSold: -1 } }
])
```

Результат:

```
< {
  _id: 'Electronics',
  totalSold: 1
}
```

11. Розрахуйте середню ціну товарів у кожній категорії

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

Результат:

```
< {
  _id: 'Electronics',
  avgPrice: 1200
}
```

12. Знайдіть користувачів, які зробили більше одного замовлення

```
db.orders.aggregate([
  {
```

```
$group: {  
  _id: "$customerId",  
  orderCount: { $sum: 1 }  
}  
},  
{ $match: { orderCount: { $gt: 1 } } }  
])
```

Висновок: Під час цієї лабораторної роботи було досліджено операції з базою даних MongoDB, включаючи додавання, вибірку та агрегування даних. Також було виконано аналіз ефективності запитів за допомогою `explain("executionStats")`, що дозволяє оцінити їх продуктивність. Виявлено, що використання індексів може значно покращити швидкість виконання запитів, мінімізуючи кількість перевірених документів у колекції.

