

Лабораторна робота №7

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

З дисципліни: Базы даних та інформаційні системи

Студента групи МІТ-31: Ларіонова С.О.

Хід роботи

Частина 1: Оптимізація продуктивності запитів

Для початку роботи було встановлено розширення MongoDB for VS Code.

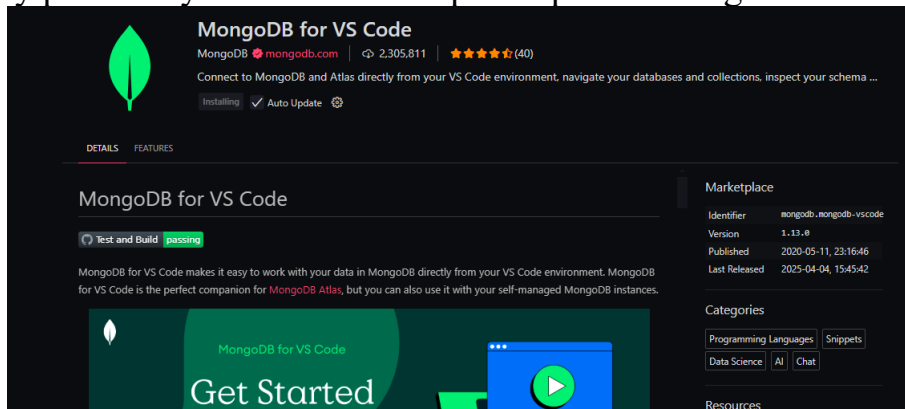


Рисунок 7.1 – Встановлення розширення

Далі було встановлено підключення до сервера MongoDB.

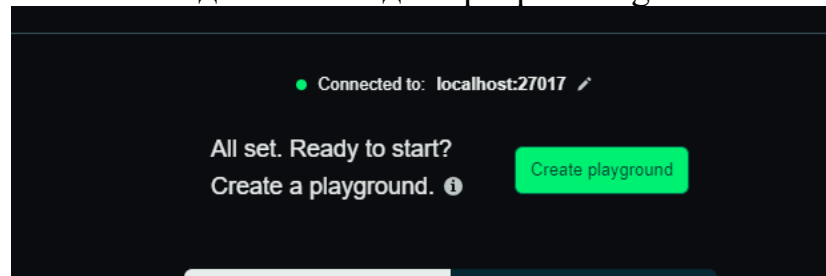


Рисунок 7.2 – Підключення встановлено

Наступним кроком було створення колекції sales у бд performance test.

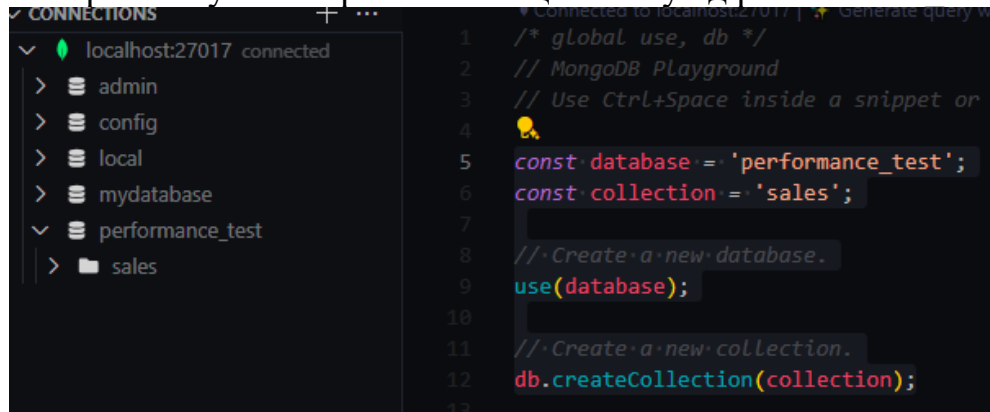


Рисунок 7.3 – Створення нової колекції

Далі було створено та запущено скрипт на Python для генерації та вставки 100000 документів до колекції sales.

```
from pymongo import MongoClient
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) # Вставка документів у колекцію
```

