Київський національний університет ім. Тараса Шевченка

Кафедра мережевих та інтернет технологій

Лабораторна робота № 7

Дисципліна: бази даних та інформаційні системи

Тема: Поглиблене вивчення MongoDB: оптимізація продуктивності, використання шардінгу та реплікації, інтеграція з Pandas та Machine Learning

Виконав: Студент групи МІТ-31

Пугач Назар

Мета: Ознайомитися з методами підвищення продуктивності MongoDB. Навчитися використовувати індекси для оптимізації запитів. Дослідити механізми реплікації та шардінгу. Використати Pandas для роботи з великим обсягом даних у MongoDB. Реалізувати просту ML-модель з використанням MongoDB як джерела даних.

Хід роботи

Частина 1: Оптимізація продуктивності запитів.

1. **Створення локальної бази MongoDB**
   Встановити MongoDB та розширення MongoDB for VS Code.
   Підключитися до MongoDB через MongoDB Explorer.
   Створити базу performance_test та колекцію sales.

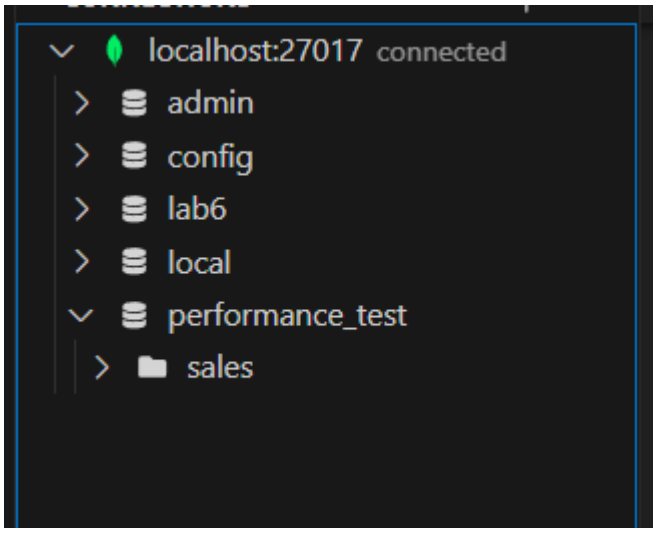

Рисунок 1- База даних та колекція

2. **Генерація та вставка тестових даних**
   Написати Python-скрипт insert_data.py для генерації 100 000 документів.
   Виконати скрипт та переконатися, що дані вставлено.

```
from pymongo import MongoClient
```

```python
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),  # Випадковий ідентифікатор покупця (1-1000)
        "category": random.choice(categories),   # Випадкова категорія з `categories`
        "amount": random.uniform(5, 500),       # Випадкова сума покупки (від $5 до $500)
        "timestamp": datetime.datetime(2024, random.randint(1, 12), random.randint(1, 28))
        # Випадкова дата у 2024 році (будь-який місяць і день до 28)
    }
    for _ in range(100000)  # Генерація 100 000 документів
]

collection.insert_many(documents)
```

Рисунок 2 – Скрипт для вставка 100к значень в БД

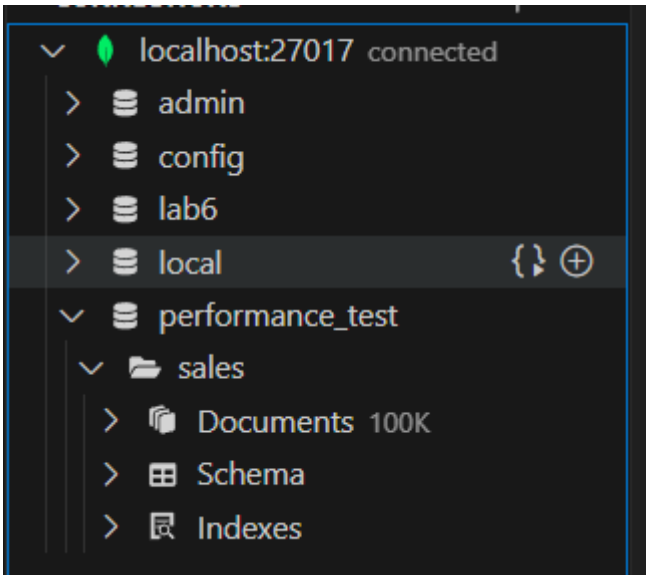

Рисунок 3 – Результат роботи скрипта

3. **Оптимізація запитів за допомогою індексів**
   Виконати пошуковий запит без індексу та виміряти час виконання (query_no_index.py).

