Hello!

I am Esmaeil Kazemi

I'm interested in learning how are you?

You can find me at @eskazemi

NOSQL

vs

SQL

Redis

Redis stands for Remote Dictionary Server

eskazemi

Map

Redis

1- Features

2- application

3- data type

4- Message Queue

5- Transactions

6- Pipelining

7- Lua Scripts

8- Persistence

9- Benchmarks

10- configuration

11- ACLs

12- Redis Cluster

13- Redis vs Memcached

14- Redis vs Hazelcast

15- Redis vs RDBMS

eskazemi

ویژگی ها

- از نوع دیتابیس های key - value
- اطلاعات را ذخیره می کند داخل رم (in-memory) که باعث می شود سرعت خواندن و نوشتن اطلاعات افزایش پیدا کند.
- با زبان C نوشته شده است
- از Lua Scripting پشتیبانی می کند
- به صورت پیش فرض replication دارد
- دارای partitioning از طریق Redis Cluster هست
- پورت پیش فرض اون 6379 هست
- به صورت open source و تحت License BSD نگه داری می شود.
- فرآیند نصب ساده ای دارد و مدیریت آن آسان است.
- قابل حمل هست و می توان از آن ها در سیستم های مختلف استفاده کرد.

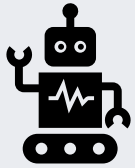
ویژگی ها

- ✓ Redis به سرعت مشهور است چون توانایی اجرای query 100000 در ثانیه را دارد. سرعت خود را با حفظ داده ها در حافظه و همچنین ذخیره آنها بر روی دیسک به دست می آورد.

- ✓ انواع زبان های مختلف چون python , c , c++, Ruby, java و ... می توانند به پایگاه داده redis متصل شوند و از آن استفاده کنند. در صورتی که زبان مورد نظر شما پشتیبانی نمی شود، می توانید کتابخانه client خود را ایجاد کنید زیرا پروتکل Redis ساده است.

- ✓ عملیات Redis بر روی انواع داده های مختلف Atomicity است و اجرای ایمن وظایفی مانند ایجاد کلیدها، اضافه کردن / حذف عناصر مجموعه ، افزایش شمارنده ها و... را تضمین می کند.

موارد استفاده از آن



دیتابیس اصلی

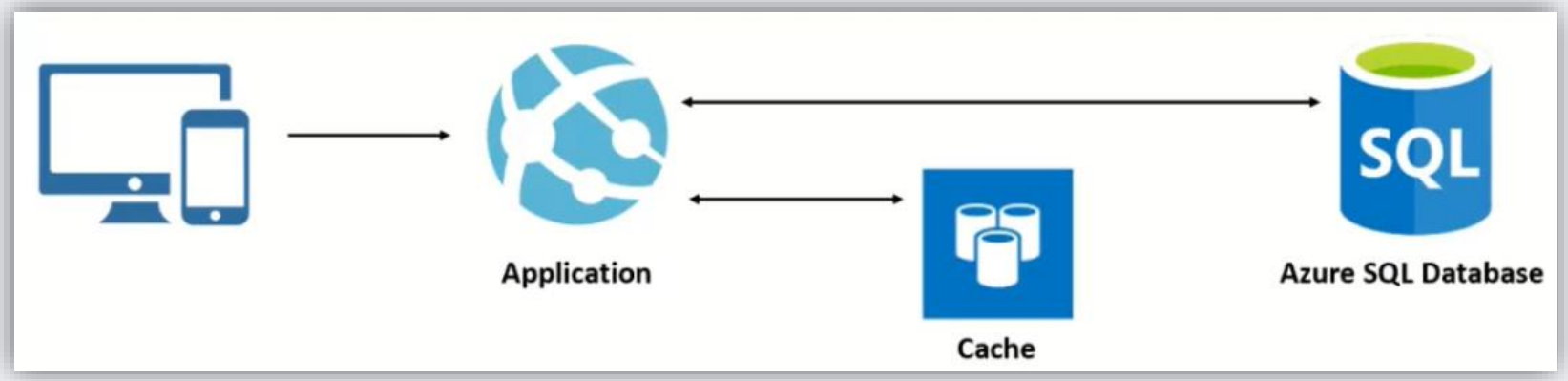
حافظه نهان (cache)

message broker

What is Caching?

حافظه نهان یک نوع حافظه بسیار سریع است که به عنوان یک بافر بین RAM و CPU عمل می کند.

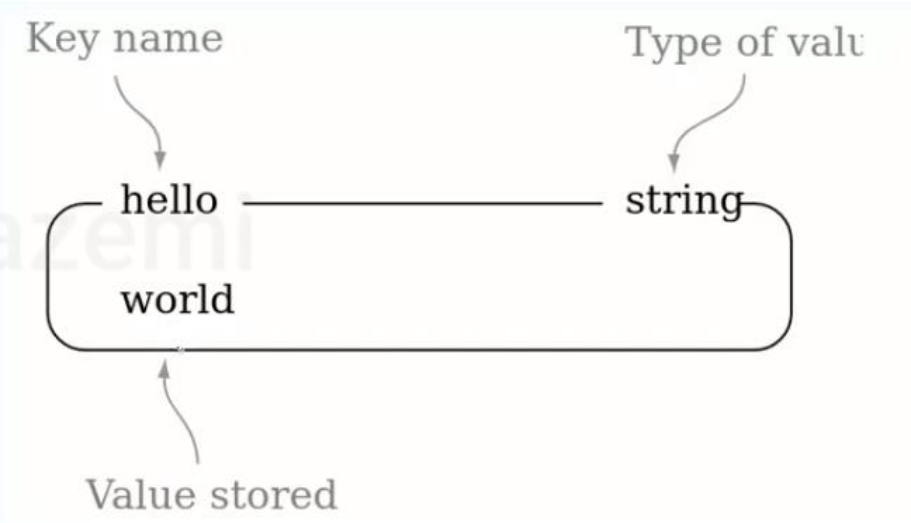
Cache memory is an extremely fast memory type that acts as a buffer between RAM and the CPU. It holds frequently requested data and instructions so that they are immediately available to the CPU when needed. Cache memory is used to reduce the average time to access data from the Main memory. So data can be served faster by the cache.



توصیه های مختلف به هنگام استفاده از Redis

یکی از ایرادات این نوع پایگاه داده ها عدم وجود ویژگی سازگاری در آنها است. اگر داده‌هایی که می‌خواهیم ذخیره کنیم دارای ارتباطات متفاوتی باشند، پایگاه‌های داده کلید-مقدار عملکرد خود را از دست می‌دهند. این پایگاه های داده معمولا زمانی پیشنهاد می شوند که حجم داده های نوشته شده در پایگاه داده کم و خوانده شده زیاد باشد.

مدل ذخیره سازی اطلاعات در Redis



SET and KEY

SET

برای ایجاد کردن کلید و تعیین مقدار برای آن از دستور SET استفاده می کینم

GET

برای خواندن اطلاعات یک کلید از دستور GET استفاده می کنیم.

