B4M36DS2, BE4M36DS2: **Database Systems 2**

https://cw.fel.cvut.cz/b221/courses/b4m36ds2/

Practical Class 5

Redis: data types

Yuliia Prokop
prokoyul@fel.cvut.cz

23. 10. 2022

Authors: Martin Svoboda (martin.svoboda@matfyz.cuni.cz),
Yuliia Prokop

Czech Technical University in Prague, Faculty of Electrical Engineering

Redis

Redis

- **In-memory data structure store**
  - Open source, master-slave replication architecture, sharding, high availability, various persistence levels, …
- http://redis.io/
- Developed by **Redis Labs**
- Implemented in **C**
- First release in 2009

Interface

`redis-cli` command line client

- Two modes are available…
- **Basic**
  - Commands are passed as standard command line arguments
  - E.g. `redis-cli PING`
    `redis-cli -n 16 DBSIZE`
  - Batch processing is possible as well
  - E.g. `cat script.txt | redis-cli`
- **Interactive**
  - Users type database commands at the prompt
  - `redis-cli`

RESP (*REdis Serialization Protocol*)

First Steps

Connect to our NoSQL server

- SSH / SFTP and PuTTY / WinSCP
- nosql.felk.cvut.cz

Check Redis status

- `redis-cli PING`

Open Redis client (interactive mode)

- `redis-cli`

Select your database

- **SELECT** number
- Your database number

First Steps

Basic Commands

- **HELP** command
  - Provides basic information about Redis commands
- **CLEAR**
  - Clears the terminal screen
- **FLUSHDB**
  - Deletes all the keys in the currently selected database
- **BGSAVE**
  - Saves the current dataset (asynchronously, on background)
  - I.e. stores the database snapshot to the hard drive
- **QUIT**
  - Closes the connection

Example 1 - Strings

```
127.0.0.1:6379 > set mykey somevalue
OK
127.0.0.1:6379 > get mykey
"somevalue"
127.0.0.1:6379> set counter 100
OK
127.0.0.1:6379> get counter
"100"
127.0.0.1:6379> mset a 10 b 20 c 30
OK
127.0.0.1:6379> mget a b c
1) "10"
2) "20"
3) "30"
127.0.0.1:6379> incr counter
(integer) 101
127.0.0.1:6379> incr counter
(integer) 102
127.0.0.1:6379> incrby counter 50
(integer) 152
```

```
127.0.0.1:6379> set mykey somevalue
OK
127.0.0.1:6379> get mykey
"somevalue"
127.0.0.1:6379> strlen mykey
(integer) 9
127.0.0.1:6379> append mykey 666
(integer) 12
127.0.0.1:6379> get mykey
"somevalue666"
127.0.0.1:6379> getrange mykey 2 8
"mevalue"
127.0.0.1:6379> setrange mykey 5 xxxx
(integer) 12
127.0.0.1:6379> get mykey
"somevxxxx666"
```

Exercise 2 - Objects

```
127.0.0.1:6379> exists mykey
(integer) 1
127.0.0.1:6379> del mykey
(integer) 1
127.0.0.1:6379> exists mykey
(integer) 0
```

```
127.0.0.1:6379> set a hello
OK
127.0.0.1:6379> get a
"hello"
127.0.0.1:6379> rename a ahoj
OK
127.0.0.1:6379> get a
(nil)
127.0.0.1:6379> exists a
(integer) 0
127.0.0.1:6379> get ahoj
"hello"
```

```
127.0.0.1:6379> type ahoj
string
127.0.0.1:6379> set x 120
OK
127.0.0.1:6379> type x
string
```

Exercise 3 - Volatile Objects

```
127.0.0.1:6379> set key some-value
OK
127.0.0.1:6379> expire key 20
(integer) 1
127.0.0.1:6379> get key
"some-value"
127.0.0.1:6379> get key
"some-value"
127.0.0.1:6379> get key
(nil)

127.0.0.1:6379> set key 100 ex 10
OK
127.0.0.1:6379> ttl key
(integer) 2
```

