## *Redis*

- Redis is an open source, in-memory data structure store, used as a database.
- Stores data in form of Key Value pair
- Open Source, NoSql DB, Written in C
- Remote Dictionary Server
- strings, hashes, lists, sets, sorted sets

## *In-Memory Database*

- Keeps data in RAM
- Means that for every interaction with database, you will only access the Main memory
- Hence the operation will be faster as it directly aceess main memoery instead of secondary memory

## *How does Redis work*

All Redis data resides in-memory, in contrast to databases that store data on disk or SSDs. By eliminating the need to access disks, in-memory data stores such as Redis avoid seek time delays and can access data in microseconds. Redis features versatile data structures, high availability, geospatial, Lua scripting, transactions, on-disk persistence, and cluster support making it simpler to build real-time internet scale apps.

## *Redis Vs Memcached*

Both Redis and MemCached are in-memory, open-source data stores. Memcached, a high-performance distributed memory cache service, is designed for simplicity while Redis offers a rich set of features that make it effective for a wide range of use cases. They work with relational or key-value databases to improve performance, such as MySQL, Postgres, Aurora, Oracle, SQL Server, DynamoDB, and more.

## *Redis Features*

- Simple to install setup and manage
- Redis is very fast (1m qps)
- Redis is very fast because it loads the data from main memory
- Redis supports a variety of languages
- Redis supports master to slave replication

## *Redis Drawbacks*

- It is single threaded
- It has got limited client support for consistent hashing
- It has significant overhead for persistence

## *Redis vs RDBMS*

| Redis | RDBMS |
|---|---|
| NoSql | Sql |
| Key value structure | Table Structure |
| Fast | Slow |
| Stores in main memory | Stores in secondary memory |
| Generally stores small data | Stores any kind of data |
| Not follow ACID | Follow ACID |

## *AOF(Append-only files)*

- To avoid data loss in case of system crash Redis provides persistence by Snapshotting & AOF(Append-only files)
- AOF copies incoming write command to secondary memory
  - always — Every writes to redis, writes to seconday memory
  - everysec — Writes to disk every second

## *Redis String*

- Redis Strings are binary safe
- Redis string can contain any kind of data
- A String value can be at max 512 MB in length

  SET key value

  GET key

  SET key value NX | XX

  NX => set a value to the key if the key is not exist
  XX => set a value to the key only if the key already exists

# *Redis List*

- Redis Lists are simply lists of strings
- Lists are sorted by insertion order
- The max length of a list is $2^{32}$ — 1 elements
- List Commands
    - LPUSH — Inserts a new element on the head
        - *LPUSH A 1*
    - RPUSH — Inserts a new element on the tail
        - *RPUSH A 2*
    - LPOP — Removes an element from the head
    - RPOP — Removes an element from tail
    - LRANGE — Prints an element from the specified position
        - *LRANGE A 0 -1*

## *Redis Set*

- Redis Sets are an unordered collection of Strings.
- It is possible to add, remove, and get in O(1)
- The max number of members in a set is $2^{32}$ — 1
- Commands
  - SADD — Adds a new element to the SET
  - SMEMBERS — To retrieve values from the KEY
  - SISMEMBER — To verify whether specified element is present in the SET
  - SPOP — To pop the element out from the SET
  - SUNION — Union of all the given sets.
  - SINTER — Intersection of all the given sets.
  - SUNIONSTORE — This command is equal to SUNION, but instead of returning the resulting set, it is stored in destination.
  - SINTERSTORE — This command is equal to SINTER, but instead of returning the resulting set, it is stored in destination.

## *Redis Hashes*

- Redis Hashes are maps between string fields and string values, so they are the perfect data type to represent objects like user-details
- Every hash can store up to $2^{32}$ — 1 field-value pairs
- Commands
  - HMSET — Sets the specified fields to their respective values in the hash stored at key. *Time complexity:* O(N)
  - HGET — Returns the value associated with field in the hash stored at key. *Time complexity: O(1)*
  - HGETALL — Returns all fields and values of the hash stored at key. *Time complexity:* O(N)

## *Redis Sorted sets*

- Redis Sorted Sets are similar to Redis Sets with the unique feature of values stored in a set. The difference is, every member of a Sorted Set is associated with a score, that is used in order to take the sorted set ordered, from the smallest to the greatest score.
- The max number of members in a set is $2^{32}$ — 1
- Commands
  - ZADD — Adds all the specified members with the specified scores to the sorted set stored at key. *Time complexity:* O(log(N))
  - ZREM — Removes the specified members from the sorted set stored at key. Non existing members are ignored. *Time complexity:* O(M*log(N))
  - ZRANGE — Returns the specified range of elements in the sorted set stored at key. *Time complexity:* O(log(N)+M)
  - ZRANGEBYSCORE — Returns all the elements in the sorted set at key with a score between min and max. *Time complexity*: O(log(N)+M)

## *Refrences*


- *https://aws.amazon.com/redis/*
- *https://medium.com/@cosmicconvallis/interview-questions-on-redis-for-developers-c27769b410e9*