**Redis Documentation**

www.redis.io

REDIS 🡪 REmote Dictionary Server

* dictionary/hashmap based
* in memory database, stored in RAM not HDD

When to use Redis 🡪 when the main database is slow, a query spends > 2 seconds

Diagram

Description automatically generated Diagram

Description automatically generated

1st request: User query to App 🡪 App query to database sql/third party App, needs 5 seconds 🡪 query data is caches in redis 🡪 query data sent to user

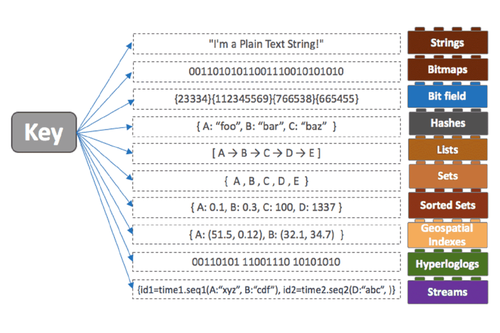
2nd, 3rd request: User query to App 🡪 App query to redis cache 🡪 data query sent to user

Redis architecture: Client – Server 🡪 request – response based

**Running redis**

Better to use redis docker image

**Redis data type**



**Redis operation**

* select 1 🡪 go to database 1, default database is 0. Database in Redis does not have name, instead they are named with number 0, 1, 2 …
* set key1 “eko” 🡪 store a key value pair
* get key1 🡪 call the value of key1
* exist key1 🡪 check if key1 exists
* keys \* 🡪 see all keys
* del key1 🡪 delete key1
* append key1 “kurniawan” 🡪 append additional value on key
* keys \*1 🡪 call value with conditional regex
* setrange key1 0 “eka” 🡪 set value on the mentioned range
* getrange key1 0 2 🡪 get value on the mentioned range
* mset key1 “eko” key2 “eka” key3 “eki” 🡪 set multiple key value pairs
* mget key1 key2 key3 🡪 get multiple key value pairs
* expire key1 60 🡪 set expiration key1 for 60 seconds
* setex key2 60 “eka” 🡪 set key and its expiration for 60 seconds
* ttl key1 🡪 time to live, check expiration of key1
* incr key1 🡪 increment by 1, only if value is integer. Use this function instead of increment manually, to diminish the possibility of value conflict on a very fast multiple process.
* decr key1 🡪 decrement by 1, only if value is integer
* incrby key1 10 🡪 increment by n, only if value is integer integer
* decrby key1 10 🡪 decrement by n, only if value is integer
* flushdb 🡪 remove all keys from the current database
* flushall 🡪 remove all keys from all database: database 0, database 1, etc

**Redis pipeline**

To make bulk data migration e.g. from mysql to redis, instead of one by one redis operation set, get, etc

redis-cli --pipe

make sets.txt file containing mulstiple set keyX valueX pairs

**Redis transaction**

To make multi operation, queued, and processed once you want. Like transcation in sql, all operation must be success, otherwise all operations will be cancelled

* multi
* write the operations
* exec or discard

**Redis monitor**

To monitor all requests entering redis server, to know if there is any wrong operation to redis server

* monitor 🡪 will make the terminal monitor every request to redis server. Open other terminal to make redis client request

**Redis server information**

To check redis server information and stats: config, memory used, etc

* info 🡪 get info and stats of redis server
* config get \* 🡪 get config information
* slowlog get 🡪 check what queries are among the top 10 slowests

**Redis client connection information**

To check redis client information

* client list 🡪 get the list od client connections
* client id 🡪 get client id for the current connection
* client kill ip:port 🡪 kill client’s connection

**Redis security**

As default, redis can only process request/connection from local host 127.0.0.1 or in same VM

**Redis authentication**

To make request from other VMs, use authentication

set password for clients in master’s redis.conf:

user default on +@connection

user user1 on +@all -DEBUG ~\* >mypassword

* auth user1 mypassword 🡪 user1 is connected, but still cant do redis operation 🡪 use authorization

**Redis authorization**

To give redis operation access rights to the authenticated users

set access rights in master’s redis.conf:

user default on +@connection

user user2 on +@all -@set ~\* >mypassword 🡪 user2 can do all operation but not set

user user3 on +@read ~key1 >mypassword 🡪 user3 can only read key1 key

user user1 on +@all -DEBUG ~\* >mypassword

**Redis persistent**

As default, redis store data in RAM. If redis shut down, the data will be removed. Redis can schedule to store data in persistent disk like HDD 🡪 snapshotting

save seconds changes 🡪 redis will store to HDD if there is change for n seconds and n changes byte

in master’s redis.conf:

save 300 10 🡪 if in 300 seconds there is 10 data changes

* save 🡪 synchronously save data to disk
* bgsave 🡪 asynchronously save data to disk

**Redis eviction**

When the memory is full, redis will try to delete the rarely used data with a policy

in master’s redis.conf:

maxmemory 10GB 🡪 max memory used by redis is 10GB

maxmemory-policy thepolicy 🡪 set eviction policy, read redis docs for elaboration