Redis: In memory database

* Introduction

1. **What is Redis?**

Redis is key-value store database, using Redis we can build high performance and scalable applications.

1. **What are the best features of Redis?**
2. Redis holds its database entirely in the memory, using the disk only for persistence as a result **performance** will be high
3. Redis has a rich set of **data types** when compared to many key-value data stores.
4. Redis can **replicate** data to any number of slaves.
5. **What are data types supported by Redis?**

List

Set

Sorted Set

Hashes

1. **Are Redis operations atomic?**

Yes, two clients concurrently access, Redis server will receive the updated value.

1. **What are the use cases where you can use Redis?**
2. Caching
3. any short-lived data in your application, such as web application sessions, web page hit counts
4. messaging-queues (Redis natively supports Publish/Subscribe)
5. **How to setup redis?**

Can learn later

Note: In redis Keys are case sensitive kw\_87 and KW\_87 will be two different keys

1. **Redis Commands:**
   * **redis-cli**: this command is used to connect to redis from command line interface. Executing just **redis-cli** will connect to redis server running in local.

Using some options we can connect to the redis on remote server

* + - -h : need to mention ip(host) address
    - -p: Need to mention port

Ex: redis-cli -h 10.10.10.10 -p 10007

* + **Connection related commands**
    - **ping**: using this command we can check if server is running or not. Server gives response as pong ☺
  + **Key related command:** set of commands to manage key
    - **Del** KEY\_NAME : This command deletes the key, if it exists.
    - **Dump** KEY\_NAME: This command returns a serialized version of the value stored at the specified key.
    - **EXISTS** KEY\_NAME: This command checks whether the key exists or not.
    - **Expire KEY\_NAME TIME\_IN\_SECONDS:** Sets the expiry of the key after the specified time.
    - **Expireat KEY\_NAME TIME\_IN\_UNIX\_TIMESTAMP:** Sets the expiry of the key after the specified time. Here time is in Unix timestamp format.
    - **PEXPIRE KEY\_NAME TIME\_IN\_MILLISECONDS:** Set the expiry of key in milliseconds.
    - **PEXPIREAT KEY\_NAME TIME\_IN\_MILLISECONDS\_IN\_UNIX\_TIMESTAMP:** Sets the expiry of the key in Unix timestamp specified as milliseconds.
    - **KEYS PATTERN**: finds all keys matching the specified pattern.

Ex: KEYS \*, KEYS tutorial\*

* + - **MOVE KEY\_NAME DESTINATION\_DATABASE**: Moves a key to another database.
    - **PERSIST KEY\_NAME:** Removes the expiration from the key.
    - **PTTL KEY\_NAME:** Gets the remaining time in keys expiry in milliseconds. (Mostly TTL means total time left)
    - **TTL KEY\_NAME:** Gets the remaining time in keys expiry.
    - **RANDOMKEY:** Returns a random key from Redis.
    - **RENAME OLD\_KEY\_NAME NEW\_KEY\_NAME:** Changes the key name.
    - **RENAMENX OLD\_KEY\_NAME NEW\_KEY\_NAME:** Renames the key, if a new key doesn't exist.
  + **Value Datatype: String related commands :** Redis can store values of type String below are the commands to play with that
    - **SET KEY\_NAME VALUE:** This command sets the STRING value at the specified key.

**Options:** using options we can alter the behavior of the commands. Some of the options provided are

**EX seconds** – Using this option we can give expire time for this command in seconds after specified seconds code will come out of commands execution. It is nothing but time out

**PX milliseconds** – Same as Ex but time unit is milliseconds

**NX** − Value will be modified only if key does not exist.

**XX** − Value will be modified only if key already exists.

* + - **GET KEY\_NAME:** This command is used to get the value stored in the specified key

**Note:** thereare other commands I have read in website need to copy here when I have more time

<https://www.tutorialspoint.com/redis/redis_strings.htm>

* + **Value Datatype: List Related commands:**

Redis list is simply list of Strings

Redis list maintains insertion order

Elements can be added to head or tail of the list

Can add around 4 billion of elements to list.

Below are the commands to play with List

* + - **LPUSH KEY\_NAME VALUE1 … VALUEN :** This command inserts all the specified values at the head of the list which is stored at specified key.

If key does not exists inserts key and list with mentioned values.

If the specified key is not having list value then an error will be returned

Return value:

This command returns length of the list after push operation

* + - **LPOP KEY\_NAME:** This command removes the first element and returns the same from the list stored at the key.

