Лабораторна робота №8

Тема: Робота з Redis

З дисципліни: Бази даних та інформаційні системи
Студента групи МІТ-31: Ларіонова С.О.

Хід роботи

1. **Встановлення Redis та запуск**

Для початку роботи було запущено Docker контейнер “redis”. Після цього було виконано підключення до redis-cli всередині контейнера та виконано команду PING для перевірки роботи redis.

```
C:\Users\Max>docker run -d --name redis -p 6379:6379 redis
3e36a1aa18b2f21f593c93a300cdbf43710b8524c337618db63ac3f95db36324
```

Рисунок 8.1 – Запуск контейнера

```
C:\Users\Max>docker exec -it redis redis-cli
127.0.0.1:6379> PING
PONG
```

Рисунок 8.2 – Підключення до redis-cli та команда PING

2. **Операції з рядками**

Далі було виконано операції SET та GET для створення ключа та виведення його значення.

```
127.0.0.1:6379> SET student "Maksym"
OK
127.0.0.1:6379>
127.0.0.1:6379> GET student
"Maksym"
```

Рисунок 8.3 – Команди GET та SET

Після цього було використано команду INCR для створення лічильника та збільшення його значення.

```
127.0.0.1:6379> INCR mycounter
(integer) 1
127.0.0.1:6379> INCR mycounter
(integer) 2
127.0.0.1:6379> GET mycounter
"2"
```

Рисунок 8.4 – Команда INCR

3. Структури даних

Наступним кроком стало дослідження роботи списків. Було використано команди: LPUSH – для додавання значення у список; LRANGE – для виведення значень; LPOP для вилучення та виведення даних зі списку.

```
127.0.0.1:6379> LPUSH products "Phone"
(integer) 1
127.0.0.1:6379> LPUSH products "Tablet"
(integer) 2
127.0.0.1:6379> LPUSH products "Laptop"
(integer) 3
```

Рисунок 8.5 – Команда LPUSH

```
127.0.0.1:6379> LRANGE products 0 -1
1) "Laptop"
2) "Tablet"
3) "Phone"
```

Рисунок 8.6 – Команда LRANGE

```
127.0.0.1:6379> LPOP products
"Laptop"
127.0.0.1:6379> LRANGE products 0 -1
1) "Tablet"
2) "Phone"
```

Рисунок 8.7 – Команда LPOP

Далі було досліджено роботу з множинами. Було використано команди: SADD – для додавання елемента до множини; SMEMBERS – для виведення елементів множини; SISMEMBER – для перевірки наявності елемента у множині. На відміну від списків, множини не дозволяють зберігання однакових значень.

```
127.0.0.1:6379> SADD shapes "Circle"
(integer) 1
127.0.0.1:6379> SADD shapes "Square"
(integer) 1
127.0.0.1:6379> SADD shapes "Triangle"
(integer) 1
```

Рисунок 8.8 – Команда SADD

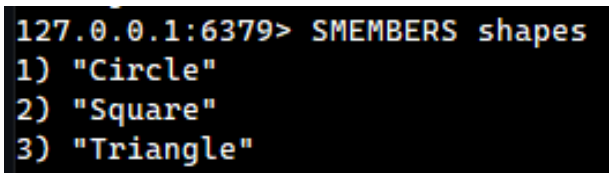

Рисунок 8.9 – Команда SMEMBERS

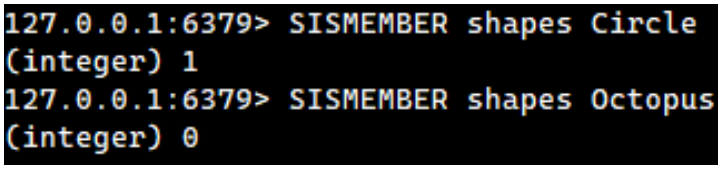

Рисунок 8.10 – Команда SISMEMBER

Далі було виконано дослідження роботи з хешами у Redis. Було використано команди: HSET – для додавання значення до поля хеша; HGET – для виведення конкретного значення; HGETALL – для виведення всіх значень.

