# Лабораторна робота №6

**Тема:** Практичне використання Aggregation Framework у MongoDB

**З дисципліни:** Бази даних та інформаційні системи

**Студента групи МІТ-31:** Циба Ярослава

# Створення та наповнення бази даних:

use mydatabase db.products.insertMany([

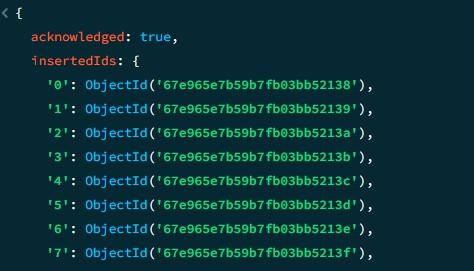
{ "name": "Laptop", "category": "Electronics", "price": 1200, "stock": 15 },

{ "name": "Sofa", "category": "Furniture", "price": 700, "stock": 10 },

{ "name": "Running Shoes", "category": "Clothing", "price": 100, "stock": 25 },

…

{ "name": "Drone", "category": "Electronics", "price": 600, "stock": 10 }

])

\

Рисунок 6.1 – Результат виконання

db.customers.insertMany([{ "name": "Alice Smith", "email": ["alice.smith@example.com](mailto:alice.smith@example.com)", "city": "Los Angeles", "registeredAt": ISODate("2022-01-10") },

{ "name": "Bob Johnson", "email": "[bob.johnson@example.com",](mailto:bob.johnson@example.com) "city": "Chicago", "registeredAt": ISODate("2023-06-25") },

{ "name": "Charlie Brown", "email": ["charlie.brown@example.com](mailto:charlie.brown@example.com)", "city": "Houston", "registeredAt": ISODate("2020-11-30") },

…

{ "name": "Daniel Bennett", "email": "[daniel.bennett@example.com",](mailto:daniel.bennett@example.com) "city": "Albuquerque", "registeredAt": ISODate("2023-08-14") }

])

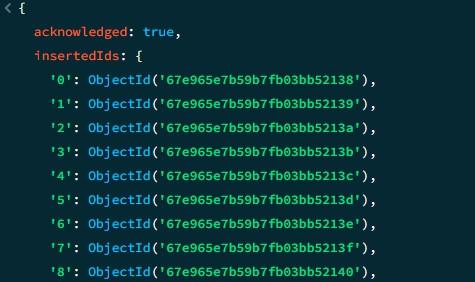


Рисунок 6.2 – Результат виконання

db.orders.insertMany([

{

"orderId": "ORD001",

"customerId": ObjectId("67e9674ab59b7fb03bb52156"), "date": ISODate("2024-01-12"),

"items": [

{ "product": "Laptop", "quantity": 1, "price": 1200 },

{ "product": "Mouse", "quantity": 2, "price": 50 }

],

"status": "Completed"

},

…

{

"orderId": "ORD030",

"customerId": ObjectId("67e9674ab59b7fb03bb52156"), "date": ISODate("2024-01-15"),

"items": [

{ "product": "Refrigerator", "quantity": 1, "price": 1200 },

{ "product": "Microwave", "quantity": 1, "price": 300 }

],

"status": "Shipped"

}

])

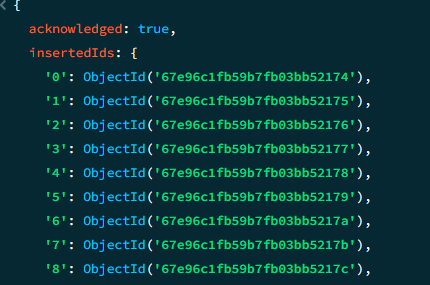


Рисунок 6.3 – Результат виконання

# Виконання запитів:

**Частина 1: Базові агрегаційні операції**

1. Відфільтруйте замовлення за останні 3 місяці const threeMonthsAgo = new Date()

threeMonthsAgo.setMonth(threeMonthsAgo.getMonth()-3) db.orders.aggregate([{$match: {date:{$gt: threeMonthsAgo}}}])



Рисунок 6.4 – Результат виконання запиту

1. Групування замовлень за місяцем

db.orders.aggregate({$group: {\_id: {$month: "$date"}, orders: {$push : "$$ROOT"}}})



Рисунок 6.5 – Результат виконання запиту

1. Сортування за сумою замовлення db.orders.aggregate(

[

{

$project: { orderId: 1, totalPrice: {

$sum: {

$map: {

input: '$items', as: 'item',

in: {

$multiply: [ '$$item.quantity', '$$item.price'

]

}

}

}

}

}

},

{ $sort: { totalPrice: -1 } }

]

)



Рисунок 6.6 – Результат виконання запиту

# Частина 2: Робота з масивами

1. Розгорніть масив items у замовленнях db.orders.aggregate([{

"$unwind": {

"path": "$items"

}

}

])



Рисунок 6.7 – Результат виконання запиту

1. Підрахуйте кількість проданих одиниць товарів db.orders.aggregate(

[

{ $unwind: { path: '$items' } },

{

$group: {

\_id: '$items.product',

bought: { $sum: '$items.quantity' }

}

}

])