Return Value

Value of the first element from the list

* + - **RPUSH KEY\_NAME VALUE1..VALUEN:** This command inserts all the specified values at the tail of the list which is stored at specified key.

If key does not exists inserts key and list with mentioned values.

If the specified key is not having list value then an error will be returned

Return value:

This command returns length of the list after push operation

* + - **RPOP KEY\_NAME:** This command removes last element and returns the same from the list stored at the key.

Return Value

Value of the last element from the list

* + - **LINDEX KEY\_NAME INDEX\_POSITION:** This command is used to get the element at the index in the list stored at the key.

The index is zero-based, so 0 means the first element, 1 the second element, and so on. Negative values can be used to access elements from the tail of the list. Here, -1 means the last element, -2 means last but one, and so forth.

* + **Value Datatype: Set Related commands:**

Redis Set will have only unique values.

Redis Set will not maintain insertion order.

Can add around 4 billion of elements to list.

Below are the commands to play with List.

* + - **SADD KEY\_NAME VALUE1..VALUEN:** add members to a set stored at specified key. If the member already exists, then it is ignored as set maintains only unique elements.

If the key does not exist, then a new set is created and the members are added into it.

If the value stored at the key is not set, then an error is returned.

Return Value when success  
only the number of elements that were added to the set in this command

* + - **SREM KEY MEMBER1..MEMBERN:** command is used to remove the specified member from the set stored at the key.

Return Value when success.  
Only the number of elements that were removed from the set in this command.

* + **Value Datatype: SortedSet Related commands:**

This is similar to set but sortedSet is sorted

* + - **ZADD KEY\_NAME SCORE1 VALUE1.. SCOREN VALUEN:**

adds all the specified members with the specified scores to the sorted set stored at the key.

* + **Value Datatype: Hashes related commnds:** In Redis hashes are like hashmaps in java. This means we can store map<String,String > as value to keys.Below are some of the useful commands.
    - **HMSET KEY\_NAME FIELD1 VALUE1 ...FIELDN VALUEN:**  It sets multiple key value pair to hashmap.If key already exists,if not it creates new one.
    - **HMGET KEY\_NAME FIELD1...FIELDN:** Gets the values of all the given hash fields
    - **HSET KEY\_NAME FIELD VALUE:** Sets the string value of a hash field
    - **HGET KEY\_NAME FIELD\_NAME:** Gets the value of a hash field stored at the specified key.

**Note:** thereare other commands I have read in

website need to copy here when I have more time

<https://www.tutorialspoint.com/redis/redis_hashes.ht>m

* + **Value Datatype: HyperLogLog related commnds**

Redis HyperLogLog is an algorithm that uses randomization in order to provide an approximation of the number of unique elements in a set using just a constant, and small amount of memory.

HyperLogLog provides a very good approximation of the cardinality of a set even using a very small amount of memory around 12 kbytes per key with a standard error of 0.81%. There is no limit to the number of items you can count, unless you approach 264 items.

* + - **PFADD KEY\_NAME ELEMENTS\_TO\_BE\_ADDED:** adds all the element arguments to the HyperLogLog data structure stored at the key name specified as first argument. PFADD mykey a b c d e f g h i j
  + **Redis publish and subscribe feature:**

Redis Publish/Subscribe implements the messaging system where the senders (publishers) sends the messages while the receivers (subscribers) receive them.

What is Channel in case of redis publish and subscribe?

The link by which the messages are transferred is called channel.

* + - **PSUBSCRIBE CHANNEL\_NAME\_OR\_PATTERN [PATTERN...]:** This command is used to subscribe to a particular channel or channels matching the given patterns.

Following listing shows some supported patterns in Redis.

h?llo subscribes to hello, hallo and hxllo

h\*llo subscribes to hllo and heeeello

h[ae]llo subscribes to hello and hallo, but not hillo

* + - **PUBLISH channel message:** Redis PUBLISH command posts a message to a channel.
    - **PUNSUBSCRIBE [pattern [pattern ...]] :** unsubscribes the client from the given patterns, or from all of them if none is given

PUNSUBSCRIBE mychannel

* + - **SUBSCRIBE channel [channel ...] :** subscribes the client to the specified channels.

Return value:

Array reply

* + **Redis Transaction feature:** Using this feature we can execute group of commands in a single step.

Properties of the Redis transactions.

1. All commands in a transaction are sequentially executed as a single operation. When transaction being executed request issued by another client will not be served.

Pending Things:

In Redis, by default 0th database is selected, so now we are moving the generated key in the second database.

What is 0th databse and 1st database in redis??

https://www.tutorialspoint.com/redis/keys\_move.htm