B4M36DS2: **Database Systems 2**

http://www.ksi.mff.cuni.cz/~svoboda/courses/2016-1-B4M36DS2/

Practical Class 3

Redis Data Structure Store

Martin Svoboda

svoboda@ksi.mff.cuni.cz

31. 10. and 1. 11. 2016

Charles University in Prague, Faculty of Mathematics and Physics

Czech Technical University in Prague, Faculty of Electrical Engineering

Redis Overview

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

Redis Overview

Functionality

- Standard **key-value store**
- Support for **structured values** (e.g. lists, sets, …)
- **Time-to-live**
- Transactions

Real-world users

- Twitter, GitHub, Pinterest, StackOverflow, Flicker, …

Data Model

Data model

$\boxed{\text{Instance} \rightarrow \textbf{databases} \rightarrow \textbf{objects}}$

- **Database** = collection of objects
  - Databases do not have names, but integer identifiers
- **Object** = **key-value pair**
  - Key is a **string** (i.e. any binary data)
  - Values can be…
    - Atomic: **string**
    - Structured: **list**, **set**, **ordered set**, **hash**

Data Types

Available **data types**

- **String**
  - The only **atomic data type**
  - May contain any binary data
    (e.g. string, integer counter, PNG image, …)
  - Maximal allowed size is 512 MB
- **List**
  - **Ordered collection of strings**
  - Elements should preferably be read / written at the head / tail

Data Types

Available **data types**

- **Set**
  - **Unordered collection of strings**
  - Duplicate values are not allowed
- **Sorted set**
  - **Ordered collection of strings**
  - The order is given by a score (floating number value) associated with each element (from the smallest to the greatest score)
- **Hash**
  - **Associative map between string fields and string values**
  - Field names have to be mutually distinct

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 (*__RE__dis __S__erialization __P__rotocol*)

Tutorial: Redis

First Steps

Remotely connect to our NoSQL server

- SSH and SFTP access
- PuTTY and WinSCP on Windows
- **147.32.83.196:22**

Check Redis status

- `redis-cli PING`

Open Redis client (interactive mode of command line interface)

- `redis-cli`

Select your database

- `SELECT number`
- Your database number: sent by e-mail

First Steps

Basic Commands

- `HELP command`
  - Provides basic information about Redis commands
- `CLEAR`
  - Clears the terminal screen
- `FLUSHDB`
  - Deletes all the keys of 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

Strings

Basic commands

- `SET key value` – inserts / replaces a given string
- `GET key` – returns a given string

String operations

- `STRLEN key` – returns a string length
- `APPEND key value` – appends a value at the end of a string
- `GETRANGE key start end` – returns a substring
  - Both the boundaries are considered to be inclusive
  - Positions start at 0
  - Negative offsets for positions starting at the end
- `SETRANGE key offset value` – replaces a substring
  - Binary 0 are padded when the original string is not long enough

Strings

Counter operations

- `INCR key`
  `DECR key`
  - Increments / decrements a value by 1
- `INCRBY key increment`
  `DECRBY key increment`
  - Increments / decrements a value by a given amount

Keys

Object querying

- `EXISTS key` – determines whether a key exists
- `KEYS pattern` – finds all the keys matching a pattern (*, ?, …)
    - E.g. `KEYS *`

Modification of objects

- `DEL key …` – removes a given object / objects
- `RENAME key newkey` – changes the key of a given object

Type information

- `TYPE key` – determines the type of a given object
    - Types: `string`, `list`, `set`, `zset` and `hash`

Volatile Keys

Keys with limited time to live

- When a specified timeout elapses, a given object is removed
- Works with any data type

Commands

- `EXPIRE key seconds`
  - Sets a timeout for a given object, i.e. makes the object volatile
  - Can be called repeatedly to change the timeout
- `TTL key`
  - Returns the remaining time to live for a key that has a timeout
- `PERSIST key`
  - Removes the existing timeout, i.e. makes the object persistent

Lists

Insertion of new elements

- `LPUSH key value`
  `RPUSH key value`
  - Adds a new element to the head / tail
- `LINSERT key BEFORE|AFTER pivot value`
  - Inserts an element before / after another one

Retrieval of elements

- `LPOP key`
  `RPOP key`
  - Removes and returns the first / last element

Lists

Retrieval of elements

- `LINDEX key index` – gets an element by its index
  - The first item is at position 0
  - Negative positions are allowed as well
- `LRANGE key start stop` – gets a range of elements

Removal of elements

- `LREM key count value`
  - Removes a given number of matching elements from a list
    - Positive / negative = moving from head to tail / tail to head
    - 0 = all the items are removed

Other operations

- `LLEN key` – gets the length of a list

Sets

Basic operations

- `SADD key value …`
  - Adds an element / elements into a set
- `SREM key value …`
  - Removes an element / elements from a set

Data querying

- `SISMEMBER key value`
  - Determines whether a set contains a given element
- `SMEMBERS key` – gets all the elements of a set

Other operations

- `SCARD key` – gets the number of elements in a set

Sets

Set operations

- `SUNION key …`
  `SINTER key …`
  `SDIFF key …`
  - Calculates and returns a set union / intersection / difference of two or more sets

Hashes

Basic operations

- `HSET key field value` – sets the value of a hash field
- `HGET key field` – gets the value of a hash field

Batch alternatives

- `HMSET key field value … …`
  - Sets values of multiple fields of a given hash
- `HMGET key field …`
  - Gets values of multiple fields of a given hash

Hashes

Field retrieval operations

- `HEXISTS key field` – determines whether a given field exists
- `HGETALL key` – gets all the fields and values
    - Individual fields and values are interleaved
- `HKEYS key` – gets all the fields in a given hash
- `HVALS key` – gets all the values in a given hash

Other operations

- `HDEL key field …`
    - Removes a given field / fields from a hash
- `HLEN key` – returns the number of fields in a given hash

Sorted Sets

Basic operations

- `ZADD key score value … …`
  - Inserts one element / multiple elements into a sorted set
- `ZREM key value …`
  - Removes one element / multiple elements from a sorted set

Working with score

- `ZSCORE key value`
  - Gets the score associated with a given element
- `ZINCRBY key increment value`
  - Increments the score of a given element

Sorted Sets

Retrieval of elements

- `ZRANGE key start stop`
  - Returns all the elements within a given range based on positions
- `ZRANGEBYSCORE key min max`
  - Returns all the elements within a given range based on scores

Other operations

- `ZCARD key`
  - Gets the overall number of all elements
- `ZCOUNT key min max`
  - Counts all the elements within a given range based on score

References

Commands

- http://redis.io/commands

Documentation

- http://redis.io/documentation

Data types

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