

REDIS

Note:

It still looks like there are a lot of commands to remember, but many of these commands perform similar functions.

- **String**

- Setters

- SET

- Used to set a string value to a key

- ex: SET message 'HI'

- GET

- GET is an optional to the SET command => Get then update

- ex: SET message 'Hello' GET => returns the value of message then update it to 'Hello'

- if no previous value for the key message then it will return 'nil'

- XX

- XX is an optional to the SET command => update if exist

- ex: SET message 'HEY' XX => if message is already exist, then it will update it

- else it will return 'nil'

- NX

- NX is an optional to the SET command => set if not exist

- ex: SET message 'HEY' NX => if message is not already exist, then it will set it

- else it will return 'nil'

- EX

EX is an optional to the SET command => set for only a couple of seconds

ex: SET message 'HEY' EX 5 => after 5 seconds delete this message key

- SETRANGE

Used for updating a portion of a string value.

ex: SETRANGE message 3 'ASA'

if message value was Hello it will be => 'HeASA'

- There are more SET commands but not commonly used.

- Getters

- GET

ex: GET name => Ahmed

- GETRANGE

Used for getting a portion of a string value.

ex: GET data 0 2

if data was 'ASA22' => it will return 'ASA'

- **Number**

- INCR, DECR

Used for incrementing or decrementing a value.

ex: INCR age

if age = 20 it will be 21

- INCRBY, DECRBY

Used for incrementing or decrementing a value.

ex: INCRBY age 10

if age = 20 it will be 31

- INCRBYFLOAT

Used for incrementing and decrementing a value by a float number.

by using minus before the number it means decrease.

ex: INCRBYFLOAT size -10.5

if size = 25 it will be 14.5

- So you may ask, 'Why do these number commands exist when I can retrieve and update this value on the server side and then save it back to the database?'

The answer is, if we were to perform the update process on the server side, we would need to carry out two database operations—one for retrieving the data and another for setting it back.

This would compromise the efficiency of using an in-memory database, which is all about speed.

Also, updating values on the server side may lead to dealing with inconsistent values. This can occur when two requests for obtaining the same value happen simultaneously.

In this case, both requests will read the same value. For instance, if both requests increase it by 1 and write it back, the data will have been increased by one when it should have been increased by two.

- **HASH**

- HSET

- Used for adding multiple Key value pairs to a single key or decrementing a value.

- ex: HSET person name 'Ahmed' age 22

- HGET

- Used for retrieving a Hash Key value by the field name.
ex: HGET person age
- HGETALL
 - Used for retrieving ALL the Hash Key value pairs.
ex: HGETALL person
- HEXISTS
 - Used for checking for the existence of a field in the hash.
ex: HEXISTS person age
return 0 or 1
- HDEL
 - Used for deleting a field from the hash.
ex: HDEL person age
- DEL
 - Delete the entire hash.
ex: DEL person
- HINCRBY
 - Adding an integer number value to a hash field.
ex: HINCRBY person age 10
- HINCRBYFLOAT
 - Adding float number value to a hash field.
ex: HINCRBYFLOAT person age 10.5
- HSTRLEN
 - Return the string length of a hash field.
ex: HSTRLEN person name
- HKEYS
 - Return the hash keys.
ex: HKEYS person
- HVALS
 - Return the hash values.
ex: HVALS person