Рисунок 7.4 – Скрипт для генерації документів

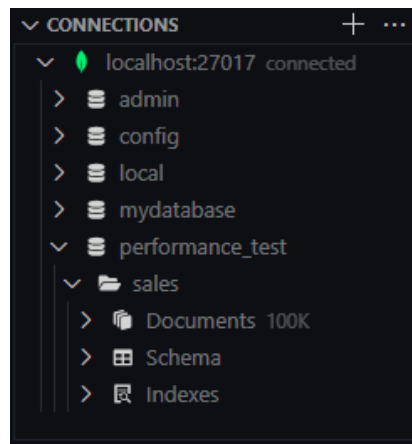


Рисунок 7.5 – У колекції sales 100000 документів

Далі було створено скрипт query_no_index.py що вимірює час запиту для отримання всіх документів sales, де значення поля Category – Electronics.

```
from pymongo import MongoClient
import time

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

time_start = time.time()

electronics_sales = collection.find({"category": "Electronics"}).to_list()

time_end = time.time()
time_diff = time_end - time_start
print(f"Time taken to query sales in Electronics category: {time_diff:.2f} seconds")
```

Рисунок 7.6 – Код скрипту

Time taken to query sales in Electronics category: 0.22 seconds

Рисунок 7.7 – Результат виконання

Після цього було створено та запущено скрипт create_index.py для створення індекси для категорій та знову запущено query_no_index.py для порівняння швидкості.

```
from pymongo import MongoClient

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

collection.create_index([("category", 1)]) # Створення індексу на поле category
print("Index on 'category' field created successfully.")
```

Рисунок 7.8 – Код create_index.py

Time taken to query sales in Electronics category: 0.16 seconds

Рисунок 7.9 – Швидкість виконання запиту після додавання індексу

Індексація прискорила роботу запиту приблизно на 27%.

Далі було створено складений індекс (category, timestamp) з використанням MongoDB Compass. Після цього було виконано запит {"category": "Electronics"} та переглянуто вкладку Explain.

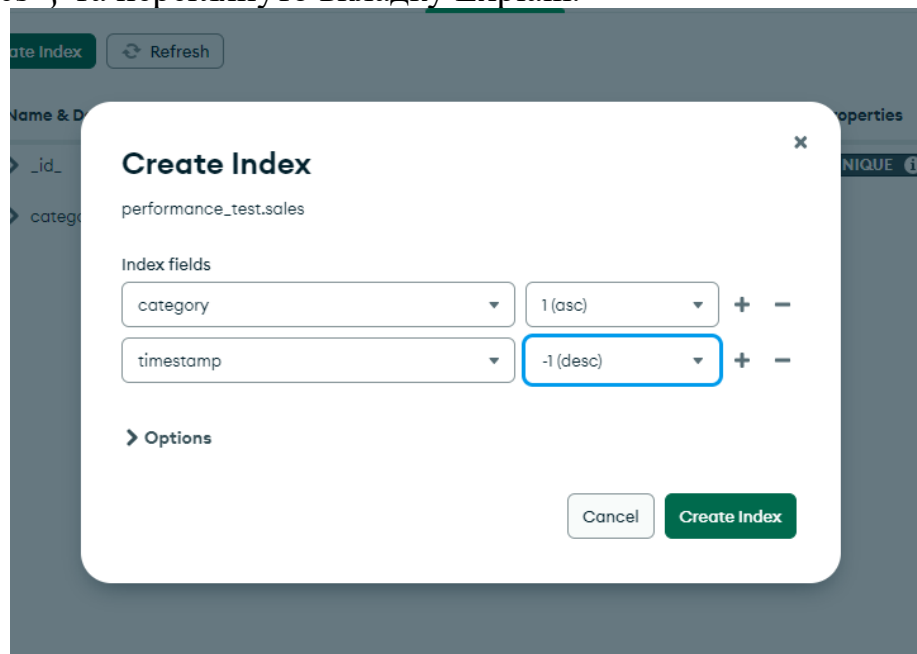


Рисунок 7.10 - Створення складеного індексу у MongoDB Compass

Explain Plan

Explain provides key execution metrics that help diagnose slow queries and optimize index usage. [Learn more](#)