Створити індекс на полі category (create_index.py).
Повторно виконати запит та порівняти час виконання.
Дослідити складений індекс (category, timestamp) та його вплив на продуктивність.

```python
from pymongo import MongoClient
from time import time

client = MongoClient("mongodb://localhost:27017")

db = client["performance_test"]

collection = db["sales"]

start_time = time()

data = list(collection.find({"category": "Electronics"}))

end_time = time()

print(f"Time taken: {end_time-start_time:.4f}")
```

Рисунок 4 – Скрипт для пошуку відповідних документів

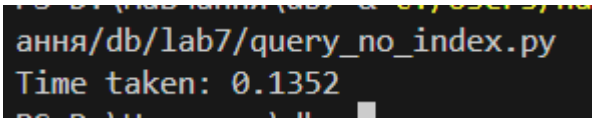

Рисунок 5 – Час на пошук без індексу

```python
from pymongo import MongoClient
from time import time

client = MongoClient("mongodb://localhost:27017")

db = client["performance_test"]

collection = db["sales"]

# collection.drop_index("category_1")

collection.create_index([("category", 1)])
```

Рисунок 6 – Створення індексу

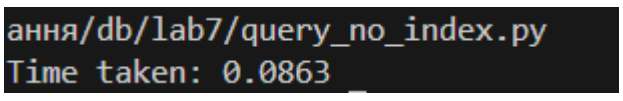

Рисунок 7 – Час на пошук з індексом

Частина 2: Налаштування реплікації у MongoDB

2. Запустити три екземпляри MongoDB у режимі реплікації.

   mongod --dbpath D:\mongodb\27018 --port 27018 --replSet "rs0"

mongod --dbpath D:\mongodb\27019 --port 27019 --replSet "rs0"

mongod --dbpath D:\mongodb\27020 --port 27020 --replSet "rs0"

3. Ініціалізувати реплікаційний набір та додати вузли.

   rs.initaite()

   rs.add("localhost:27019")

   rs.add("localhost:27020")

   rs.status()

