Київський національний університет
Імені Тараса Шевченка.
Кафедра: Мережевих та інтернет-технологій.

Лабораторна робота №7

з дисципліни: Бази даних та інформаційні системи

На тему:«Поглиблене вивчення MongoDB: оптимізація продуктивності, використання шардінгу та реплікації, інтеграція з Pandas та Machine Learning”

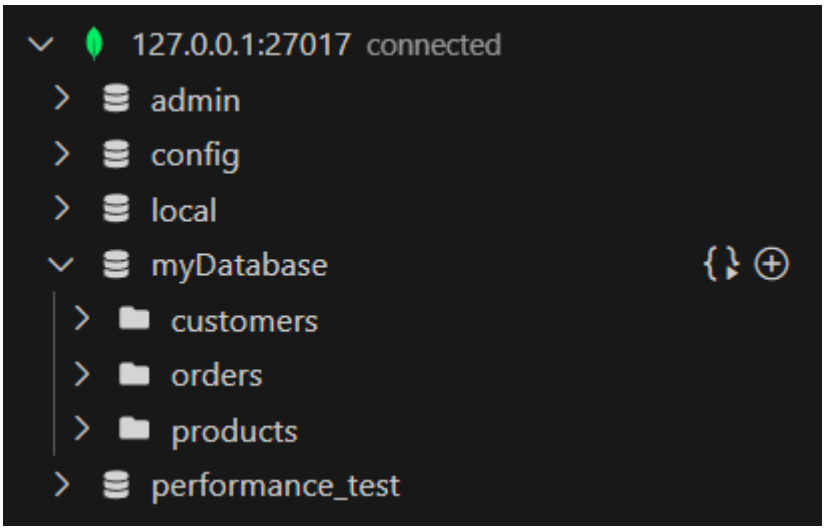
Студента 3 курсу:
Групи МІТ-31
Мулико Володимира

Київ - 2025р.

Хід роботи

Частина 1: Оптимізація продуктивності запитів.

- Підключаємося через mongo у VScode за строкою: *mongodb://127.0.0.1:27017*



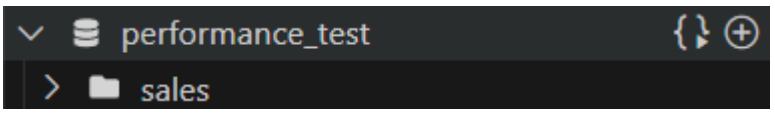
- Створюємо нову базу: *performance_test* та колекцію у цій базі даних: *sales (робимо це вручну, змінюючи просто назву:*

```
const database = 'NEW_DATABASE_NAME';
const collection = 'NEW_COLLECTION_NAME';
```

на

```
const database = 'performance_test';
const collection = 'sales';
```

у нас виконався запрос і створилося все що потрібно:



- Далі створюємо файл *insert_data.py*

```
D:\Learning\Бази Даних\lab7\insert_data.py
import random
import datetime

client = MongoClient("mongodb://localhost:27017")
db = client["performance_test"]
collection = db["sales"]

categories = ["Electronics", "Clothing", "Books", "Home", "Sports"]

documents = [
    {
        "customer_id": random.randint(1, 1000),
        "category": random.choice(categories),
        "amount": round(random.uniform(5, 500), 2),
        "timestamp": datetime.datetime(2024, random.randint(1, 12), random.randint(1, 28))
    }
    for _ in range(100000)
]

collection.insert_many(documents)
print("Дані успішно вставлені.")
```

Ми запустили файл і отримали:

```
PS D:\Learning\Бази Даних\lab7>
a.py"
Дані успішно вставлені.
```

- Створюємо файл *query_no_index.py*

```python
from pymongo import MongoClient
import time

client = MongoClient("mongodb://localhost:27017")
db = client["performance_test"]
collection = db["sales"]

start = time.time()
results = list(collection.find({"category": "Electronics"}))
end = time.time()

print(f"Time taken: {end - start:.6f} seconds")
print(f"Documents found: {len(results)}")
```

Запускаємо і отримуємо вивід:

```
PS D:\Learning\Бази Даних\lab7>
ndex.py"
Time taken: 0.273221 seconds
Documents found: 39752
```