Exercise 4 - Lists

```
127.0.0.1:6379> rpush mylist A
(integer) 1
127.0.0.1:6379> rpush mylist B
(integer) 2
127.0.0.1:6379> lpush mylist first
(integer) 3
127.0.0.1:6379> lrange mylist 0 -1
1) "first"
2) "A"
3) "B"
127.0.0.1:6379> rpush mylist 1 2 3 4 5 "foo bar"
(integer) 9
127.0.0.1:6379> rpop mylist
"foo bar"
127.0.0.1:6379> rpop mylist
"5"
127.0.0.1:6379> rpop mylist
"4"
```

```
127.0.0.1:6379> del mylist
(integer) 1
127.0.0.1:6379> rpush mylist 1 2 3 4 5
(integer) 5
127.0.0.1:6379> ltrim mylist 0 2
OK
127.0.0.1:6379> lrange mylist 0 -1
1) "1"
2) "2"
3) "3"
```

Exercise 4 – Lists (cont.)

Remove 2 elements with value "2"

127.0.0.1:6379> **rpush** mylist 2
(integer) 4
127.0.0.1:6379> **rpush** mylist 2
(integer) 5
127.0.0.1:6379> **lrange** mylist 0 999
1) **"1"**
2) "2"
3) **"3"**
4) "2"
5) **"2”**
127.0.0.1:6379> **lrem** mylist 2 "2"
(integer) 2
127.0.0.1:6379> **lrange** mylist 0 999
1) "1"
2) "3"
3) "2"

Exercise 5 – Sets

Create a set with elements 1, 2, 3

```
127.0.0.1:6379> sadd myset 1 2 3
(integer) 3
127.0.0.1:6379> smembers myset
1) "1"
2) "2"
3) "3"
```

Test existence of 3 and 30 in the set

```
127.0.0.1:6379> sismember myset 3
(integer) 1
127.0.0.1:6379> sismember myset 30
(integer) 0
```

Exercise 5 – Sets (cont.)

Tags 1, 2, 5 and 77 are associated with the news item 1000:

```
127.0.0.1:6379> sadd news:1000:tags 1 2 5 77
(integer) 4
```

Reverse - the list of all the news tagged with a given tag 1000:

```
127.0.0.1:6379> sadd tag:1:news 1000
(integer) 1
127.0.0.1:6379> sadd tag:2:news 1000
(integer) 1
127.0.0.1:6379> sadd tag:5:news 1000
(integer) 1
127.0.0.1:6379> sadd tag:77:news 1000
(integer) 1
127.0.0.1:6379> smembers news:1000:tags
1) "1"
2) "2"
3) "5"
4) "77"
```

Exercise 5 – Sets (cont.)

```
127.0.0.1:6379> sadd set1 a b c d
(integer) 4
127.0.0.1:6379> sadd set2 c
(integer) 1
127.0.0.1:6379> sadd set3 a c e
(integer) 3
127.0.0.1:6379> sunion set1 set2 set3
1) "a"
2) "e"
3) "c"
4) "d"
5) "b"
127.0.0.1:6379> sinter set1 set2 set3
1) "c"
127.0.0.1:6379> sdiff set1 set2 set3
1) "b"
2) "d"
```

Exercise 6 – Hashes

```
127.0.0.1:6379> hset user:1000 username antirez birthyear 1977
verified 1
(integer) 3
127.0.0.1:6379> hget user:1000 username
"antirez"
127.0.0.1:6379> hget user:1000 birthyear
"1977"
127.0.0.1:6379> hgetall  user:1000
1) "username"
2) "antirez"
3) "birthyear"
4) "1977"
5) "verified"
6) "1"
```

Exercise 6 – Hashes (cont.)