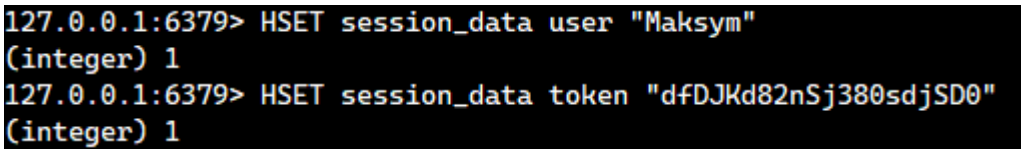

Рисунок 8.11 – Команда HSET

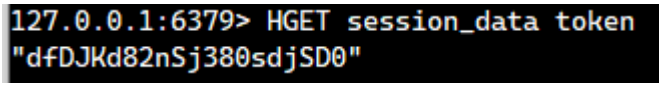

Рисунок 8.12 – Команда HGET

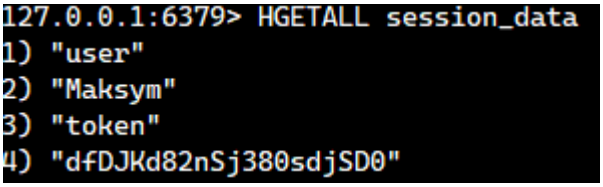

Рисунок 8.13 – Команда HGETALL

Після цього було досліджено використання Sorted Set (відсортована множина). Було використано команди: ZADD – для додавання значення; ZREVRANGE – для виведення ключів (має необов’язковий параметр WITHSCORES для виведення значень); ZRANK – для виведення рангу ключа.

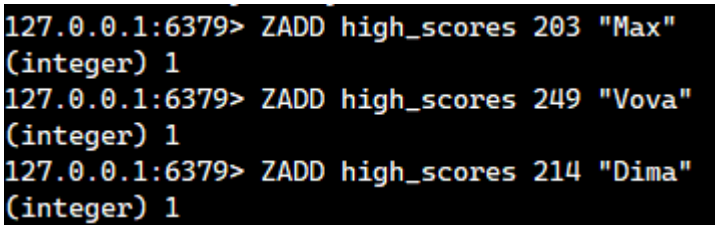

Рисунок 8.14 – Команда ZADD

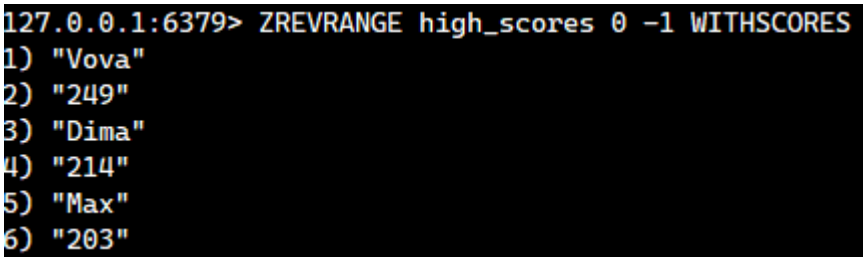

Рисунок 8.15 – Команда ZREVRANGE

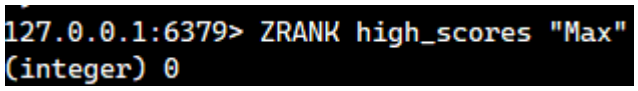

Рисунок 8.16 – Команда ZRANK

4. Робота з TTL

Наступним кроком було дослідження роботи з тимчасовими даними з використанням TTL. Було використано параметр EX для визначення часу існування даних в пам’яті, та команду TTL для виведення часу, що залишився. Після того, як час існування вичерпався TTL повертатиме -2, а GET – (nil).

