INFO 610 Fall 2020

Week 10, Oct 20, 2020

Redis - Fun Tuesday

Redis for SQL Experts

chrisfauerbach.github.io/info610_fall_2020/

Redis

https://redis.io/

Redis is an open source (BSD licensed), in-memory data structure store, used as a database, cache and message broker. It supports data structures such as strings, hashes, lists, sets, sorted sets with range queries, bitmaps, hyperloglogs, geospatial indexes with radius queries and streams. Redis has built-in replication, Lua scripting, LRU eviction, transactions and different levels of on-disk persistence, and provides high availability via Redis Sentinel and automatic partitioning with Redis Cluster.

Redis - When do you use it

As an In Memory data store, Redis doesn't touch the 'disk' of the computer it's running on.

As a result Redis is blazingly fast.

Redis can store/process/pass data at the rate of millions of times per second.

Redis is safe because it can store 'snapshots' on disk for local resiliency

Key Concepts

- 1) Data Storage
- 2) Transactions
- 3) Publish / Subscribe
- 4) Keys with TTLs
- Streams
- Lua Scripting
- LRU/LFU Eviction

Data Storage

Everything in Redis is a 'key'. Like a JSON key or even like a column in a relational database.

Keys are NOT like a 'table' in a RDBMS

Keys are more like a row of the table.

In fact, Redis doesn't have 'tables'. It doesn't have 'joins', etc.

Data types in Redis

Basic data types

- Strings - Default / Simple data type, all values are strings
- Lists - Collection of string elements sorted by order of insertion. 'Linked Lists'
- Sets - Collection of unique, unsorted strings
- Sorted Sets - Collection of unique, sorted strings
- Hashes - Maps composed of fields and associated values. Fields and values are strings

Math, Super Smart data types

- Bitmaps - Datastructure, like a String, that allows you to manipulate individual bits
- Hyperloglogs - Probabalistic data structure in order to estimate cardinality of a set.

Now it's all about commands!

The one reference: https://redis.io/commands

Group by different 'keys'.

e.g. Strings - Contains the basic commands for dealing with strings...

Most data types have common commands like 'GET', 'SET', 'DELETE', 'UPDATE', etc.

Let's check them out

Strings

- SET - (set the string value)
- SETEX - (with a TTL in 'seconds')
- PSETX - (with a TTL in 'milliseconds')

(CHF - Using the 'redis-cli' interactive prompt for examples)

```
127.0.0.1:6379> SET my_key_name "This is my key value"
OK
127.0.0.1:6379> GET my_key_name
"This is my key value"
127.0.0.1:6379> SETEX my_key_name 3 "Will dissapear in 3 seconds"
OK
127.0.0.1:6379> GET my_key_name
"Will dissapear in 3 seconds"
127.0.0.1:6379> GET my_key_name
(nil)
```

- STRLEN
- APPEND
- GETSET

A few interesting commands on 'strings'

- INCR / INCRBY / INCRBYFLOAT
- DECRBY / DECRBY / DECRBYFLOAT

Why are they interesting to me?

Numeric commands on strings!

```
127.0.0.1:6379> SET my_key 1
OK
127.0.0.1:6379> GET my_key
"1"
127.0.0.1:6379> STRLEN my_key
(integer) 1
127.0.0.1:6379> INCRBY my_key 10
(integer) 11
127.0.0.1:6379> GET my_key
"11"
127.0.0.1:6379> STRLEN my_key
(integer) 2
```

Lists

```
127.0.0.1:6379> LRANGE my_new_list 0 100
(empty list or set)
127.0.0.1:6379> RPUSH my_new_list one
(integer) 1
127.0.0.1:6379> RPUSH my_new_list two
(integer) 2
127.0.0.1:6379> RPUSH my_new_list thre
(integer) 3
127.0.0.1:6379> LRANGE my_new_list 0 100
1) "one"
2) "two"
3) "thre"
127.0.0.1:6379> RPOP my_new_list
"thre"
127.0.0.1:6379> RPUSH my_new_list three
(integer) 3
127.0.0.1:6379> LRANGE my_new_list 0 100
1) "one"
2) "two"
3) "three"
127.0.0.1:6379> LPOP my_new_list
"one"
127.0.0.1:6379> LRANGE my_new_list 0 100
1) "two"
```