DEL

برای حذف یک کلید

EXISTS

برای چک کردن آیا یک کلید وجود دارد یا نه؟

KEYS

برگرداندن کلید ها براساس یک الگو (استفاده از regex)

TTL

مقدار زمان عمر یک کلید بر می گرداند

EXPRIE

you can define an expiration time or time to live abbreviated to TTL.
Expiration times can be set in milliseconds, seconds or a UNIX timestamp.

PERSIST

از بین بردن مقدار زمانی که برای عمر یک کلید در نظر گرفته شده است و ثبات بخشیدن به آن

RENAME

تغییر نام یک کلید

TYPE

برگداندن نوع ساختار داده یک کلید

Features KEYS

- نام کلید ها منحصر به فرد (unique) در دیتابیس منطقی
- کلیدها راه اصلی برای دسترسی به مقادیر داده در Redis هستند.
- کلیدهای Redis (binary safe)هستند، به این معنی که شما می توانید از هر دنباله باینری به عنوان کلید استفاده کنید(Any sequence of bytes)، مانند "foo"/“0xff/45/3.1415 تا محتوای یک فایل JPEG ، رشته خالی نیز یک کلید معتبر است
- حجم نام کلید ها کمتر از 512 مگابایت باشد. با این حال، کلیدهای بسیار بلند عموما توصیه نمی شوند.
- اگر کلیدی موجود باشد و دوباره برای کلید مقداری الصاق کنیم کلید بازنویسی می شود و مقدار جدید را می گیرد.
- Redis دیتابیس منطقی است و نام کلید ها منحصر به فرد می باشد اما نام های مشابه برای کلید می توانیم در چندین دیتابیس منطقی داشته باشیم.

Features KEYS

دو نوع کلید در Redis

Volatile

a key with an expire set

Persistent

a key that will never expire as no timeout is associated

Commands KEYS Vs SCAN

KEYS	SCAN
Blocks until complete	Iterates using a cursor
Never use in production	Returns a slot reference
Useful for debugging	May return 0 or more keys per call
	Safe for production

Using SCAN – Examples

```
127.0.0.1:6379> MSET key1 value1 key2 value2 key3 value3 key4 value4 key5 value5 key6 value6
OK
```

we can use the SCAN command to iterate over the keys.

1- SCAN یک تکرارکننده مبتنی بر مکان نما است. این بدان معنی است که در هر فراخوانی از فرمان، سرور یک مکان نمای به روز شده را برمی گرداند که کاربر باید از آن به عنوان آرگومان مکان نما در فراخوانی بعدی استفاده کند.

```
127.0.0.1:6379> SCAN 0
1) "0"
2) 1) "key4"
2) "rq:finished:default"
3) "key5"
4) "key6"
5) "key2"
6) "key3"
7) "rq:queues"
8) "key1"
```

2- مقدار اول مکان نمای جدید برای استفاده در فراخوانی بعدی، مقدار دوم آرایه ای از عناصر است.

3- دستور SCANتنها ده کلید اول در پایگاه داده را برمی گرداند. از آنجا که دستور SCANمی تواند ده عنصر اول را برگرداند.

4- شروع یک تکرار با مقدار مکان نمای ۰، و فراخوانی SCANتا زمانی که مکان نمای بازگشتی دوباره ۰ شود، یک تکرار کامل نامیده می شود.

SCAN cursor [MATCH pattern] [COUNT count]

آرگومان مکان نما در فراخوانی بعدی

دستور شمارش به شما اجازه می دهد تا تعداد کلیدهایی که دستور SCANدر هر فراخوانی می آورد را تغییر دهید. به طور پیش فرض دستور SCANده کلید دارد.

Commands DEL Vs UNLINK

- DEL key [key ...]

- UNLINK key [key ...]

UNLINK برخلاف DEL به صورت asynchronous فرایند حذف کلید را انجام می دهد و بلاک نمی کند ترمیمنال را

Encoding verified before command execution

دستور Redis OBJECT برای بررسی داخلی های Redis Objects مربوط به کلیدها استفاده می شود. برای اشکال زدایی یا درک اینکه آیا کلیدهای شما از انواع داده های کدگذاری شده(encoding) خاص برای صرفه جویی در فضا استفاده می کنند یا خیر، مفید است.

فرمان OBJECT از چندین فرمان فرعی پشتیبانی می کند:

- OBJECT REFCOUNT <key> returns the number of references of the value associated with the specified key. This command is mainly useful for debugging.

- OBJECT ENCODING <key> returns the kind of internal representation used in order to store the value associated with a key.

Encoding verified before command execution

The usage OBJECT encoding key

encoding data type

- Strings: int, embstr, raw
- Lists: ziplist (small lists), linkedlist
- Sets: intset (integers and small sets), hashtable
- Sorted Sets: ziplist (small sets), skiplist
- Hashes: zipmap (small hashes), hashtable

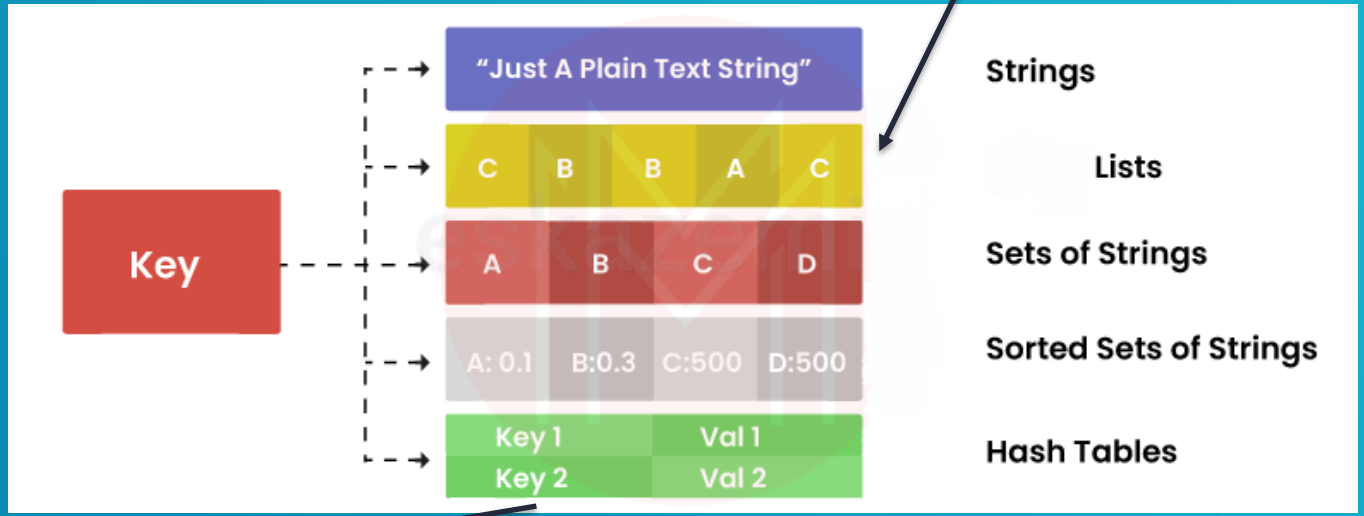
Data type

انواع ساختمان داده که Redis پشتیبانی می کند

1. String
2. hashes
3. Sets
4. Sorted sets
5. Lists
6. Bitmap
7. Stream
8. hyperloglogs

value

Valueمی تواند یه لیست باشد.