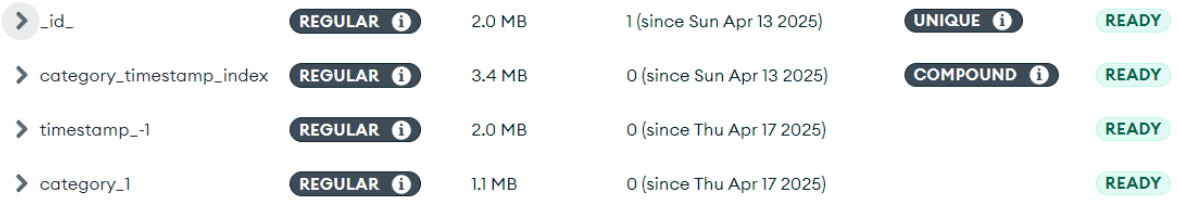
- Створюємо файл *create_index.py*

```python
from pymongo import MongoClient

client = MongoClient("mongodb://localhost:27017")
db = client["performance_test"]
collection = db["sales"]

collection.create_index("category")
print("Індекс створено.")
```

Запускаємо і отримуємо вивід:

```
PS D:\Learning\Бази Даних\lab7>
ex.py"
Індекс створено.
```

- Далі відкриваємо MongoDBcompass і створюємо нові індекси вручну: category, timestamp



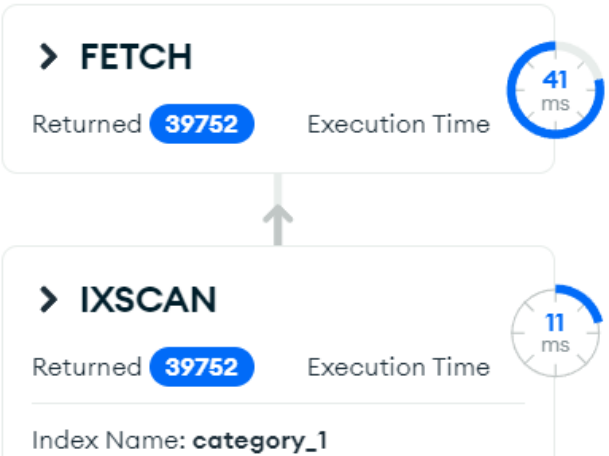
Переходимо на вкладку: documents та вводимо запрос:
{ "category": "Electronics" }

```
_id: ObjectId('67ee7c1a692a74978f581b9d')
customer_id : 248
category : "Electronics"
amount : 490.86094739215963
timestamp : 2024-06-09T00:00:00.000+00:00
```

```
_id: ObjectId('67ee7c1a692a74978f581ba0')
customer_id : 460
category : "Electronics"
amount : 260.1663963005491
timestamp : 2024-07-04T00:00:00.000+00:00
```

Бачимо такий результат, значить дані по моєму запросу були знайдені.

- Відкриваємо Explain Plan Tree і бачимо запроси і отримання час отримання даних:



- Запуск 3 екземплярів MongoDB (перед цим необхідно створити три папки, щоб туди зберігалися дані, які будуть запущені і використані)
    1. *mongod --replSet rs0 --port 27017 --dbpath D:\Learning\DataBase\lab7\rs0-1 --bind_ip localhost*
    2. *mongod --replSet rs0 --port 27018 --dbpath D:\Learning\DataBase\lab7\rs0-2 --bind_ip localhost*
    3. *mongod --replSet rs0 --port 27019 --dbpath D:\Learning\DataBase\lab7\rs0-3 --bind_ip localhost*
- Ініціалізація реплікації (процес автоматичного копіювання та синхронізації даних з одного вузла (Primary) на інші вузли (Secondary) у кластері)
  Підключаємося до першого порта.

```
C:\Users\Vovam>mongosh --port 27018
Current Mongosh Log ID: 6800f5251eaba8f875b71235
Connecting to:          mongodb://127.0.0.1:27018/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2
.4.2
Using MongoDB:          8.0.5
Using Mongosh:          2.4.2
mongosh 2.5.0 is available for download: https://www.mongodb.com/try/download/shell

For mongosh info see: https://www.mongodb.com/docs/mongodb-shell/

------
   The server generated these startup warnings when booting
   2025-04-17T15:31:14.578+03:00: Access control is not enabled for the database. Read and write access to data and conf
iguration is unrestricted
------
```

Виконуємо команди:
rs.initiate()
rs.add("localhost:27018")
rs.add("localhost:270219")
rs.status()

- Перевірка реплікації

```
name: 'localhost:27020',
health: 1,
state: 2,
stateStr: 'SECONDARY',
```

```
name: 'localhost:27019',
health: 1,
state: 2,
stateStr: 'SECONDARY',
```

- Шардінг (це процес збереження записів даних на декількох машинах).