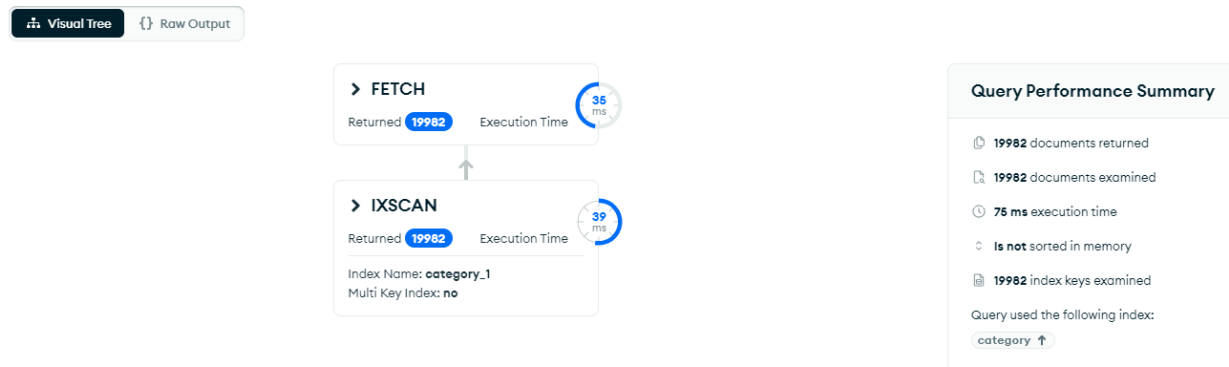


Рисунок 7.11 – Вкладка Explain. Використовується попередньо створений індекс category_1

Як бачимо, новий, складений індекс не було застосовано. Щоб використовувався саме він можна видалити попередній.

Explain Plan

Explain provides key execution metrics that help diagnose slow queries and optimize index usage. [Learn more](#)