می تواند key-value باشد.

string

- حداکثر حجم هر string می تواند 512 مگابایت باشد.
- شما می توانید تقریبا هر چیزی که می توانید تصور کنید را در یک string ذخیره کنید. Integers، مقادیر باینری، مقادیر جدا شده از هم، JSONسریال شده. و از آنجا که آن ها امن باینری هستند، شما حتی می توانید اشیا بزرگ تر مانند تصاویر، ویدیوها، اسناد و صدا را ذخیره کنید
- Redis تمام مقادیر به صورت default به صورت string می بینه .

So what are some practical uses for Redis strings?

The most common use is for caching: API responses, session storage, and HTML pages.
Redis strings are also great for implementing counters, as there is built-in support for incrementing and decrementing integer and floating point values.

string commands

	command	description
1	SET key value	This command sets the value at the specified key.
2	GET key	Gets the value of a key.
3	GETRANGE key start end	Gets a substring of the string stored at a key.
4	GETSET key value	Sets the string value of a key and return its old value.
5	SETEX	Sets the value with the expiry of a key
6	SETNX	Sets the value of a key, only if the key does not exist

string commands

	command	description
7	STRLEN key	Gets the length of the value stored in a key
8	MSET key value [key value ...]	Sets multiple keys to multiple values
9	MSETNX key value [key value ...]	Sets multiple keys to multiple values, only if none of the keys exist
10	INCR key	Increments the integer value of a key by one
11	INCRBYFLOAT key increment	Increments the float value of a key by the given amount
12	DECR key	Decrements the integer value of a key by one

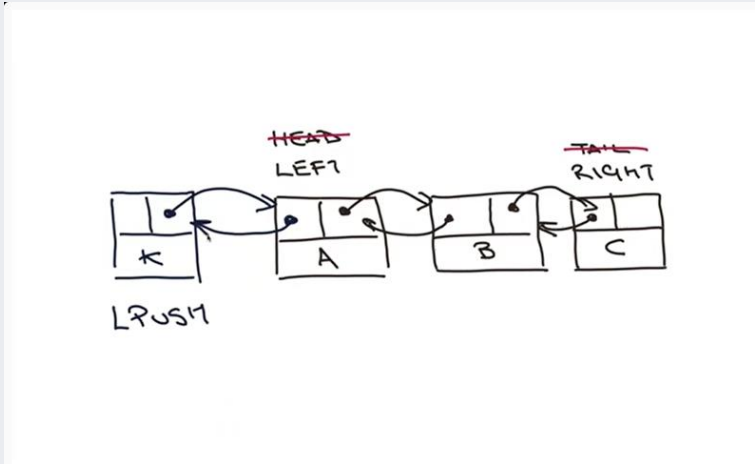
- محتویات رشته را می توان بین متن، عدد، باینری در هر نقطه تغییر داد. برایRedis، همیشه یک نوع داده رشته ای است.
- این عمل هیچ اشکالی ندارد چون Redis پشتیبانی می کند از چند ریختی (polymorphism)
- نکته ای که باید در نظر بگیرید وقتی دارید از دستور DECRBY استفاده می کنید این دستور روی مقادیری که به integer کد شده اند (encoded as integer) عمل می کند.
- Redis از دو معیار Datatype و Encoding of value برای تعیین اینکه آیا می توان عملیاتی را روی یک کلید انجام داد یا خیر، استفاده می کند.

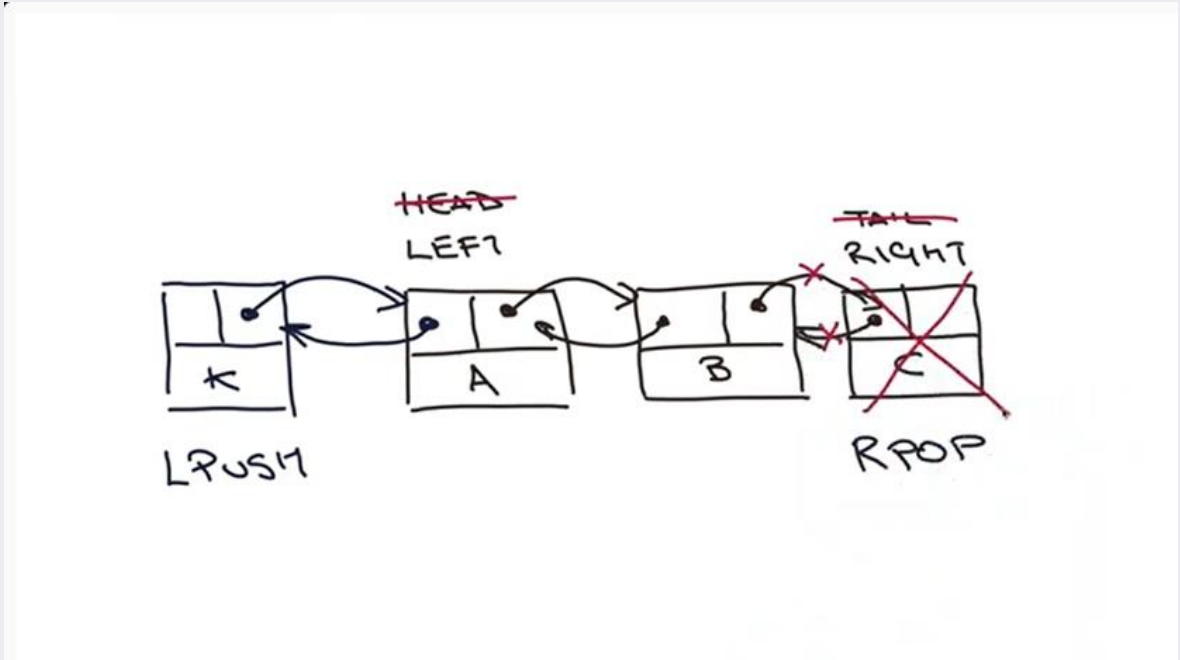
List

- یک ساختار اساسی برای ذخیره و دستکاری مجموعه منظمی از عناصر فراهم می کنند. این ترتیب در طول درج و حذف عناصر حفظ می شود در نتیجه المان های لیست به ترتیب ورود طبقه بندی می شوند یعنی ترتیب ورود اطلاعات مهم است.
- با استفاده از لیست ها می توانند چندین مقدار داخل یک کلید ذخیره بکنیدو اجازه تکرار مقادیر را می دهد.
- هر لیست می تواند بیشتر از 4 میلیارد المان در خود ذخیره کند.
- راه ساده ای را برای پیاده سازی پشته ها(stacks)، صف ها(queues) و دیگر ساختارهای داده ای که به ترتیب متکی هستند، ارائه می دهند.
- زیربنای تمام ساختارها در Redisنوع داده رشته ای است. بنابراین یک ساختار می تواند تنها از رشته ها تشکیل شده باشد، نه لیست ها، مجموعه ها، مجموعه های مرتب شده، یا هش ها. بنابراین هیچ مفهومی از nested، سلسله مراتب یا گراف در این ساختارهای داده ای وجود ندارد.
- به صورت داخلی Redis لیست ها را با استفاده از یک لیست پیوندی(doubly linked list) دو طرفه پیاده سازی می کند.

