**REDIS LAB**

**Open and Create Database**

1.Open/login cloud:

Use local Redis DB

cd redis-5.0.5

cd src

/redis-server

2.Open a new command window and get into src

/redis-cli

3.Create / drop database:

you do not create a database in Redis with a command, the number of databases is defined in the configuration file (the default value is 16)

4.Use a desktop manager

**Basic Commands**

Five data types: string, list, set, hash, zset

1.String

**SET <key> <value>**: sets a key value pair

set UggClassic 160

**MSET <key> <value> <key> <value>…**: sets multiple key value pairs

mset UggClassicMini 140 UggCosmos 190

**GET <key>**: retrieves data based on key specified

get UggClassic

**DEL <key>**: to delete a key

del UggCosmos

**EXISTS <key>**: checks if a specified key exists in Redis database or not

exists UggCosmos

**INCRBY <key> <value>:** Increment by Specified amount

get UggClassic

incrby UggClassic 20

**DECRBY <key> <value>:** Decrement by Specified amount

get UggClassicMini

decrby UggClassicMini 10

2. List Data structure commands:

a. Lists are a group of strings, i.e. 1-D array of strings.

b. Sorted by insertion order.

c. Elements can be added to head or tail of the list.

d. Often used as producer/ consumer queries, or event queues, or notification systems.

**Push**: create lists

**LPUSH**<keyname><value>

LPUSH DB REDIS

LPUSH DB RavenDB

LPUSH DB MongoDB

**RPUSH** <keyname> <value>

RPUSH DB Cassandra

RPUSH DB Couchbase

**LLEN** <keyname>: get length of a list

LLEN DB

**LRANGE**<key name><start index><end index>: peep a list

lrange DB 0 -1

lrange DB 0 1

**LPOP**<keyname>: pop elements from head or tail

LPOP DB

LPOP DB

**RPOP**<keyname>

RPOP DB

RPOP DB

**LINSERT**<keyname> before/after <value><new\_value>: insert elements in the middle

linsert DB before REDIS OrientDB

linsert DB after REDIS RavenDB

3. Hashes Data structure commands

Hash is a map between string fields and string values. {key1:value1, key2:value2}. Useful when we want to create a container of unique fields and their values to represent objects.

**HMSET** <hash name> <key1> <value> <key2> <value>: create hashes

HMSET user1 name "Steve Rogers" email "captain@gmail.com" address "Brooklyn"

HMSET user2 name "Anthony Stark" email "iron@gmail.com" address "Malibu"

HMSET user3 name "Bruce Wayne" email "bat@gmail.com" address "Gotham"

**HMGET** <Hash Name> <key1> <key2>: read hashes

HMGET user1 name address

HMGET user1 name email

HGETALL <Hash Name>

HGETALL user1

**HMSET** <hash name> <key1> <value>: update hashes

HMSET user3 email "batman@gmail.com"

**HDEL** <Hash Name> <key>: delete hashes

HDEL user1 name

HDEL user1 email

HDEL user1 address

**HEXISTS** <hash name> <key>: to check if a key exists in the hash

HEXISTS user2 name

HEXISTS user2 gender

**HLEN** <hash name>: to get the size of the hash

HLEN user2

4．Sorted sets

Definition/Properties: Since sets are unsorted, they can pose problems for some projects. Sorted sets were created to solve that issue. Every element in sorted sets is associated with a ‘score’, which must be a float or number. Score is not unique, however, values are unique. Like sets, elements may only appear once. Users can access data very quickly through this way.

**ZADD** <key> <score> <member>: create sets

ZADD follow 100 user1

ZADD follow 120 user2

ZADD follow 70 user3

ZADD follow 69 user4

**ZREM** <key> <member>: remove member from sets

ZREM follow user2

**ZREMRANGEBYRANK** <key> <start> <stop>: remove a range of members by rank

ZREMRANGEBYRANK follow 0 1

ZREMRANGEBYSCORE key min max(inclusive)

ZREMRANGEBYSCORE follow 50 70

**ZRANGE** <key> <start> <stop> [WITHSCORES]: return a range of members

ZRANGE follow 0 -1

ZRANGE follow 0 -1 withscores

ZREVRANGE key start stop

ZREVRANGE follow 0 -1 withscores

**ZREVRANGEBYSCORE** <key> <min> <max> [WITHSCORES] [LIMIT]

ZREVRANGEBYSCORE follow 50 80 withscores

ZREVRANGEBYSCORE follow 50 99 limit 0 3 withscores

**ZCARD** <key>：Returns the sorted set cardinality (number of elements) of the sorted set stored at key

ZCARD follow

**ZCOUNT** <key> <min> <max>: count members in a range of scores

ZCOUNT follow 99 200

**ZINTERSTORE** <destination> <numkeys> <key> [key……]: get the intersection of several zsets

ZADD Apple 5 Sam 6 Kat 7 Tim

ZADD Dell 6 Sam 4 Kat 9 Sid

ZADD HP 7 Sam 2 Kat 3 Sid

ZINTERSTORE CP 3 Apple Dell HP

ZRANGE CP 0 -1 withscores

ZINTERSTORE CP 3 Apple Dell HP aggregate max

ZRANGE CP 0 -1 withscores

**ZUNIONSTORE** destination numkeys key [key……]: get the union of several zsets

ZUNIONSTORE CP 3 Apple Dell HP