

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 -p port -a password`**
 - 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 -p port -a password`

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

set mykey somevalue

OK

get mykey

"somevalue"

set counter 100

OK

get counter

"100"

mset a 10 b 20 c 30

OK

mget a b c

1) "10"

2) "20"

3) "30"

incr counter

(integer) 101

incr counter

(integer) 102

incrby counter 50

(integer) 152

set mykey somevalue

OK

get mykey

"somevalue"

strlen mykey

(integer) 9

append mykey 666

(integer) 12

get mykey

"somevalue666"

getrange mykey 2 8

"mevalue"

setrange mykey 5 xxxx

(integer) 12

get mykey

"somevxxxx666"

Exercise 3 – Operations with keys

SET product:1001:name "Ergonomic Office Chair"

OK

SET product:1002:name "Executive Desk"

OK

SET product:1003:name "Filing Cabinet"

OK

SET product:1004:name "Conference Table"

OK

SET product:1005:name "Bookshelf"

OK

GET product:1001:name

"Ergonomic Office Chair"

GET product:1002:name

"Executive Desk"

GET product:1003:name

"Filing Cabinet"

GET product:1004:name

"Conference Table"

GET product:1005:name

"Bookshelf"

MSET product:1001:price 199.99 product:1001:stock 50

MGET product:1001:price product:1001:stock

Exercise 3 – Operations with keys (cont.)

The **KEYS** command in Redis is used to search for keys matching a specified pattern.

KEYS product*:name

- 1) "product:1001:name"
- 2) "product:1002:name"
- 3) "product:1005:name"
- 4) "product:1004:name"
- 5) "product:1003:name"

KEYS product:100[1-3]:name

- 1) "product:1001:name"
- 2) "product:1002:name"
- 3) "product:1003:name"

Exercise 3 – Operations with keys (cont.)

SCAN is a Redis command for iteratively scanning keys in the database.

SCAN 0 MATCH product:*:name COUNT 10

- 1) "0" # The scan is complete
- 2) 1) "product:1001:name"
- 2) "product:1003:name"
- 3) "product:1002:name"
- 4) "product:1004:name"
- 5) "product:1005:name"

SCAN 0 MATCH product:*:name COUNT 3

- 1) "10" # The scan is not complete
- 2) 1) "product:1001:name"

Exercise 3 – Operations with keys (cont.)

SCAN 10 MATCH product*:name COUNT 3

- 1) "13" # The scan is not complete
- 2) 1) "product:1003:name"
2) "product:1002:name"

SCAN 13 MATCH product*:name COUNT 3

- 1) "15" # The scan is not complete
- 2) 1) "product:1003:name"
2) "product:1002:name"

SCAN 15 MATCH product*:name COUNT 3

- 1) "0" # The scan is complete
- 2) (empty array)

Exercise 3 - Volatile Objects

```
set key some-value
```

```
OK
```

```
expire key 20
```

```
(integer) 1
```

```
get key
```

```
"some-value"
```

```
get key
```

```
"some-value"
```

```
get key
```

```
(nil)
```

```
set key 100 ex 10
```

```
OK
```

```
ttd key
```

```
(integer) 2
```

Exercise 4 - Lists

rpush mylist A

(integer) 1

rpush mylist B

(integer) 2

lpush mylist first

(integer) 3

lrange mylist 0 -1

1) "first"

2) "A"

3) "B"

rpush mylist 1 2 3 4 5 "foo bar"

(integer) 9

rpop mylist

"foo bar"

rpop mylist

"5"

rpop mylist

"4"

del mylist

(integer) 1

rpush mylist 1 2 3 4 5

(integer) 5

ltrim mylist 0 2

OK

lrange mylist 0 -1

1) "1"

2) "2"

3) "3"

Exercise 4 – Lists (cont.)