doubly linked list

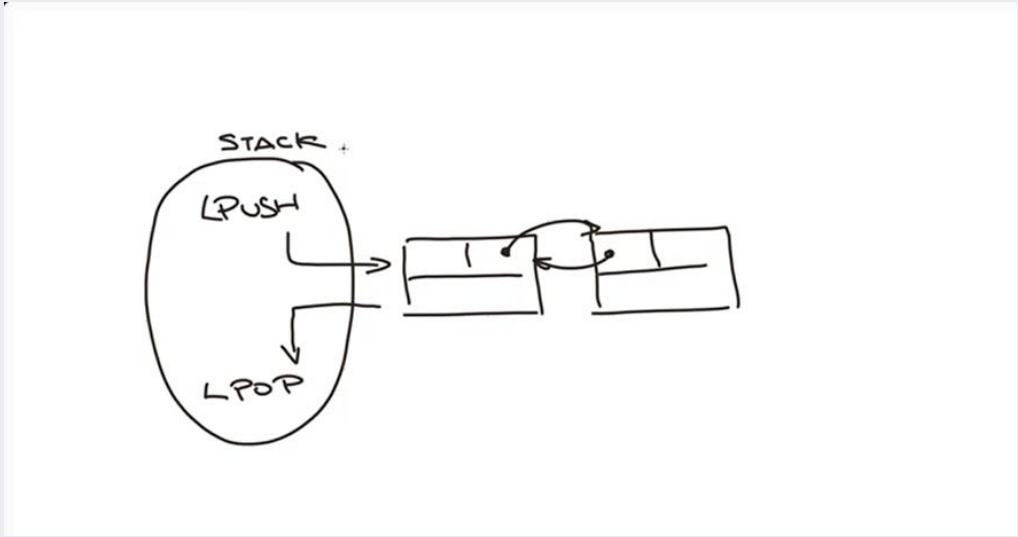
Let's take a look at how lists are organized. We have three elements. Each element has a value A, B, and C. Redis uses a doubly linked list so element A has a pointer to the next element B and B has a previous pointer to A. B and C are similarly related. Typically the element on the left of this list is referred to as the head. The element on the right is the tail. In Redis, we simplify this just to left and right.




HEAD
LEFT
TAIL
RIGHT
K
A
B
C
LPUSH
RPOP

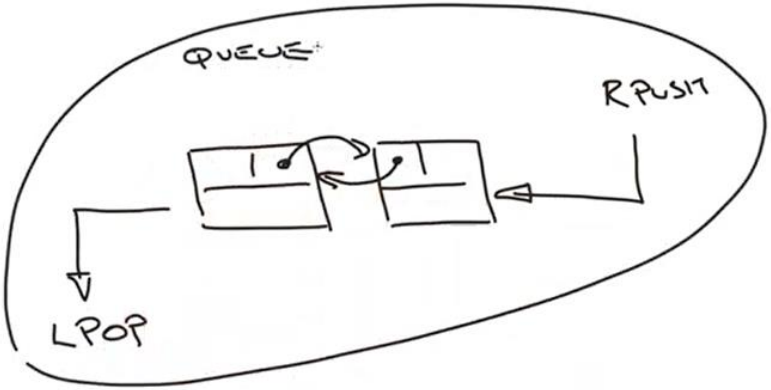
Stack

By adding new elements to the left and removing elements from the left we have implemented a stack construct out of the list datatype.



Queue

If new elements are added to the right of the list and removed from the left of the list we've implemented a queue.



Use Cases

برای مثال زمانی که نیاز به دانستن جدیدترین فعالیت ها دارید، جدیدترین پست ها در یک جریان فعالیت (activity stream) مثلا فیسبوک یا Slack

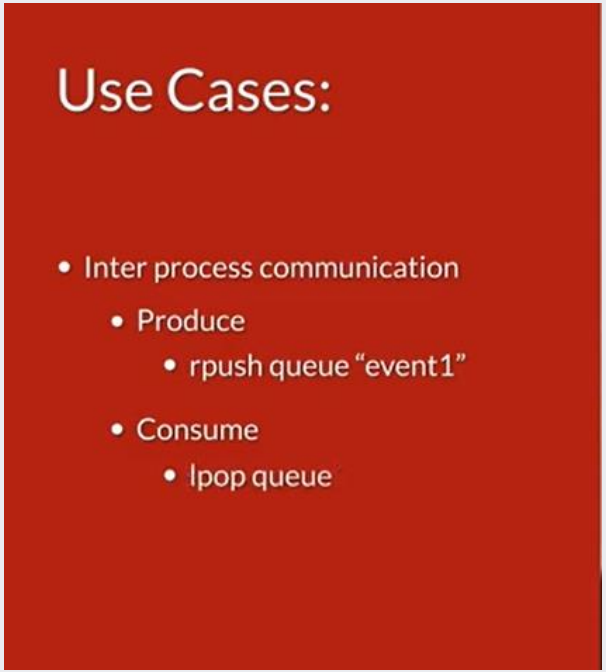
LPUSH می توانند مواردی را به سمت چپ لیست اضافه کنند.

برای به دست آوردن سه مورد آخر از ورودی های LRANGE می توان استفاده کرد. در این حالت صفر تا دو را برای به دست آوردن سه عنصر آخر مشخص می کنیم. به یاد داشته باشید که شاخص ها مبتنی بر صفر هستند.

از LTRIM می توان برای مرتب کردن لیست با حذف عناصر سمت راست استفاده کرد.

از LTRIM می توان برای مرتب کردن لیست با حذف عناصر سمت راست استفاده کرد. در واقع ما قدیمی ترین ورودی ها را هرس می کنیم. در این حالت می خواهیم چهار عنصر را حفظ کنیم بنابراین صفر تا سه را مشخص می کنیم.

Use Cases


Use Cases:
• Inter process communication
• Produce
• rpush queue “event1”
• Consume
• lpop queue

List commands

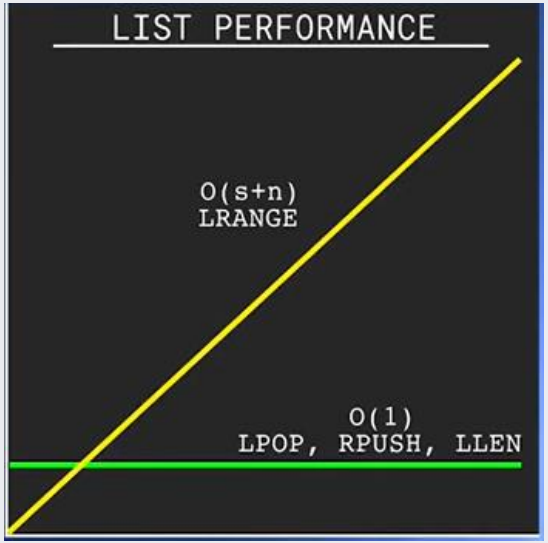
	command	description
1	LPUSH key value1 [value2]	Prepends one or multiple values to a list
2	RPUSH key value1 [value2]	Appends one or multiple values to a list
3	LPUSHX key value	Prepends a value to a list, only if the list exists
4	RPUSHX key value	Appends a value to a list, only if the list exists
5	LINDEX key index	Gets an element from a list by its index

