

Hello!



I am Esmaeil Kazemi

I'm interested in learning how are you?

You can find me at @eskazemi





NOSQL

vs

SQL

Redis

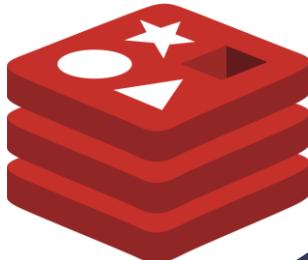
Redis stands for Remote Dictionary Server





eskazemi

Map Redis



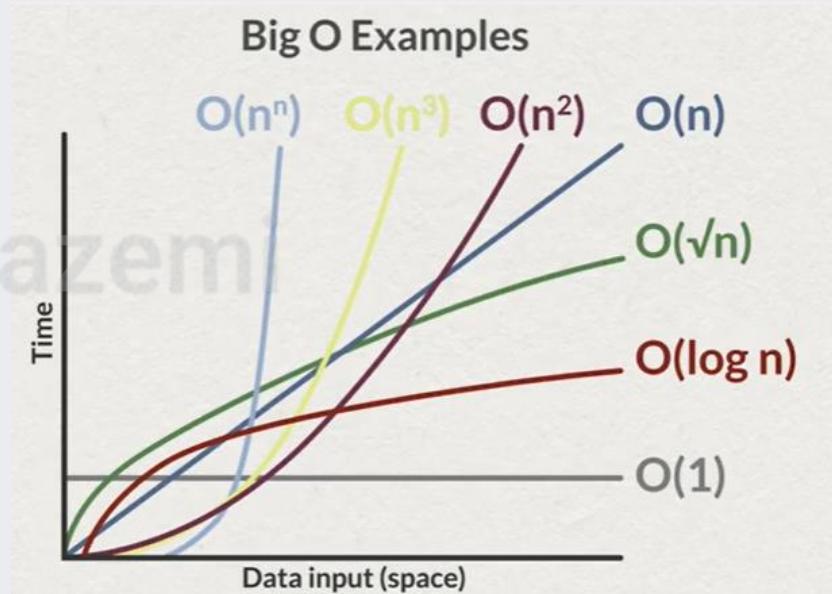
- 1- Features
- 2- application
- 3- data type
- 4- Message Queue
- 5- Transactions
- 6- Pipelining
- 7- Lua Scripts
- 8- Persistence
- 9- Benchmarks
- 10- configuration
- 11- ACLs
- 12- Redis Cluster
- 13- Redis vs Memcached
- 14- Redis vs Hazelcast
- 15- Redis vs RDBMS



eskazemi

Big O:

Describes the limiting behavior of a function
Created by Paul Bachmann
And Edmund Landau
From the German word:
“Ordung” = “order of”





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 **Redis Cluster** از طریق **partitioning** 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

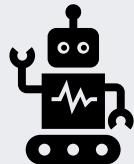


Redis به سرعت مشهور است چون توانایی اجرای query 100000 در ثانیه را دارد. سرعت خود را با حفظ داده ها در حافظه و همچنین ذخیره آنها بر روی دیسک به دست می آورد.

انواع زبان های مختلف چون python , c , c++, Ruby, java ... 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