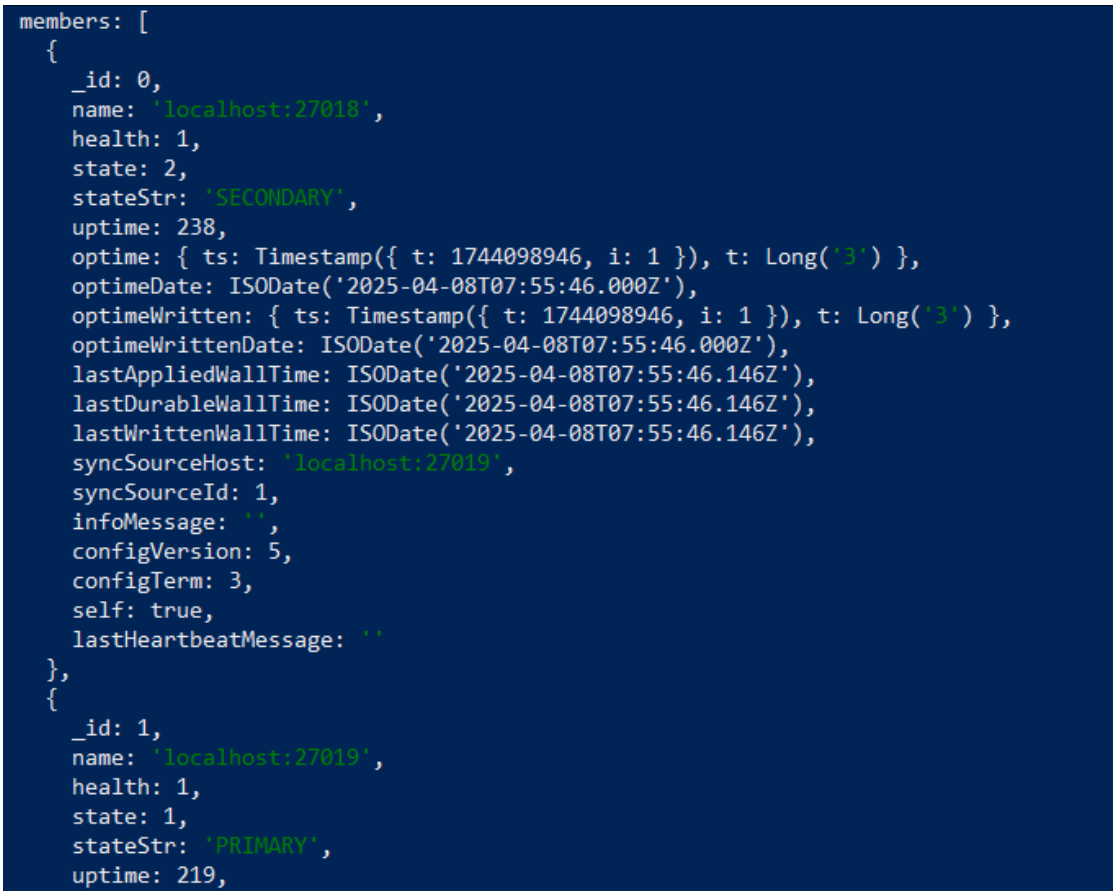

Рисунок 8 – Налаштована реплікація

4. Виконати запис у Primary та перевірити доступність даних на Secondary-вузлах.

Рисунок 9 – Створення бази даних на основному вузлі

```
rs0 [direct: secondary] test> use new_database
switched to db new_database
rs0 [direct: secondary] new_database> show collections
person
rs0 [direct: secondary] new_database>
```

Рисунок 10 – Дані наявні на другорядному вузлі

5. Відключити Primary та дослідити поведінку системи.

   Після відключення основного вузла, один з другорядних стає основним

```
members: [
  {
    _id: 0,
    name: 'localhost:27018',
    health: 1,
    state: 1,
    stateStr: 'PRIMARY',
    uptime: 766,
    optime: { ts: Timestamp({ t: 1744099470, i: 1
    optimeWritten: { ts: Timestamp({ t: 174409947
    optimeWrittenDate: ISODate('2025-04-08T08:04:
    lastAppliedWallTime: ISODate('2025-04-08T08:0
    lastDurableWallTime: ISODate('2025-04-08T08:0
    lastWrittenWallTime: ISODate('2025-04-08T08:0
    syncSourceHost: '',
    syncSourceId: -1,
    infoMessage: '',
    electionTime: Timestamp({ t: 1744099440, i: 1
    electionDate: ISODate('2025-04-08T08:04:00.00
    configVersion: 5,
    configTerm: 4,
    self: true,
    lastHeartbeatMessage: ''
  },
  {
    _id: 1,
    name: 'localhost:27019',
    health: 0,
    state: 8,
    stateStr: '(not reachable/healthy)',
```

Рисунок 11 – Після результат відключення основного вузла

Частина 3: Реалізація шардінгу у MongoDB

1. Створити конфігураційний репліка сервер.

   mongod --configsvr --replSet configReplSet --port 26050 --dbpath D:\mongodb\config1 --bind_ip localhost --logpath D:\mongodb\config1\log.txt

   mongod --configsvr --replSet configReplSet --port 26051 --dbpath D:\mongodb\config2 --bind_ip localhost --logpath D:\mongodb\config2\log.txt

mongod --configsvr --replSet configReplSet --port 26052 --dbpath D:\mongodb\config3 --bind_ip localhost --logpath D:\mongodb\config3\log.txt

```
C:\Users\Назар>mongosh --port 26050
Current Mongosh Log ID: 67f4fae7d1e70e3ff6b71235
Connecting to:          mongodb://127.0.0.1:26050/?directConnection=true&serv
Using MongoDB:          8.0.5
Using Mongosh:          2.4.2

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

------
   The server generated these startup warnings when booting
   2025-04-08T13:29:14.602+03:00: Access control is not enabled for the datab
------

test> rs.initiate({
... _id: "configReplSet",
... configsvr: true,
... members:[
... {_id: 0, host: "localhost:26050"},
... {_id: 1, host: "localhost:26051"},
... {_id: 2, host: "localhost:26052"}
... ]
... })
```

Рисунок 12 – Конфігураційний репліка сервер

2. Запустити два екземпляри MongoDB у режимі шардів.

   mongod --shardsvr --replSet shardRepl1 --port 27018 --dbpath D:\mongodb\shard1 --bind_ip localhost --logpath D:\mongodb\shard1\log.txt

   mongod --shardsvr --replSet shardRepl2 --port 27019 --dbpath D:\mongodb\shard2 --bind_ip localhost --logpath D:\mongodb\shard2\log.txt

3. Додати шарди через MongoDB Shell.

```
test> rs.initiate({ _id: "shardRepl1", members: [ { _id: 0, host: "localhost:27018" }] })
{
  ok: 1,
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1744109392, i: 1 }),
    signature: {
      hash: Binary.createFromBase64('AAAAAAAAAAAAAAAAAAAAAAAAAAA=', 0),
      keyId: Long('0')
    }
  },
  operationTime: Timestamp({ t: 1744109392, i: 1 })
}
shardRepl1 [direct: secondary] test>
```

Рисунок 13 – Налаштування реплікація на шардах.