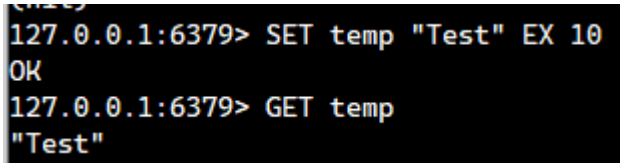

Рисунок 8.17 – Параметр EX

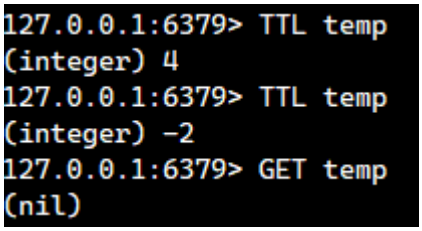

Рисунок 8.18 – Команда TTL

5. Міні-програма з Redis (додатково)

Далі було створено просту програму-месенджер на Python з використанням redis. Програма містить 2 скрипти: перший (message_display.py) – для відображення повідомлень у реальному часі, другий (messenger.py) – клієнт для введення та відправки повідомлень. Повідомлення зберігаються тимчасово (5 сек.) у redis у змінній last_message. Скрипт для відображення повідомлень звертається до бази даних кожні 0.2 сек. для виведення останнього повідомлення

```python
import redis, time

r = redis.Redis(host='localhost', port=6379, decode_responses=True)

last_message = ""
while True:
    message = r.get('last_message') or ""
    if message not in [last_message, ""]:
        print(f"{message}")
        last_message = message
    time.sleep(0.2)
```

Рисунок 8.19 – Код скрипта для відображення повідомлень у реальному часі

```python
import redis

r = redis.Redis(host='localhost', port=6379, decode_responses=True)

username = input("Enter your username: ")

while True:
    msg = input('> ')
    if msg!="":

        msg = f"{username}: {msg}"
        r.set('last_message', msg, ex=5)
```

Рисунок 8.20 – Код скрипта-клієнта

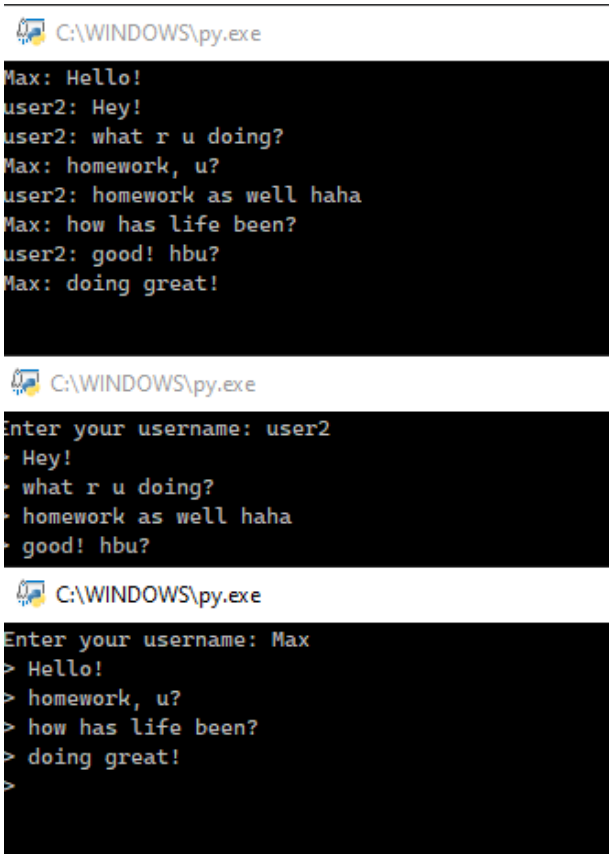

Рисунок 8.21 – Запущена програма. Перше вікно для відображення повідомлень, друге та третє – клієнти

6. **Аналіз даних (теоретичне завдання)**

Де можна використати Redis у реальних проєктах:

- **Для кешування:** популярно у веб-додатках, зокрема для кешування сторінок та API-відповідей. Допомагає покращити швидкість роботи.
- **Для лідербордів та рейтинг систем:** можна тимчасово зберігати дані рейтинг системи з використанням sorted set.
- **Для системи черг:** наприклад, для організації асинхронних черг повідомлень. Корисно для подій, що обробляються у фоновому режимі.
- **Для керування сеансами:** redis можна застосувати для зберігання даних сеансів користувачів у веб-додатках.
- **Для збереження тимчасових даних:** наприклад, для зберігання тимчасових токенів доступу або для зберігання ліміту запитів в API для захисту від перевантаження.

Висновок: під час роботи було розглянуто базові принципи Redis. Було розглянуто як просте збереження значення у ключі, так і роботу із складнішими структурами даних: списками, множинами, хешами та відсортованими множинами.