Remove 2 elements with value "2"

```
rpush mylist 2
```

```
(integer) 4
```

```
rpush mylist 2
```

```
(integer) 5
```

```
lrange mylist 0 999
```

```
1) "1"
```

```
2) "2"
```

```
3) "3"
```

```
4) "2"
```

```
5) "2"
```

```
lrem mylist 2 "2"
```

```
(integer) 2
```

```
lrange mylist 0 999
```

```
1) "1"
```

```
2) "3"
```

```
3) "2"
```

Exercise 5 – Sets

Create a set with elements 1, 2, 3

```
sadd myset 1 2 3  
(integer) 3  
smembers myset  
1) "1"  
2) "2"  
3) "3"
```

Test existence of 3 and 30 in the set

```
sismember myset 3  
(integer) 1  
sismember myset 30  
(integer) 0
```

Exercise 5 – Sets (cont.)

News articles and tags (verbal description)

News articles:

- Article 1000: "New Technologies in Medicine" (tags: 1, 2, 5, 77)
- Article 1001: "Sports Events of the Week" (tags: 3)
- Article 1002: "Economic Forecast for 2024" (tags: 4)
- Article 1003: "AI in Healthcare" (tags: 1, 2, 5, 77, 8)
- Article 1004: "The Impact of Technology on the Economy" (tags: 1, 4, 8)
- Article 1005: "Sports Medicine Advancements" (tags: 2, 3, 77)

Tags:

- 1: Technology
- 2: Health
- 3: Sports
- 4: Economy
- 5: Science
- 8: AI
- 77: Medicine

Exercise 5 – Sets (cont.)

Storing tags for each article:

news:1000:tags -> {1, 2, 5, 77}
news:1001:tags -> {3}
news:1002:tags -> {4}
news:1003:tags -> {1, 2, 5, 77, 8}
news:1004:tags -> {1, 4, 8}
news:1005:tags -> {2, 3, 77}

News articles and tags (in Redis)

SADD news:1000:tags 1 2 5 77
SADD news:1001:tags 3
SADD news:1002:tags 4
SADD news:1003:tags 1 2 5 77 8
SADD news:1004:tags 1 4 8
SADD news:1005:tags 2 3 77

Reverse index (articles for each tag):

tag:1:news -> {1000, 1003, 1004}
tag:2:news -> {1000, 1003, 1005}
tag:3:news -> {1001, 1005}
tag:4:news -> {1002, 1004}
tag:5:news -> {1000, 1003}
tag:8:news -> {1003, 1004}
tag:77:news -> {1000, 1003, 1005}

SADD tag:1:news 1000 1003 1004
SADD tag:2:news 1000 1003 1005
SADD tag:3:news 1001 1005
SADD tag:4:news 1002 1004
SADD tag:5:news 1000 1003
SADD tag:8:news 1003 1004
SADD tag:77:news 1000 1003 1005

Exercise 5 – Sets (cont.)

Storing article content:

news:1000:content -> "New Technologies in Medicine"

news:1001:content -> "Sports Events of the Week"

news:1002:content -> "Economic Forecast for 2024"

news:1003:content -> "AI in Healthcare"

news:1004:content -> "The Impact of Technology on the Economy"

news:1005:content -> "Sports Medicine Advancements"

Redis

SET news:1000:content "New Technologies in Medicine"

SET news:1001:content "Sports Events of the Week"

SET news:1002:content "Economic Forecast for 2024"

SET news:1003:content "AI in Healthcare"

SET news:1004:content "The Impact of Technology on the Economy"

SET news:1005:content "Sports Medicine Advancements"

Exercise 5 – Sets (cont.)

Storing tag names:

tag:1:name -> "Technology"

tag:2:name -> "Health"

tag:3:name -> "Sports"

tag:4:name -> "Economy"

tag:5:name -> "Science"

tag:8:name -> "AI"

tag:77:name -> "Medicine"

Redis

SET tag:1:name "Technology"

SET tag:2:name "Health"

SET tag:3:name "Sports"

SET tag:4:name "Economy"

SET tag:5:name "Science"

SET tag:8:name "AI"

SET tag:77:name "Medicine"

Exercise 5 – Sets (cont.)

Set of all articles:

articles:all -> {1000, 1001, 1002, 1003, 1004, 1005}

Redis

SADD articles:all 1000 1001 1002 1003 1004 1005

Article Tags (Sets):

Keys like news:1000:tags, news:1001:tags, etc., are Sets containing tag IDs for each article. Each article has a set of tags associated with it.

Tag-to-Articles Index (Sets):

Keys like tag:1:news, tag:2:news, etc., are Sets containing article IDs for each tag. This is a reverse index allowing quick lookup of all articles with a specific tag.