List commands

	command	description
6	LLEN key	Gets the length of a list
7	LPOP key	Removes and gets the first element in a list
8	LRANGE key start stop	Gets a range of elements from a list
9	LSET key index value	Sets the value of an element in a list by its index
10	BRPOP key1 [key2] timeout	Removes and gets the last element in a list, or blocks until one is available

List commands

دستورات مهم توضیحات بیشتر :



complexity

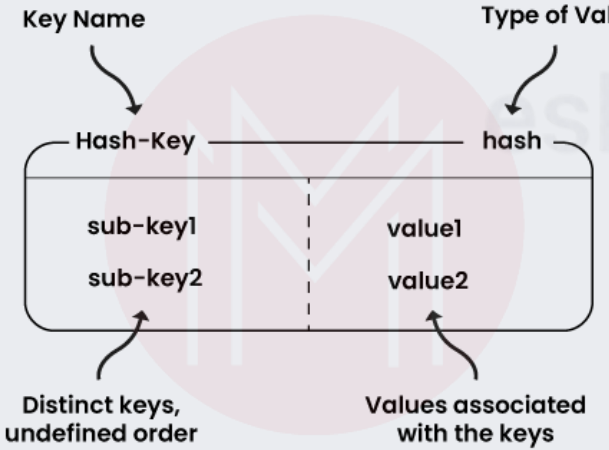
LPUSH RPUSH

هر دو ایجاد لیست با این تفاوت که LPUSH آخرین مقداری
که بهش داده میشه رو اول لیست قرار میده

BLPOP

برای حذف المان از لیست به کار می رود نکته ای که حائز اهیمت این است که زمان رو به عنوان ورودی میگیرد که اگر زمان صفر ست شود المان رو حذف می کنه اگر وجود داشته باشد و اگر نباشه اینقدر صبر می کنه تا داخل لیست المان قرار گیرد و اگر غیر صفر ست شود بعد از تایم مشخص شده غیر فعال می شود(بی خیال حذف می شود)

Hash



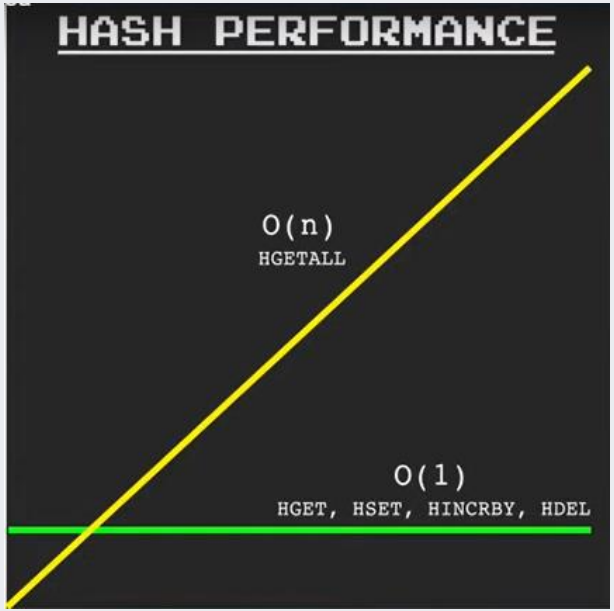
- نوعی از داده هستند که می توانید در یک کلید چندین مقدار به شکل کلید/مقدار ذخیره کنید.
- بیشترین استفاده از hash ها در Redis برای ذخیره آبجکت هاست.
- Hash ها توانایی ذخیره بیشتر از 4 میلیارد المان در خود را دارند.
- ما می توانیم جفت های مقدار فیلد را در هر زمان اضافه، تغییر، افزایش و حذف کنیم، نه فقط در اعلان اولیه. هش ها مقادیر فیلد را به صورت String ذخیره می کنند، فقط یک سطح است پشتیبانی می شود(single level) بنابراین نمی توانید لیست ها، مجموعه ها یا ساختارهای دیگر را در یک هش جاسازی کنید.
- Redis Hashs بدون طرح است، شما هنوز هم می توانید آن ها را به عنوان اشیا سبک وزن یا به عنوان ردیف در یک جدول پایگاه داده رابطه ای در نظر بگیرید.

Hash commands

	دستور	توضیح
1	HSET key field value	Sets the string value of a hash field
2	HMSET key field1 value1 [field2 value2]	Sets multiple hash fields to multiple values
3	HGET key field	Gets the value of a hash field stored at the specified key.
4	HGETALL key	Gets all the fields and values stored in a hash at the specified key
5	HDEL key field2 [field2]	Deletes one or more hash fields.

Complexity Commands

Most Hash commands are O(1), which means they will always perform a task in a constant amount of time,
regardless of the size of the Hash.
Constant time commands are as efficient as it gets.
The HGETALL command is O(n), with n being the number of fields within a Hash.
This means that the amount of time for the task to complete is dependent on how many fields it has to retrieve.
In this case, the n of O(n) is the number of fields that the Hash contains.
HGETALL is practical for relatively small Hashes,



Use Cases

Use Cases:

- Session cache
  - hmset session:a3fWa ts 1518132669 host www.example.org
  - hincrby session:a3fWa requests 1
  - expire session:a3fWa 60

Use Cases:

- Rate Limiting
  - hmset ep-20180210 "/pet/{petid}" 100 "/booking/{petid}" 100
  - hincrby ep-20180210 "/pet/{petid}" -1
  - expire ep-20180210 86400

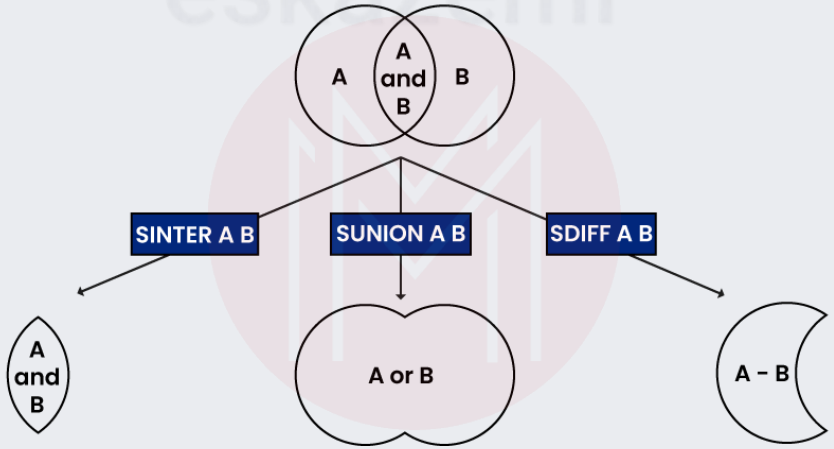
SET

- مجموعه ای از المان هایی String است که به شکل نامنظم در یک کلید ذخیره می شوند که هیچ تکراری ندارند.
- هر set توانایی ذخیره بیشتر از 4 میلیارد المان در خود را دارند.
- از عملیات های استاندارد مجموعه ریاضی مانند اشتراک، اجتماع و تفاضل پشتیبانی می کند.