Sets

```
127.0.0.1:6379> SADD my_new_set one two three four "multi string"
(integer) 5
127.0.0.1:6379> SMEMBERS my_new_set
1) "four"
2) "two"
3) "one"
4) "three"
5) "multi string"
127.0.0.1:6379> SADD my_other_set TWO two THREE three "this is a big string"
(integer) 5
127.0.0.1:6379> SMEMBERS my_other_set
1) "TWO"
2) "THREE"
3) "two"
4) "three"
5) "this is a big string"
127.0.0.1:6379> SINTER my_new_set my_other_set
1) "two"
2) "three"
127.0.0.1:6379> SISMEMBER my_new_set three
(integer) 1
127.0.0.1:6379> SISMEMBER my_new_set somethingnotthere
(integer) 0
127.0.0.1:6379> SRANDMEMBER my_new_set
```

Hashes

```
127.0.0.1:6379> HSET person_1 first_name "Chris"
(integer) 1
127.0.0.1:6379> HSET person_1 last_name "Fauerbach"
(integer) 1
127.0.0.1:6379> HSET person_1 email "chfauerbach@vcu.edu"
(integer) 1
127.0.0.1:6379> HGET person_1 email
"chfauerbach@vcu.edu"
127.0.0.1:6379> HGETALL person_1
1) "first_name"
2) "Chris"
3) "last_name"
4) "Fauerbach"
5) "email"
6) "chfauerbach@vcu.edu"
127.0.0.1:6379> HKEYS person_1
1) "first_name"
2) "last_name"
3) "email"
127.0.0.1:6379> HDEL person_1 email
(integer) 1
127.0.0.1:6379> HGETALL person_1
1) "first_name"
2) "Chris"
```

HYPERLOGLOG

My favorite.

The HyperLogLog data structure can be used in order to count unique elements in a set using just a small constant amount of memory, specifically 12k bytes for every HyperLogLog (plus a few bytes for the key itself).

The returned cardinality of the observed set is not exact, but approximated with a standard error of 0.81%.

```
127.0.0.1:6379> PFADD my_hll dog
(integer) 1
127.0.0.1:6379> PFADD my_hll cat
(integer) 1
127.0.0.1:6379> PFADD my_hll cat
(integer) 0
127.0.0.1:6379>
127.0.0.1:6379> PFADD my_hll two
(integer) 1
127.0.0.1:6379> PFADD my_hll three
(integer) 1
127.0.0.1:6379> PFADD my_hll "..."
(integer) 1
127.0.0.1:6379> PFADD my_hll "3million more"
```

Commands on Keys

```
127.0.0.1:6379> SET my_key 10
OK
127.0.0.1:6379> EXPIRE my_key 120
(integer) 1
127.0.0.1:6379> TTL my_key
(integer) 117
127.0.0.1:6379> TTL my_key
(integer) 115
127.0.0.1:6379> PERSIST my_key
(integer) 1
127.0.0.1:6379> TTL my_key
(integer) -1
127.0.0.1:6379> EXISTS my_key
(integer) 1
127.0.0.1:6379> DEL my_key
(integer) 1
127.0.0.1:6379> EXISTS my_key
(integer) 0
127.0.0.1:6379> KEYS *
1) "my_hll"
2) "person1_"
3) "my_new_set"
4) "my_new_list"
5) "person_1"
```

Mikes Bikes

- How do we model customers?
- How do we model bikes?
- How do we 'store' multiple customers?
- How do we 'query' products for a web site?

See you Thursday!