Article Content (Strings):

Keys like news:1000:content, news:1001:content, etc., are Strings storing the content or title of each article.

Tag Names (Strings):

Keys like tag:1:name, tag:2:name, etc., are Strings storing the human-readable names of tags.

All Articles (Set):

The key articles:all is a Set containing IDs of all articles in the system.

Exercise 5 – Sets (cont.)

This structure allows for efficient operations such as:

- Finding all tags for an article
- Finding all articles with a specific tag
- Retrieving article content
- Getting tag names
- Listing all articles

The use of Sets for tags and reverse indexes enables fast intersection, union, and difference operations, which are useful for complex queries involving multiple tags.

Exercise 5 – Sets (cont.)

Get all tags for an article:

SMEMBERS news:1000:tags

Find all articles with a specific tag:

SMEMBERS tag:1:news

Find articles with both tag 1 (Technology) AND tag 2 (Health):

SINTER tag:1:news tag:2:news

Exercise 5 – Sets (cont.)

1. Add the tag "Research" (tag ID: 6) to the article with ID 1000
2. Remove the tag "Health" (tag ID: 2) from the article with ID 1000
3. Check if the article with ID 1000 has the "Science" tag (tag ID: 5)
4. Count how many tags are associated with the article ID 1000
5. Find articles that have both "Technology" (tag ID: 1) and "Health" (tag ID: 2) tags
6. Add a new article (ID: 1003) with tags "Technology" (1) and "Economy" (4)
7. Get a list of all article IDs
8. Rename the tag "Economy" (tag ID: 4) to "Finance"
9. Find all news items that have either tag 1 or tag 77
10. Find news items with tag 1 but without tag 4
11. Check if news item 1003 has tag 5
12. Add tag 6 to news item 1002
13. Remove tag 77 from news item 1000
14. Count how many news items have tag 2
15. Get a random news item with tag 3
16. Move news item 1001 from tag 3 to tag 4

Exercise 6 – Hashes

hset user:1000 username antirez birthyear 1977 verified 1
(integer) 3

hget user:1000 username
"antirez"

hget user:1000 birthyear
"1977"

hgetall user:1000

- 1) "username"
- 2) "antirez"
- 3) "birthyear"
- 4) "1977"
- 5) "verified"
- 6) "1"

Exercise 6 – Hashes (cont.)

hmget user:1000 username birthyear no-such-field

1) "antirez"

2) "1977"

3) (nil)

hincrby user:1000 birthyear 10

(integer) 1987

hincrby user:1000 birthyear 10

(integer) 1997

Exercise 7 – Sorted Sets

zadd hackers 1940 "Alan Kay"
zadd hackers 1957 "Sophie Wilson"
zadd hackers 1953 "Richard Stallman"
zadd hackers 1949 "Anita Borg"
zadd hackers 1965 "Yukihiro Matsumoto"
zadd hackers 1914 "Hedy Lamarr"
zadd hackers 1916 "Claude Shannon"
zadd hackers 1969 "Linus Torvalds"
zadd hackers 1912 "Alan Turing"

Sorted Sets (cont.)

Sorted Set represents a leaderboard for an online game.

```
ZADD game:leaderboard 5000 "Player1" 3500 "Player2" 4200  
"Player3" 2800 "Player4" 6100 "Player5" 3900 "Player6" 5500  
"Player7" 4800 "Player8" 3200 "Player9" 5900 "Player10"
```

Sorted Sets (cont.)

Exercises:

1. Get the top 5 players
2. Get the 3 players with the lowest scores
3. Get the rank of "Player3" (starting from 0, ascending score)
4. Get the rank of "Player3" (starting from 0, descending score)
5. Get the score of "Player5"
6. Increase "Player2" score by 750 points
7. Get the number of players in the leaderboard
8. Count the number of players with scores between 4000 and 5000
9. Get all players with scores between 3000 and 4500
10. Remove players with scores below 3000
11. Get players with ranks 3 to 7 (by descending score)
12. Intersection with another sorted set (assuming we have another set `game:daily_leaders`)
13. Get scores of multiple players at once

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 Poland 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

- <https://redis.io/docs/latest/develop/data-types/>