Ministry of Higher Education and Scientific Research
University of Mila
Institute of Mathematics and Computer Science
Department of Computer Science
Master 2 I2A – Big Data
2025/2026

Practical Works 7 - Advanced MongoDB : Analytics & Performance

Prerequisites:

- Google Colab or Local Python Environment (`pymongo`).
- The `techstore_db` from TP 6.

Part 1: Data Generation (Simulating Big Data)

To test performance, we need more than 3 documents. We will generate 10,000 orders using Python.
Run this script to populate your database:

```python
from pymongo import MongoClient
import random
import datetime

client = MongoClient("YOUR_CONNECTION_STRING") # Update this!
db = client['techstore_db']

# Clean start
db.orders.drop()

products = [
    {"id": "p1", "name": "Laptop", "price": 1200},
    {"id": "p2", "name": "Mouse", "price": 20},
    {"id": "p3", "name": "Keyboard", "price": 50},
    {"id": "p4", "name": "Monitor", "price": 300},
    {"id": "p5", "name": "Headset", "price": 80}
]

bulk_orders = []
print("Generating 10,000 orders... (Please wait)")

for i in range(10000):
    # Pick random product and quantity
    prod = random.choice(products)
    qty = random.randint(1, 3)

    order = {
        "user_id": random.randint(100, 500), # Random user ID
        "order_date": datetime.datetime(2023, random.randint(1, 12), random.randint(1, 28)),
        "status": random.choice(["completed", "pending", "cancelled"]),
        "items": [
            {
                "product_id": prod["id"],
                "name": prod["name"], # Snapshot pattern
                "price": prod["price"],
                "qty": qty
            }
        ],
        "total_amount": prod["price"] * qty
    }
    bulk_orders.append(order)

# Bulk Insert is much faster than loop insert
db.orders.insert_many(bulk_orders)
print("✅ Database populated with 10,000 orders.")
```

Part 2: Aggregation Pipeline (Analytics)

Write a Python script to answer the following questions using `db.orders.aggregate(pipeline)`.

Task A: Sales by Product Calculate the total revenue generated by each product name.

- *Pipeline:* `$unwind` items -> `$group` by name -> `$sum` the total.
- *Output:* Print the top 3 best-selling products.

Task B: Monthly Sales Report Calculate total sales for each month in 2023.

- *Pipeline:* `$project` (to extract month) -> `$group` by month -> `$sort`.

Part 3: The Performance Lab (Indexing)

We will simulate a "Slow Query" and fix it.

1. The Slow Query: We want to find all **"completed"** orders with a total amount **greater than 1000**.

```
import time
def run_benchmark():
    start_time = time.time()

    # Query: status="completed" AND total_amount > 1000
    query = { "status": "completed", "total_amount": { "$gt": 1000 } }

    # We use list() to force Python to actually fetch all data
    results = list(db.orders.find(query))

    end_time = time.time()
    print(f"Found {len(results)} docs in {end_time - start_time:.4f} seconds")

print("--- RUN 1: No Index ---")
run_benchmark()
```

2. Analyze: Go to MongoDB Compass (or use `.explain()` in code) to see that this query used a `COLLSCAN` (Collection Scan). It had to check all 10,000 documents.

3. Apply Index: Create a compound index to optimize this query based on the ESR rule (Equality first, Range last).

```
# Create your index here using db.orders.create_index(...)
# Tip: Equality on "status", Range on "total_amount"
```

4. Re-Run: Run the benchmark function again.

- Did the time improve?
- Even with 10k documents, the difference might be small (milliseconds), but imagine this on 10 million rows!

Part 4: Visualization (Bonus)

Use `matplotlib` to graph the result of your **Task B (Monthly Sales)** aggregation.

```
import matplotlib.pyplot as plt
# 1. Run Aggregation
# ... (your code from Part 2 Task B) ...
# results = list(db.orders.aggregate(pipeline))

# 2. Extract Data
months = [r['_id'] for r in results]
sales = [r['total'] for r in results]

# 3. Plot
plt.bar(months, sales, color='skyblue')
plt.title('Monthly Sales 2023')
plt.xlabel('Month')
plt.ylabel('Revenue (DA)')
plt.show()
```