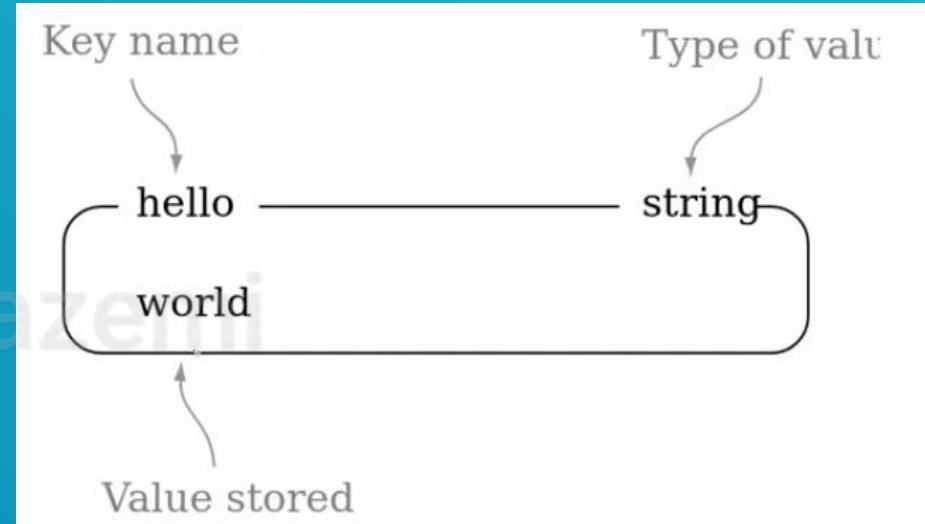


توصیه های مختلف به هنگام استفاده از Redis

یکی از ایرادات این نوع پایگاه داده ها عدم وجود **ویژگی سازگاری** در آنها است. اگر داده هایی که می خواهیم ذخیره کنیم دارای ارتباطات متفاوتی باشند، پایگاه های داده کلید-مقدار عملکرد خود را از دست می دهند. 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





commands

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

برای چک کردن آیا یک کلید وجود دارد یا نه؟

مقدار زمان عمر یک کلید بر می گرداند

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

برگداشتن نوع ساختار داده یک کلید

SET

GET

DEL

EXISTS

KEYS

TTL

EXPIRE

PERSIST

RENAME

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

برای حذف یک کلید

برگرداندن کلید ها براساس یک الگو (regex) (استفاده از TTL)

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.

تغییر نام یک کلید

TYPE



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 (0xff/45/3.1415/"foo")، مانند 0xff/45/3.1415/"foo" 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 عموماً 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

KEYS

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

“

SCAN cursor [MATCH pattern] [COUNT count]

آرگومان مکان نما در فراخوانی بعدی

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

”

KEYS

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 برmi گرداند که کاربر باید از آن به عنوان آرگومان مکان نما در فراخوانی بعدی استفاده کند.