SUNION و SINTER به چه معناست؟

SUNION مجموعه ای برگردانده می شود که از اجتماع تمامی مجموعه هایی که داده می شود حاصل می شود.

SINTER مجموعه ای برگردانده می شود که از اشتراک همه مجموعه هایی که داده می شود حاصل می شود.



Set commands

	دستور	توضیح
1	SADD key member1 [member2]	Adds one or more members to a set
2	SCARD key	Gets the number of members in a set
3	SPOP key	Removes and returns a random member from a set
4	SREM key member1 [member2]	Removes one or more members from a set
5	SRANDMEMBER key [count]	Gets one or multiple random members from a set.
6	SMEMBERS key	Gets all the members in a set
7	SISMEMBER key member	Determines if a given value is a member of a set

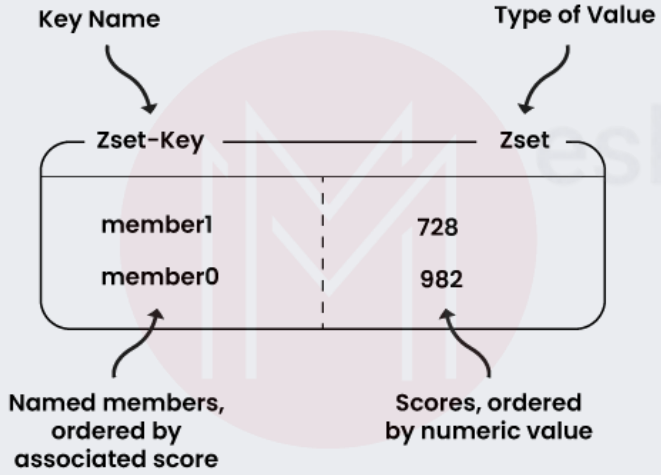
Use Cases

Creating a tag cloud is pretty simple. For each object we want to tag, a separate list of tags is maintained. We can find all the tags for a specific object using SSCAN. To find the tags that match across objects is simple. SINTER allows the intersection to be created and the matching tags returned. So it's simple to find that "wrench" and "coin" both have the tag "metal". To create a set of all the known tags, SUNION can be used to create the Union between the sets specified.

Use Cases

در مثالی دیگر، فرض کنید می‌خواهید بازدیدکنندگان منحصربه‌فرد هر صفحه را برای مدت زمان معینی در وب‌سایت خود ثبت کنید. برای هر URL منحصر به فرد و دوره زمانی می توان مجموعه ای ایجاد کرد. به عنوان مثال " "about.html و 10 فوریه 2018. کوکی هر بازدید کننده را می توان به عنوان عضوی از مجموعه ثبت کرد. این مجموعه اکنون شامل تمام بازدیدکنندگان منحصربه‌فرد آن صفحه در بازه زمانی است که با دستور SSCAN قابل بازیابی هستند. از EXPIRE می توان برای تعریف دوره نگهداری برای متریک، در این مورد، 86400 ثانیه یا یک روز استفاده کرد. از آنجایی که انقضا به کلید مربوط می شود نه عناصر موجود در مجموعه، پس از انقضای کلید همه عناصر حذف می شوند.

ZSET



- این نوع داده دقیقا شبیه setهاست و نمی توان المان های تکراری در آن ذخیره کرد . تنها تفاوتی که این دو باهم دارند این است که المان ها sorted_set یک score یا امتیاز دارند
- (از نوع float می باشد) که Redis المان ها را به ترتیب این score ها ذخیره می کند. در این نوع داده المان ها نمی توانند موارد تکراری داشته باشند اما امتیاز ها می توانند تکراری شوند.
- اگر دو المان با امتیاز یکسان داشته باشید، تساوی با ترتیب واژگانی المان ها شکسته می شود. مجموعه های مرتب را می توان به چند روش دستکاری کرد: با مقدار شون (value)، با موقعیت یا رتبه، با امتیاز و lexicography.
- همچنین، می‌توانیم به ترتیب مرتب‌سازی شده و همچنین مقادیر امتیازات به موارد مجموعه دسترسی داشته باشیم.
- در ZSET ها اعمال خواندن ، اضافه کردن ، حذف کردن و آبدیت کردن المان ها بسیار سریع است.

دستورات مهم:

ZADD: تمام اعداد مشخص شده با امتیازات مشخص شده به مجموعه مرتب شده ای که در کلید ذخیره شده است , اضافه می شوند.

ZREM : اعداد مشخص شده از مجموعه مرتب شده که در کلید ذخیره شده اند حذف می شوند . اعضای غیر موجود را نادیده می گیرد .

ZRANGE: مقادیر را براساس موقعیت و نه براساس امتیاز باز می گرداند. بنابراین این کار نتایج صحیح را برمی گرداند، اما در صورت اضافه یا حذف اعضا از مجموعه مرتب شده یا تغییر امتیاز اعضا، مقادیر نادرست را بر می گرداند.

ZRANGEBTSCORE: ارزش ها را براساس امتیاز عضو برمی گرداند. نماد پارس ")" نشان دهنده یک محدوده انحصاری است که الزامات بیش از ۳ را برآورده می کند.

zrangebyscore hw1-8 (3 +inf

Key

h

Value

completion	score
hi	50
hello	44
help	30
how are you	30
happy	29
how many	29
here	28
...	...

اگر امتیاز ها برابر باشند ، completion براساس حروف الفبا مرتب می شوند

به طور خلاصه:

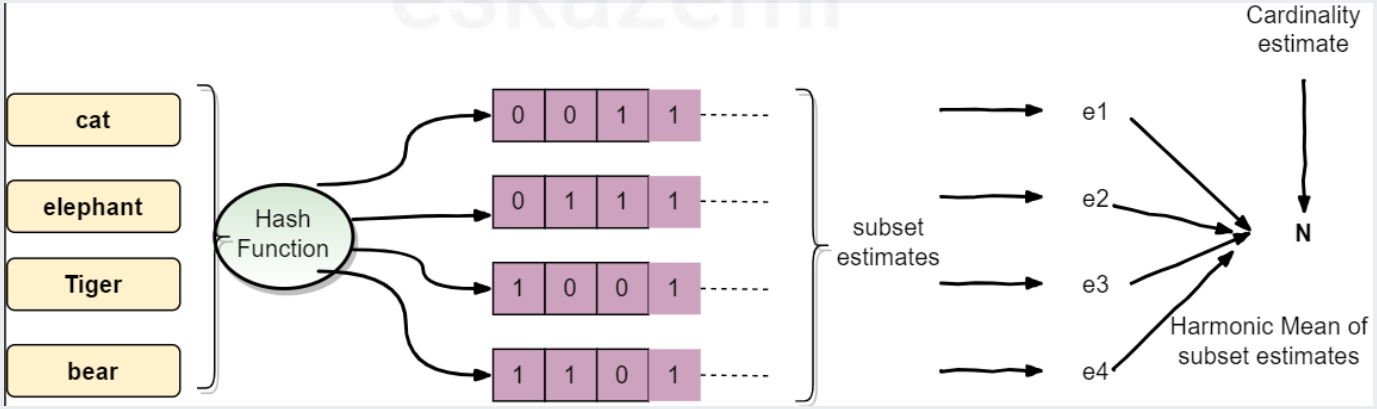

Sorted Sets:
Ordered collection of unique strings
Floating point Score
Manipulation by value, position, score or lexigraphically
Set commands Sorted Sets:
Union
Intersection
Typical Use Cases:
Priority Queue
Leaderboards