127.0.0.1:6379> **hmget** user:1000 username birthyear no-such-field
1) "antirez"
2) "1977"
3) (nil)
127.0.0.1:6379> **hincrby** user:1000 birthyear 10
(integer) 1987
127.0.0.1:6379> **hincrby** user:1000 birthyear 10
(integer) 1997

Exercise 7 – Sorted Sets

```
127.0.0.1:6379> zadd hackers 1940 "Alan Kay"
(integer) 1
127.0.0.1:6379> zadd hackers 1957 "Sophie Wilson"
(integer) 1
127.0.0.1:6379> zadd hackers 1953 "Richard Stallman"
(integer) 1
127.0.0.1:6379> zadd hackers 1949 "Anita Borg"
(integer) 1
127.0.0.1:6379> zadd hackers 1965 "Yukihiro Matsumoto"
(integer) 1
127.0.0.1:6379> zadd hackers 1914 "Hedy Lamarr"
(integer) 1
127.0.0.1:6379> zadd hackers 1916 "Claude Shannon"
(integer) 1
127.0.0.1:6379> zadd hackers 1969 "Linus Torvalds"
(integer) 1
127.0.0.1:6379> zadd hackers 1912 "Alan Turing"
(integer) 1
```

Sorted Sets (cont.)

127.0.0.1:6379> **zrange hackers 0 -1**
1) "Alan Turing"
2) "Hedy Lamarr"
3) "Claude Shannon"
4) "Alan Kay"
5) "Anita Borg"
6) "Richard Stallman"
7) "Sophie Wilson"
8) "Yukihiro Matsumoto"
9) "Linus Torvalds"

127.0.0.1:6379> **zrange hackers 0 -1 withscores**
1) "Alan Turing"
2) "1912"
3) "Hedy Lamarr"
4) "1914"
5) "Claude Shannon"
6) "1916"
7) "Alan Kay"
8) "1940"
9) "Anita Borg"
10) "1949"
11) "Richard Stallman"
12) "1953"
13) "Sophie Wilson"
14) "1957"
15) "Yukihiro Matsumoto"
16) "1965"
17) "Linus Torvalds"
18) "1969"

Exercise 7 – Sorted Sets (cont.)

Born before 1950:

127.0.0.1:6379> **zrangebyscore hackers -inf 1950**
1) "Alan Turing"
2) "Hedy Lamarr"
3) "Claude Shannon"
4) "Alan Kay"
5) "Anita Borg"

Remove hackers born between 1940 and 1960:

127.0.0.1:6379> **zremrangebyscore hackers 1940 1960**
(integer) 4

Exercise 7 – Sorted Sets (cont.)

```
127.0.0.1:6379> zrange hackers 0 -1 withscores
 1) "Alan Turing"
 2) "1912"
 3) "Hedy Lamarr"
 4) "1914"
 5) "Claude Shannon"
 6) "1916"
 7) "Yukihiro Matsumoto"
 8) "1965"
 9) "Linus Torvalds"
10) "1969"
```

Rank of Hedy Lamarr:

```
127.0.0.1:6379> zrank hackers "Hedy Lamarr"
(integer) 1
```

Exercise 8

The information about students:

Alice Brown with email alice.brown@email.com in 2015 travelled to Italy and visited Roma, Milan, Venice. In 2016 she travelled to Polland and visited Warsaw and Krakov and also Czech Republic and visited Prague and Brno.

Alex Fisher in 2016 travelled to Germany and visited Berlin, Erfurt and Koln.

Betty Fox with email bet.fox@email.com travelled to Istanbul in 2015 and to Roma in 2017.

Insert this data with appropriate types into your Redis DB

Exercise 9

A. Retrieve the email address of Alice Brown.

B. Check if Betty Fox travelled in 2016

C. Retrieve all the places that Alex Fisher visited in 2016

D. Retrieve all journeys made by Betty Fox

E. Where did the students travel in 2015?

References

Commands

- http://redis.io/commands

Documentation

- http://redis.io/documentation

Data types

- http://redis.io/topics/data-types