4. Запуск роутера

mongos --configdb configReplSet/localhost:26050,localhost:26051,localhost:26052 --port 27020 --bind_ip localhost --logpath D:\mongodb\mongos\log.txt

sh.addShard("shardRepl1/localhost:27018")

sh.addShard("shardRepl2/localhost:27019")

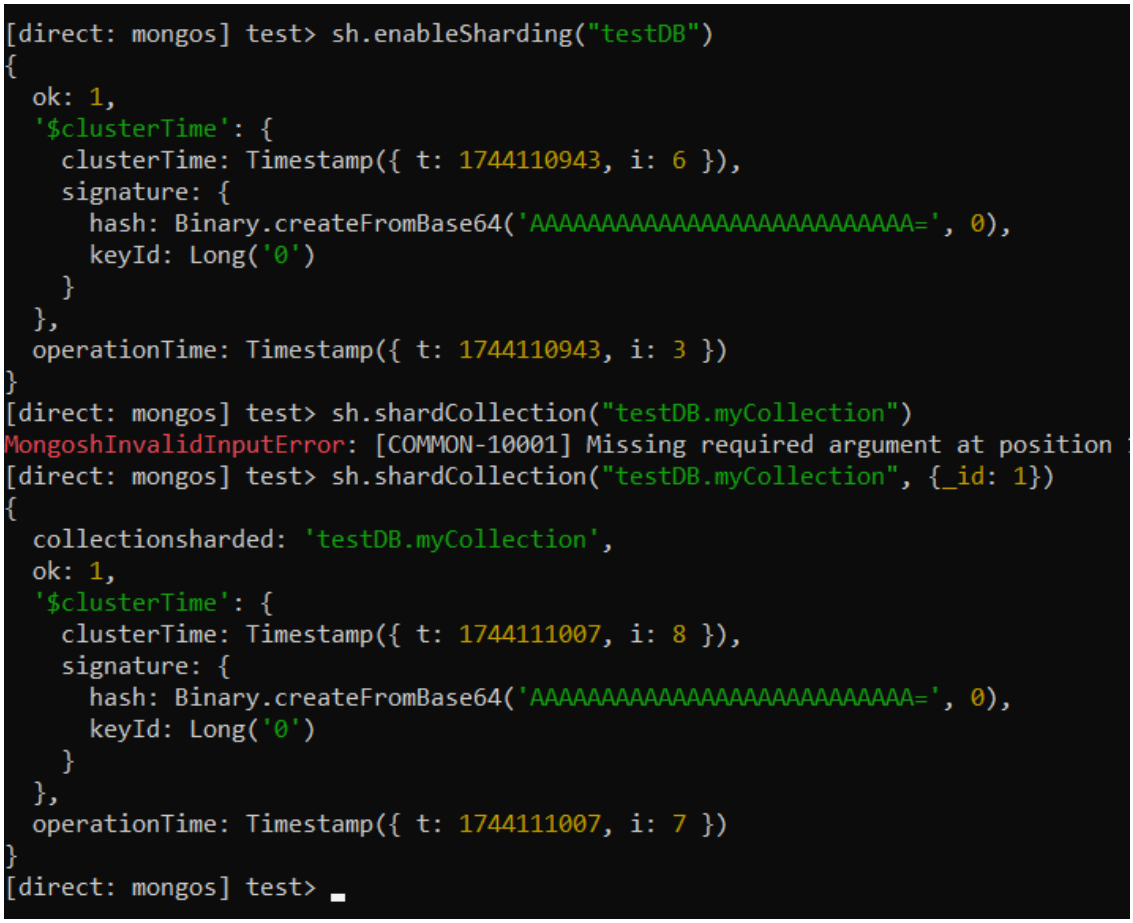

Рисунок 14 – Увімкнення шардінгу

Висновок: Під час виконання лабораторної роботи я ознайомився з методами підвищення продуктивності MongoDB. Навчитися використовувати індекси для оптимізації запитів. Дослідити механізми реплікації та шардінгу.