hyperloglog

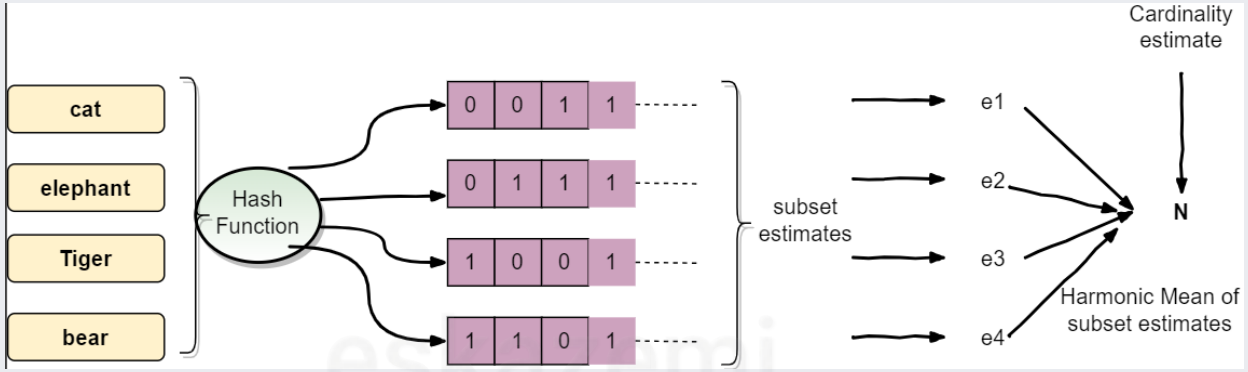
- مثل set ها هستند . یعنی امکان ذخیره موارد تکراری وجود ندارد و اینکه ترتیب ورود اطلاعات مهم نیست.
- این ساختار برای زمانی مناسب است که خود اطلاعات اهمیتی ندارند و فقط تعداد آنها مهم است .
- مثلا برنامه ای که به صورت real-time نیاز به ذخیره تعداد کاربر های Uniqe دارد یا اگر شما نیاز باشه تعداد ایمیل های uniqe در یک مجموعه 10 میلیون آدرس ایمیل بدست بیارید. جایی که نیمی از ایمیل ها unique باشند مقدار حافظه کافی برای ذخیره این اطلاعات نیازه که می توان از این نوع داده استفاده کنید که حل کرده این قضیه را با حداقل حافظه با استفاده از الگوریتم randomization algorithm (مخترع این الگوریتم ,Philippe Flajolet به همین خاطر که در ابتدای دستورات مربوط به این نوع داده از PF استفاده می کنیم.)
- بسیار بهینه تر از setها هستند. استفاده از منابع نسبت به set ها کمتر .

Algorithm Hyporloglog

این الگوریتم بر روی اعداد تصادفی توزیع شده یکنواخت کار می کند و برای تخمین تعداد آیتم های منحصر به فرد در یک مجموعه استفاده می شود. مفهوم کلیدی پشت این الگوریتم این است که اگر حداکثر تعداد صفرهای پیشرو در آیتم های مجموعه را بشماریم (که به اعداد تصادفی یکنواخت تبدیل شده اند و در باینری نمایش داده می شوند)، می توانیم تعداد آیتم های منحصر به فرد را با یک محاسبه ساده تخمین بزنیم.



If the maximum number of leading zeros is *n*, the estimated cardinality of the set is 2^n



فرض بگیرید می خواهیم تعداد بازدید های منحصر به فرد از یک سایت را بشماریم و ما داریم مجموعه ای بزرگ از IP های پشت سرهم که بازدید کردن از سایت ما:

ابتدا آدرس های IP ورودی را با استفاده از تابع هش به مجموعه ای از اعداد تصادفی توزیع شده یکنواخت تبدیل می کنیم cardinality .در اینجا تغییر نمی کند زیرا تنها آدرس های IP به اعداد تصادفی توزیع شده یکنواخت تبدیل می شوند.

این اعداد تصادفی با استفاده از چند بیت اولیه به زیرمجموعه های مختلفی تقسیم می شوند .تعداد حداکثر صفرهای پیشرو، در مقادیر آن، در حافظه ذخیره می شوند.

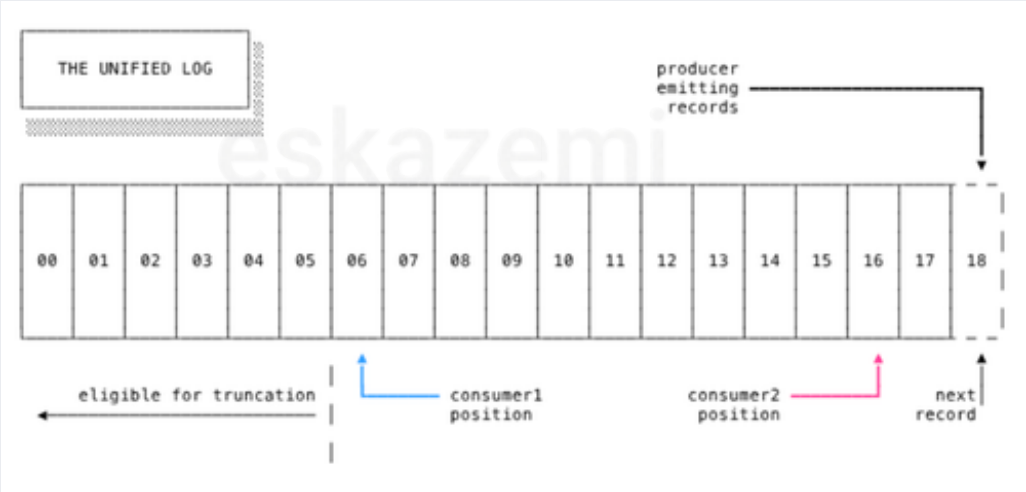
ما میانگین هارمونیک برآوردها را برای تمام زیرمجموعه های محاسبه شده قبلی محاسبه می کنیم.

میانگین هارمونیک محاسبه شده در بالا تخمینی از تعداد بازدید کننده منحصر به فرد در صفحه وب است.

Hyperloglog commands

	دستور	توضیح
1	PFADD key element [element ...]	Adds the specified elements to the specified HyperLogLog.
2	PFCOUNT key [key ...]	Returns the approximated cardinality of the set(s) observed by the HyperLogLog at key(s)
3	PFMERGE destkey sourcekey [sourcekey ...]	Merges N different HyperLogLogs into a single one.