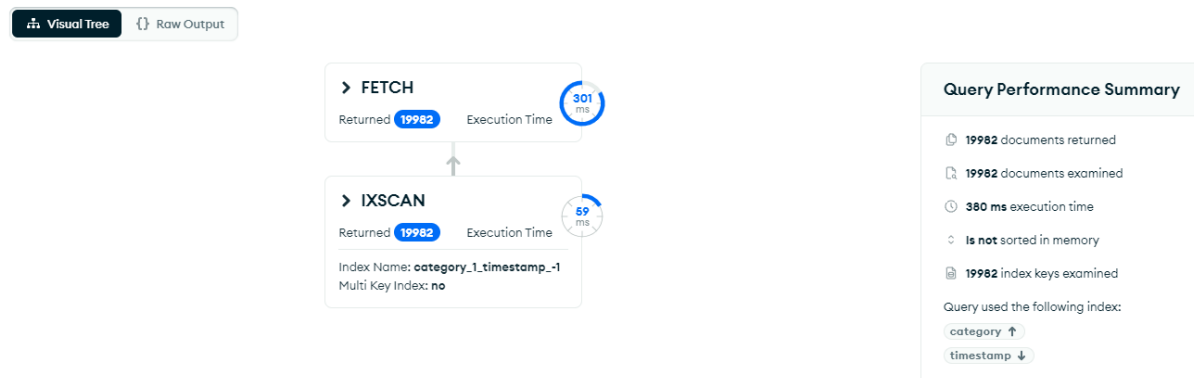


Рисунок 7.12 – Після видалення старого індекса застосовується складений

Частина 2: Налаштування реплікації

Для початку роботи було запущено три екземпляри MongoDB в різних терміналах з параметром --replSet. Далі було виконано підключення до 1шого з них за допомогою MongoDB Compass, після чого виконано команди ініціалізації реплікації та додавання інших екземплярів.

```
rs.initiate()
{
  info2: 'no configuration specified. Using a default configuration for the set',
  me: 'localhost:27017',
  ok: 1,
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1745607620, i: 1 }),
    signature: {
      hash: Binary.createFromBase64('AAAAAAAAAAAAAAAAAAAAAAAAAA=', 0),
      keyId: Long('0')
    }
  },
  operationTime: Timestamp({ t: 1745607620, i: 1 })
}
rs.add("localhost:27018")
{
  ok: 1,
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1745607638, i: 1 }),
    signature: {
      hash: Binary.createFromBase64('AAAAAAAAAAAAAAAAAAAAAAAAAA=', 0),
      keyId: Long('0')
    }
  },
  operationTime: Timestamp({ t: 1745607638, i: 1 })
}
rs.add("localhost:27019")
{
  ok: 1,
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1745607714, i: 1 }),
```

Рисунок 7.13 – Команди налаштування реплікації

```
> rs.status().members.map(m => ({ name: m.name, state: m.stateStr}))
< [
  { name: 'localhost:27017', state: 'PRIMARY' },
  { name: 'localhost:27018', state: 'SECONDARY' },
  { name: 'localhost:27019', state: 'SECONDARY' }
]
```

Рисунок 7.14 – Перевірка роботи кластеру реплікації

Після відключення екземпляра зі статусом “PRIMARY”, цей статус переходить до іншого.

```
rs.status().members.map(m => ({ name: m.name, state: m.stateStr}))
[
  { name: 'localhost:27017', state: '(not reachable/healthy)' },
  { name: 'localhost:27018', state: 'PRIMARY' },
  { name: 'localhost:27019', state: 'SECONDARY' }
]
```

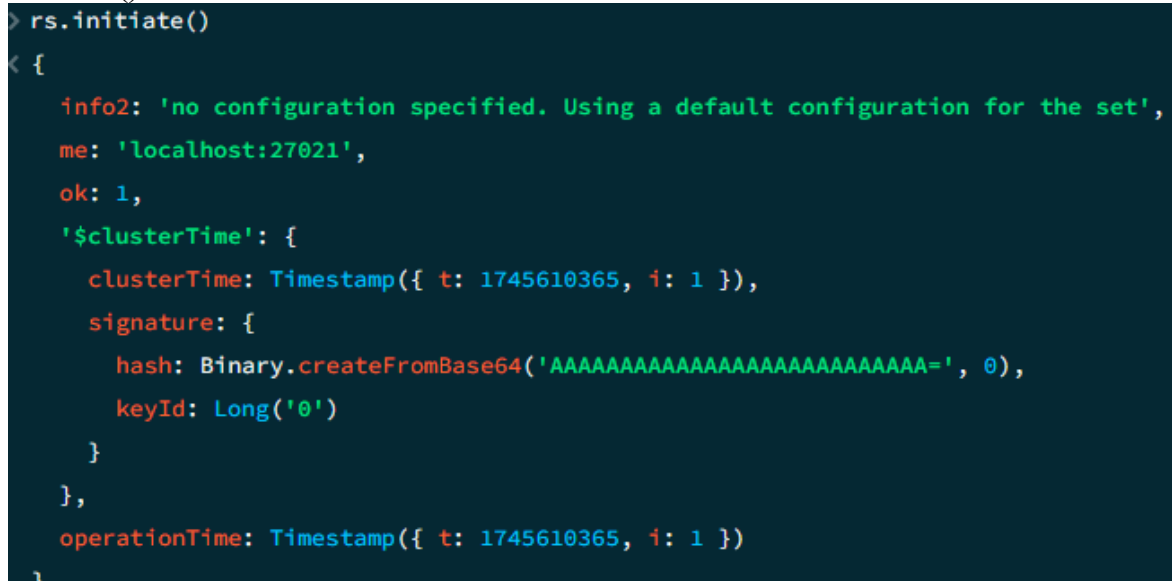
Рисунок 7.15 – Оновлений статус

Частина 3: Реалізація шардінгу

Для початку роботи було запущено ще 2 екземпляри MongoDB:

```
mongod --shardsvr --port 27021 --dbpath D:/data/shard1 --bind_ip localhost --replSet shardSet1
mongod --shardsvr --port 27022 --dbpath D:/data/shard2 --bind_ip localhost --replSet shardSet2
```