Виконуємо команди:
⇒*mongod --shardsvr --replSet shard1 --port 27021 --dbpath D:\Learning\DataBase\lab7\shard1 --bind_ip localhost*

⇒mongod --shardsvr --replSet shard1 --port 27022 --dbpath D:\Learning\DataBase\lab7\shard2 --bind_ip localhost

- Відкрити локально через базовий шардінг нам не вдалося, тому спробуймо через докер хаб. Ми написали docker.file

```
# Конфігураційні сервери
  configsvr1:
    image: mongo:latest
    container_name: configsvr1
      command: ["mongod", "--replSet", "configReplSet", "--configsvr",
"--bind_ip_all"]
    ports:
      - "26050:27017"
    volumes:
      - configsvr1_data:/data/db

  configsvr2:
    image: mongo:latest
    container_name: configsvr2
      command: ["mongod", "--replSet", "configReplSet", "--configsvr",
"--bind_ip_all"]
    ports:
      - "26051:27017"
    volumes:
      - configsvr2_data:/data/db

  configsvr3:
    image: mongo:latest
    container_name: configsvr3
      command: ["mongod", "--replSet", "configReplSet", "--configsvr",
"--bind_ip_all"]
    ports:
```

```
      - "26052:27017"
    volumes:
      - configsvr3_data:/data/db

  # Шард 1
  shard1:
    image: mongo:latest
    container_name: shard1
       command: ["mongod", "--replSet", "shard1ReplSet", "--shardsvr",
"--bind_ip_all"]
    ports:
      - "27018:27017"
    volumes:
      - shard1_data:/data/db

  # Шард 2
  shard2:
    image: mongo:latest
    container_name: shard2
       command: ["mongod", "--replSet", "shard2ReplSet", "--shardsvr",
"--bind_ip_all"]
    ports:
      - "27019:27017"
    volumes:
      - shard2_data:/data/db

  # Шард 3
  shard3:
    image: mongo:latest
    container_name: shard3
       command: ["mongod", "--replSet", "shard3ReplSet", "--shardsvr",
"--bind_ip_all"]
    ports:
      - "27020:27017"
    volumes:
      - shard3_data:/data/db

  # Mongos (Маршрутизатор)
  mongos:
    image: mongo:latest
    container_name: mongos
```

```
                        command:       ["mongos",       "--configdb",
"configReplSet/configsvr1:27017,configsvr2:27017,configsvr3:27017",
"--bind_ip_all"]
    ports:
      - "27017:27017"
    depends_on:
      - configsvr1
      - configsvr2
      - configsvr3
      - shard1
      - shard2
      - shard3

volumes:
  configsvr1_data:
  configsvr2_data:
  configsvr3_data:
  shard1_data:
  shard2_data:
  shard3_data:
```

Завантажуємо його командою: docker-compose up -d.

- Далі можемо запустити сервер і вже створити шардінг.
  sh.addShard("shard1ReplSet/shard1:27017")
  sh.addShard("shard2ReplSet/shard2:27017")
  sh.addShard("shard3ReplSet/shard3:27017")

  і перевірити командою: *sh.status()*
  І бачимо результат:

```
shards:
  [
    {
      _id: "shard1ReplSet",
      host: "shard1ReplSet/shard1:27017",
      state: 1
    },
    {
      _id: "shard2ReplSet",
      host: "shard2ReplSet/shard2:27017",
      state: 1
    },
    {
      _id: "shard3ReplSet",
      host: "shard3ReplSet/shard3:27017",
      state: 1
    }
```


Шардинг успішно доданий і готовий до використання.

Висновок: У результаті виконання лабораторної роботи було створено локальну базу MongoDB, згенеровано 100 000 документів та проаналізовано продуктивність запитів до колекції з і без індексів. Було налаштовано реплікацію з трьома вузлами та перевірено автоматичне перемикання Primary. Також реалізовано шардінг для підвищення масштабованості системи, що дозволило ефективно розподіляти навантаження між кількома шард-серверами.