Streams



Streams

- Redis Streams یک ساختار داده فوق العاده قدرتمند برای مدیریت جریان های داده با سرعت بالا (مانند صف پیام)است.
- با پارتیشن بندی (out-of-the-box)، تکرار و پایداری، می تواند میلیون ها نقطه داده در ثانیه را با تاخیر زیر میلی ثانیه دریافت و پردازش کند.
- مبتنی بر پیاده سازی کارآمد radix-tree است (الگوریتمی که در آن هر گره که تنها فرزند است با والد ادغام می شود)، که دامنه و جستجوی پرس و جوها را بسیار سریع می کند.
- تولیدکنندگان(producers) و مصرف کنندگان (consumers) را با تماس های ناهمزمان متصل می کند و از گروه های مصرف کننده(consumer) پشتیبانی می کند.

Bitfields

Bitfields روشی فشرده و کارآمد برای پیاده سازی شمارنده های چندگانه در یک آرایه واحد ارائه می دهند. این کار امکان افزایش و کاهش شمارنده ها در یک موقعیت مشخص را فراهم می کند، و هنگامی که شمارنده به حد بالایی خود می رسد، flags ها سرریز می شوند.

offset 0	offset 1	offset 2	offset 3	offset 4
1	0	0	1	1

Geospatial

Geospatial indexing تکنیکی است که در پایگاه های داده برای ذخیره و بازیابی کارآمد داده ها براساس موقعیت جغرافیایی آنها استفاده می شود . این شامل indexing داده های مکانی در یک پایگاه داده با استفاده از یک ساختار داده تخصصی است که می تواند به سرعت تشخیص دهد که کدام اشیاء یا نقاط داده در یک منطقه جغرافیایی خاص قرار دارند.
Geospatial indexing به ویژه در برنامه هایی مفید است که با حجم زیادی از داده های جغرافیایی سروکار دارند، مانند برنامه های کاربردی نقشه برداری، خدمات مکان یابی جغرافیایی و تجزیه و تحلیل مکانی. با استفاده از Geospatial indexing ، این برنامه ها می توانند به سرعت داده ها را بر اساس موقعیت مکانی خود پرس و جو و تجزیه و تحلیل کنند، بدون اینکه نیازی به اسکن مقادیر زیادی از داده ها برای یافتن اطلاعات مربوطه باشد.

Redis چندین دستور مربوط به Geospatial indexing (دستورها GEO) دارند اما برخلاف دستورهای دیگر، این دستورها فاقد نوع داده خاص خود هستند. این دستورها در واقع براساس نوع داده sorted set تنظیم می شوند. این کار با کدگذاری طول و عرض جغرافیایی در امتیاز sorted set با استفاده از الگوریتم geohash به دست می آید.

Geospatial

برای مثال فرض بگیرید 4 مکان با مختصات داریم:

- San Francisco (lat: 37.774, lng: -122.419)
- Palo Alto (lat: 37.441, lng: -122.143)
- Mountain View (lat: 37.386, lng: -122.083)
- San Jose (lat: 37.338, lng: -121.886)

How to index spatial entities?

GEOADD command can be used to index spatial entities in Redis. The time complexity for this command is O(log(N)) for each item added, where N is the number of elements in the sorted set.

Geospatial

اضافه کردن 4 لوکیشن با کلید ‘places’ با اجرای دستور زیر:

```
> GEOADD places -122.419 37.774 “San Francisco”
> GEOADD places -122.143 37.441 “Palo Alto”
> GEOADD places -122.083 37.386 “Mountain View”
> GEOADD places -121.886 37.338 “San Jose”
```

از آنجا که این موجودیت ها در یک sorted set ذخیره می شوند می توان از دستورات مربوط به sorted set استفاده کرد ، دستور ZCARD تعداد کل موجودیت های را برمی گرداند.

```
➢ ZCARD places
(integer) 4
```

Geospatial

با استفاده از دستور ZRANGE می توان لیستی از مقادیر را برگرداند.

➢ ZRANGE places 0 -1
1) "San Francisco"
2) "Mountain View"
3) "Palo Alto"
4) "San Jose"

How to perform spatial search?

To perform a spatial search of entities GEORADIUS or GEORADIUSBYMEMBER can be used in Redis 3.2.0 (and above) and GEOSEARCH or GEOSEARCHSTORE can be used in Redis 6.2 (and above).

Geospatial

پیچیدگی زمانی این دستورها تقریبا O(N+log(M)) است که در آن N تعداد عناصر داخل جعبه مرزی ناحیه دایره ای است که با مرکز و شعاع مشخص شده اند و M تعداد آیتم های داخل index است.

برای یافتن همه مکان های در شعاع 20 مایلی Mountain View به همراه فاصله آنها ، دستور زیر اجرا می کنیم :

```
➢ GEORADIUSBYMEMBER places "Mountain View" 20 mi WITHDIST
1) 1) "Mountain View"
   2) "0.0000"
2) 1) "Palo Alto"
   2) "5.0297"
3) 1) "San Jose"
   2) "11.3186"
```

Geospatial

حال فرض کنید می خواهیم نزدیک ترین مکان به شهر ردوود را با مختصات ۳۷.۴۸۴ درجه شمالی، ۱۲۲.۲۲۸ درجه غربی پیدا کنیم، می توانیم دستور GEORADIUSرا با ASC(برای صعودی)و COUNTرا به صورت ۱ (برای محدود کردن به یک نتیجه) اجرا می کنیم:

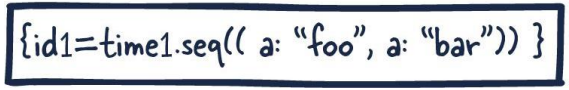
- GEORADIUS places -122.228 37.484 50 mi COUNT 1 ASC WITHCOORD WITHDIST

```
1) 1) "Palo Alto"
   2) "5.5294"
   3) 1) "-122.14300006628036499"
      2) "37.4410011922460555"
```

نکاتی که باید به خاطر بسپارید:

Geospatial

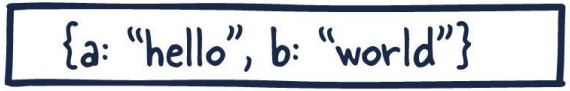
- Redis uses Haversine formula to calculate the distances as the model assumes that the Earth is a sphere. This can result in an error of up to 0.5%.
- Valid longitudes range from -180 to 180 degrees, while valid latitudes range from -85.05112878 to 85.05112878 degrees.
- The time complexities of the GEOADD and GEORADIUS commands are in the order of O(log(N)) and O(N + log(M)) respectively. It is important to keep the key size small and the data partitioned to avoid overloading any single node. Increases in radius can cause the command execution time to increase.



Stream

Geospatial

Bitmap



Hash

Bitfield

List

Set

String

Hyperlog

Sorted set

Redis چگونه اخراج داده ها را مدیریت می کند؟

Redis از مکانیسم های مختلفی برای حذف داده ها پشتیبانی می کند مانند LRU که اخیرا استفاده شده است و TTL مقدار انقضا

حداکثر تعداد کلید های که می توان در Redis ذخیره کرد به مقدار حافظه موجود در سیستم محدود می شود.

Thanks!

Any questions?

You can find me at:

- @eskazemi
- m.esmaeilkazemi@gmail.com