Після цього до кожного з них було підключено mongosh та виконано rs.initiate()



```
> rs.initiate()
< {
  info2: 'no configuration specified. Using a default configuration for the set',
  me: 'localhost:27021',
  ok: 1,
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1745610365, i: 1 }),
    signature: {
      hash: Binary.createFromBase64('AAAAAAAAAAAAAAAAAAAAAAAAAA=', 0),
      keyId: Long('0')
    }
  },
  operationTime: Timestamp({ t: 1745610365, i: 1 })
}
```

Рисунок 7.16 – Команда rs.initiate()

Наступний крок – ініціалізація сервера конфігурації mongos. Для цього завдання було запущено ще один екземпляр mongod, ініціалізовано його та запущено mongos:

```
mongod --configsvr --replSet configReplSet --port 27019 --dbpath D:\data\config1 --bind_ip localhost
```

```
mongos --configdb configReplSet/localhost:27019 --bind_ip localhost --port 27016
```

Після цього було виконано підключення mongosh до серверу mongos та додано «шарди».

```

sh.addShard("shardSet1/localhost:27021")
{
  shardAdded: 'shardSet1',
  ok: 1,
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1745611315, i: 14 }),
    signature: {
      hash: Binary.createFromBase64('AAAAAAAAAAAAAAAAAAAAAAAAAA=', 0),
      keyId: Long('0')
    }
  },
  operationTime: Timestamp({ t: 1745611315, i: 14 })
}
sh.addShard("shardSet2/localhost:27022")
{
  shardAdded: 'shardSet2',
  ok: 1,
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1745611324, i: 24 }),
    signature: {
      hash: Binary.createFromBase64('AAAAAAAAAAAAAAAAAAAAAAAAAA=', 0),
      keyId: Long('0')
    }
  },
  operationTime: Timestamp({ t: 1745611324, i: 18 })
}

```

Рисунок 7.17 – Додавання «шардів»

Далі було налаштовано «шардинг» для колекції performance_test.sales.

```

sh.enableSharding("performance_test")
{
  ok: 1,
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1745611409, i: 8 }),
    signature: {
      hash: Binary.createFromBase64('AAAAAAAAAAAAAAAAAAAAAAAAAA=', 0),
      keyId: Long('0')
    }
  },
  operationTime: Timestamp({ t: 1745611409, i: 5 })
}
sh.shardCollection("performance_test.sales", {"category": 1})
{
  collectionsSharded: 'performance_test.sales',
  ok: 1,
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1745611426, i: 2 }),
    signature: {
      hash: Binary.createFromBase64('AAAAAAAAAAAAAAAAAAAAAAAAAA=', 0),
      keyId: Long('0')
    }
  },
  operationTime: Timestamp({ t: 1745611426, i: 1 })
}

```

Рисунок 7.18 – Налаштування «шардингу» для колекції

Для перевірки «шардингу» було виконано команду `sh.status()`.

```
sh.status()

shardingVersion

{
  _id: 1,
  clusterId: ObjectId('680be9577880c11aa8f1ca37')
}

shards

[
  {
    _id: 'shardSet1',
    host: 'shardSet1/localhost:27021',
    state: 1,
    topologyTime: Timestamp({ t: 1745611315, i: 4 }),
    replSetConfigVersion: -1
  },
  {
    _id: 'shardSet2',
    host: 'shardSet2/localhost:27022',
    state: 1,
    topologyTime: Timestamp({ t: 1745611324, i: 9 }),
    replSetConfigVersion: -1
  }
]
```