Рисунок 6.8 – Результат виконання запиту

# Частина 3: З’єднання колекцій ($lookup)

1. Отримання інформації про клієнтів у замовленнях db.orders.aggregate([

{$lookup:{

from: "customers", localField: "customerId", foreignField: "\_id",

as: "customerData"

}

}])



Рисунок 6.9 – Результат виконання запиту

1. Визначте найбільш активних клієнтів db.orders.aggregate(

[

{

$group: {

\_id: '$customerId', orderAmount: { $sum: 1 }

}

},

{

$lookup: {

from: 'customers', localField: '\_id', foreignField: '\_id', as: 'customerData'

}

},

{ $sort: { orderAmount: -1 } },

{ $limit: 5 }

]

);



Рисунок 6.10 – Результат виконання запиту

# Частина 4: Оптимізація запитів

1. Перевірте продуктивність запиту

Для перевірки продуктивності певного запиту можна скористатись

.explain("executionStats"). За допомогою цього метода можна детально ознайомитись зі статистикою кожного етапу агрегації.

На прикладі попереднього запиту: db.orders.aggregate(

[

{

$group: {

\_id: '$customerId', orderAmount: { $sum: 1 }

}

},

{

$lookup: {

from: 'customers', localField: '\_id', foreignField: '\_id', as: 'customerData'

}

},

{ $sort: { orderAmount: -1 } },

{ $limit: 5 }

]

).explain("executionStats")



Рисунок 6.11 – Результат виконання запиту

1. Оптимізуйте агрегаційний запит

Для оптимізації запиту №7 можемо використати $project, щоб отримувалися тільки потрібні поля з колекції customers: db.orders.aggregate([

{

$group: {

\_id: "$customerId", orderAmount: { $sum: 1 }

}

},

{

$lookup: {

from: "customers", localField: "\_id", foreignField: "\_id",

pipeline: [{ $project: { name: 1} }], as: "customerData"

}

},

{ $sort: { orderAmount: -1 } },

{ $limit: 5 }

])

Порівняємо швидкість виконання запиту за допомогою

.explain(“executionStats”):

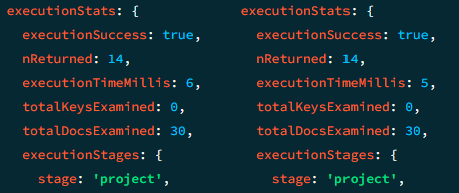


Рисунок 6.12 – Фрагмент статистики виконання неоптимізованого запиту (зліва) та оптимізованого (справа)

На порівнянні статистики (рис. 6.12) можна побачити приріст швидкості приблизно на 16.66%.

# Додаткові завдання

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

db.orders.aggregate( [

{ $project: { items: 1, \_id: 0 } },

{ $unwind: { path: '$items' } },

{

$group: {

\_id: '$items.product',

amount: { $sum: '$items.quantity' }

}

},

{

$lookup: {

from: 'products', localField: '\_id', foreignField: 'name', pipeline: [

{ $project: { category: 1, \_id: 0 } }

],

as: 'category'

}

},

{ $unwind: { path: '$category' } },

{

$group: {

\_id: '$category.category', amount: { $sum: '$amount' }

}

},

{ $sort: { amount: -1 } }

]

)

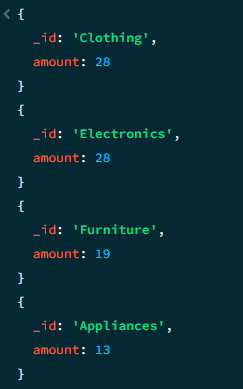


Рисунок 6.13 – Результат виконання запиту

1. Розрахуйте середню ціну товарів у кожній категорії. Використайте

$group для підрахунку середньої ціни товарів у кожній категорії.

db.products.aggregate( [

{

$group: {

\_id: '$category',

averagePrice: { $avg: '$price' }

}

}

]

)

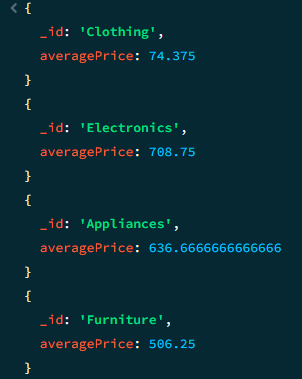


Рисунок 6.14 – Результат виконання запиту

1. Знайдіть користувачів, які зробили більше одного замовлення. Використайте $group і $match, щоб знайти клієнтів, які мали більше одного замовлення.

db.orders.aggregate( [

{

$group: {

\_id: '$customerId', orderAmount: { $sum: 1 }

}

},

{ $match: { orderAmount: { $gt: 0 } } },

{

$lookup: {

from: 'customers', localField: '\_id', foreignField: '\_id', as: 'customerData'

}

}

]

)



Рисунок 6.15 – Результат виконання запиту

**Висновок:** під час роботи було вивчено базові принципи MongoDB та засобів агрегації в MongoDB. Серед викликів були труднощі із розумінням логіки певних етапів конвеєеру агрегації, але використання MongoDB Compass допомогло з фомуванням запитів завдяки поетапній візуалізації агрегації. До найкращих підходів оптимізації відносять індексацію та коригування логіки запиту для уникання роботи з зайвими даними.