Нереляционные базы данных и MongoDB

Урок 3. Введение Redis

Оглавление

Задание 1

Цель практической работы:

Научиться выполнять простые запросы в Redis.

Что нужно сделать:

Напишите последовательность команд для Redis:
Создайте ключ index со значением “index precalculated content”.
Проверьте, есть ли ключ index в БД.
Узнайте, сколько ещё времени будет существовать ключ index.
Отмените запланированное удаление ключа index.

Что оценивается:

Верная последовательность команд.

---

1. Создайте ключ index со значением “index precalculated content”.
   ```
   127.0.0.1:6379> set index 'index precalculated content' ex 100
   OK
   ```

2. Проверьте, есть ли ключ index в БД.
   ```
   127.0.0.1:6379> get index
   "index precalculated content"
   ```

3. Узнайте, сколько ещё времени будет существовать ключ index.
   ```
   127.0.0.1:6379> ttl index
   (integer) 87
   ```

4. Отмените запланированное удаление ключа index.
   ```
   127.0.0.1:6379> persist index
   (integer) 1
   127.0.0.1:6379> ttl index
   (integer) -1
   ```

Задание 2

Цель практической работы:

Научиться работать со структурами данных в Redis.

Что нужно сделать:

Напишите последовательность команд для Redis:

Создайте в Redis структуру данных с ключом ratings для хранения следующих значений рейтингов технологий: mysql — 10, postgresql — 20, mongodb — 30, redis — 40.

По этому же ключу увеличьте значение рейтинга mysql на 15.

Удалите из структуры элемент с максимальным значением.

Выведите место в рейтинге для mysql.

Что оценивается:

Верная последовательность команд.

---

1. Создайте в Redis структуру данных с ключом ratings для хранения следующих значений рейтингов технологий: mysql — 10, postgresql — 20, mongodb — 30, redis — 40.

```
127.0.0.1:6379> zadd ratings 10 'mysql' 20 'postgresql' 30 'mongodb' 40 'redis'
        (integer) 4
127.0.0.1:6379> zrange ratings 0 -1 withscores
        1) "mysql"
        2) "10"
        3) "postgresql"
        4) "20"
        5) "mongodb"
        6) "30"
        7) "redis"
        8) "40"
```

2. По этому же ключу увеличьте значение рейтинга mysql на 15.

```
127.0.0.1:6379> ZINCRBY ratings 15 "mysql"
        "25"
127.0.0.1:6379> zrange ratings 0 -1 withscores
        1) "postgresql"
        2) "20"
        3) "mysql"
        4) "25"
        5) "mongodb"
        6) "30"
        7) "redis"
        8) "40"
```

3. Удалите из структуры элемент с максимальным значением.

```
127.0.0.1:6379> ZREMRANGEBYRANK ratings -1 -1
        (integer) 1
127.0.0.1:6379> zrange ratings 0 -1 withscores
        1) "postgresql"
```

2) "20"
3) "mysql"
4) "25"
5) "mongodb"
6) "30"

4. Выведите место в рейтинге для mysql.

127.0.0.1:6379> ZRANK ratings mysql
(integer) 1

Задание 3

Цель практической работы:

Научиться работать с механизмом Pub/Sub в Redis.

Что нужно сделать:

Напишите две команды для СУБД Redis:
Подпишитесь на все события, опубликованные на каналах, начинающихся с events.
Опубликуйте сообщение на канале events101 с текстом “Hello there”.

Что оценивается:

Верная последовательность команд.

---

1. Подпишитесь на все события, опубликованные на каналах, начинающихся с events.

```
127.0.0.1:6379> psubscribe events*
    Reading messages... (press Ctrl-C to quit)
    1) "psubscribe"
    2) "events*"
    3) (integer) 1
    1) "pmessage"
    2) "events*"
    3) "events2"
    4) "Hello!"
    1) "pmessage"
    2) "events*"
    3) "events101"
    4) "Hello there!"
```

2. Опубликуйте сообщение на канале events101 с текстом “Hello there”.

```
127.0.0.1:6379> publish events101 'Hello there!'
    (integer) 1
```

Задание 4

Цель практической работы:

Научиться работать с хранимыми функциями в Redis.

Что нужно сделать:

Сохраните в Redis функцию, которая принимает ключ и значение и сохраняет под указанным ключом квадратный корень от значения.

Что оценивается:

Верный запрос.

---

1. Выполнение Lua-скрипта с помощью EVAL

   Создание скрипта

   127.0.0.1:6379> eval "local result = ARGV[1] .. '^(1/2) = ' .. math.sqrt(ARGV[1]);
   redis.call('set', KEYS[1], result); return result" 1 sqr 2

   "2^(1/2) = 1.4142135623731"

   Проверка значения ключа sqr с помощью get

   127.0.0.1:6379> get sqr

   "2^(1/2) = 1.4142135623731"

2. Загрузка скрипта с помощью SCRIPT LOAD

   Загрузка скрипта

   127.0.0.1:6379> SCRIPT LOAD "local result = ARGV[1] .. '^(1/2) = ' .. math.sqrt(ARGV[1]);
   redis.call('set', KEYS[1], result); return result"

   "3d3442065187702ec9ec0c90b385452a0cdd925e"

   Проверка (загружены ли скрипты) с помощью команды SCRIPT EXISTS

   127.0.0.1:6379> SCRIPT EXISTS 3d3442065187702ec9ec0c90b385452a0cdd925e

   1) (integer) 1

3. Запуск загруженного скрипта с помощью EVALSHA

   127.0.0.1:6379> EVALSHA 3d3442065187702ec9ec0c90b385452a0cdd925e 1 sqr 25

   "25^(1/2) = 5"

Ссылка на репозиторий:
https://github.com/olgashenkel/GeekBrains-specialization-ELECTIVES/tree/main/11.%20Non-relational%20databases%20and%20MongoDB