Рисунок 7.19 – Частина виводу `sh.status()`

Додаткове завдання: візуалізація даних

Для візуалізації даних було створено новий скрипт `sales_per_category_chart.py`, де було використано бібліотеки `pymongo` та `matplotlib` для підключення до бд та візуалізації даних відповідно.

```
import matplotlib.pyplot as plt
from pymongo import MongoClient

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

pipeline = [
  { "$group": { "_id": "$category", "count": { "$sum": 1 } } }
]

results = list(collection.aggregate(pipeline))

categories = [doc["_id"] for doc in results]
sales = [doc["count"] for doc in results]

plt.bar(categories, sales)
plt.xlabel("Category")
plt.ylabel("Total Sales")
plt.title("Sales per Category")
plt.show()
```

Рисунок 7.20 – Код скрипта

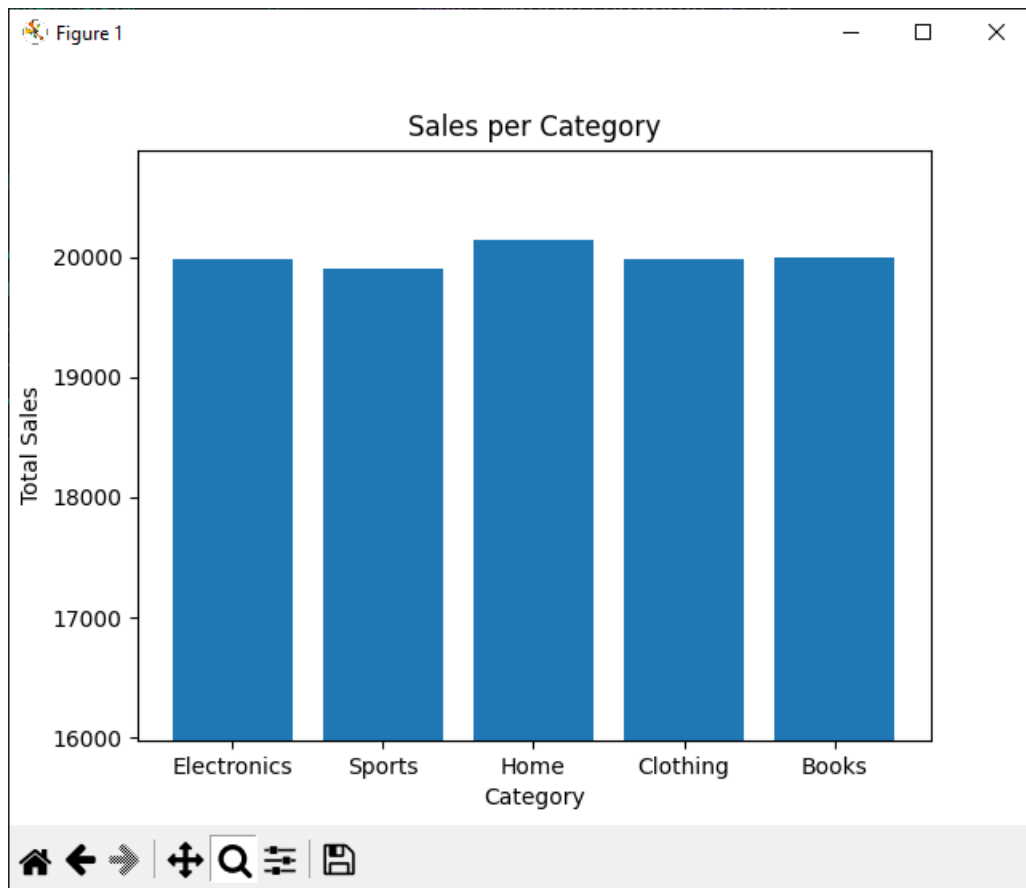


Рисунок 7.21 – Результат запуску програми

Висновок: під час роботи було розглянуто індексацію у MongoDB, реплікації та шардингу. Індєксація – це один із способів оптимізації MongoDB для прискорення запитів. А реплікація, як засіб масштабування, покращує доступність даних та розподіл навантаження. У свою чергу, шардінг допомагає ефективно управляти великими обсягами даних, розподіляючи їх між вузлами.