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

```
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- به طور پیش فرض دستور SCAN ده کلید را بر mi گرداند.

KEYS

Commands DEL Vs UNLINK

- **DEL key [key ...]**
- **UNLINK key [key ...]**

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

KEYS

Time Complexity:

- **DEL key [key ...]**

- O(1) when removing a String data type

- O(N) when multiple keys are removed

- O(M) when the key removed contains a List, Set, Sorted Set or Hash.

KEYS

Encoding verified before command execution

دستور Redis OBJECT برای بررسی داخلی های Redis Objects مربوط به کلیدها(keys) 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:

1. **OBJECT REFCOUNT <key>** returns the number of references of the value associated with the specified key. This command is mainly useful for **debugging**.
2. **OBJECT ENCODING <key>** returns the **kind of internal representation** used in order to store the value associated with a key.

KEYS

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

”

نکته بسیار مهم:

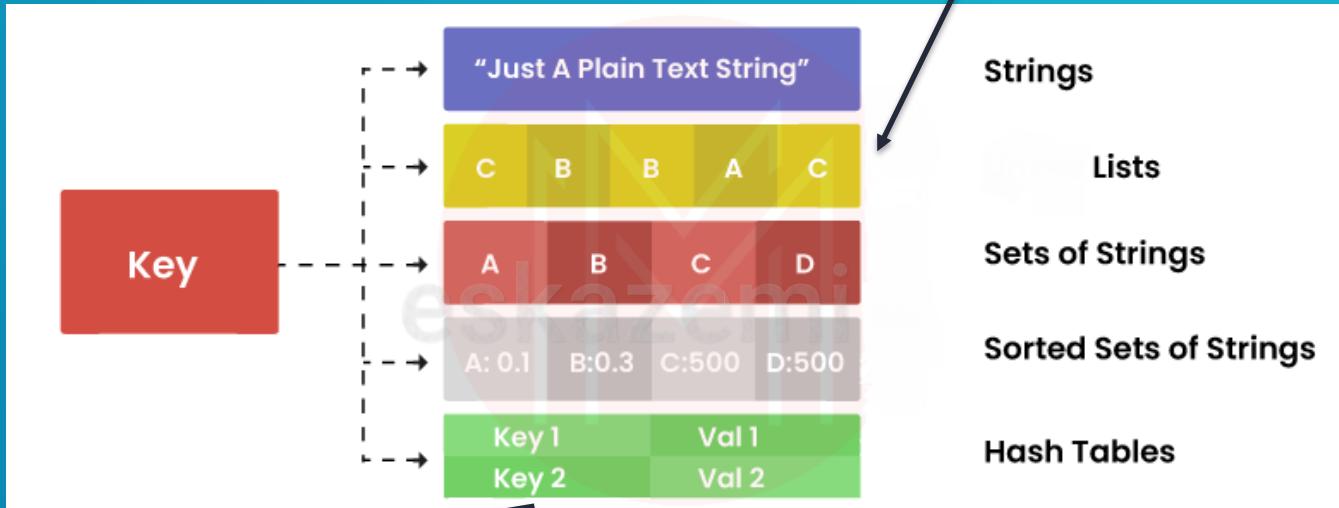
Redis به سادگی فقط کلید را مدیریت می کند . هیچ مفهوم ضمنی یا صریحی در مورد رابطه یا وابستگی بین کلید ها و جود ندارد در صورت وجود مسئولیت مدیریت بر عهده توسعه دهنده و ادمین است.

KEYS

”

value

می تواند یه لیست باشد.



می تواند key-value باشد.



Data type

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



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، JSON سریال 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.
- تمام مقادیر به صورت default Redis می بینه .

“

چند کاربرد عملی برای استفاده از string

as there is built-in support
for incrementing and
decrementing integer and
floating point values.



Caching (The most common use)

API responses

session storage

HTML pages

great for implementing counters

”

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

string

محتویات string را می توان به متن، عدد، باینری در هر نقطه تغییر داد. برای Redis ، همیشه یک نوع داده رشته ای است و 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. ✓

Encoding of value و Datatype برای Redis از دو معیار 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 بکنند و 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 Redis نوع 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 دو طرفه (doubly linked list) پیاده سازی می کند.

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	LLLEN 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

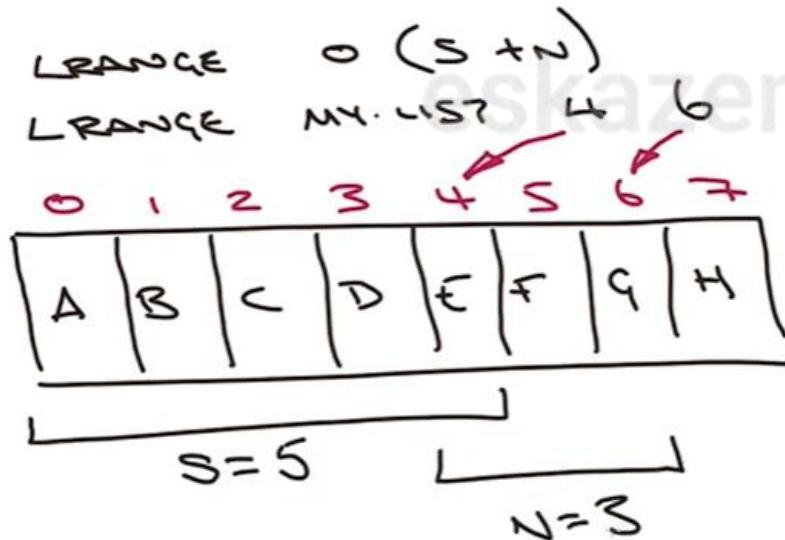
- **LRANGE key start stop**

Time complexity:

$O(S+N)$ where S is the distance of start offset from HEAD for small lists, from nearest end (HEAD or TAIL) for large lists.

N is the number of elements in the specified range

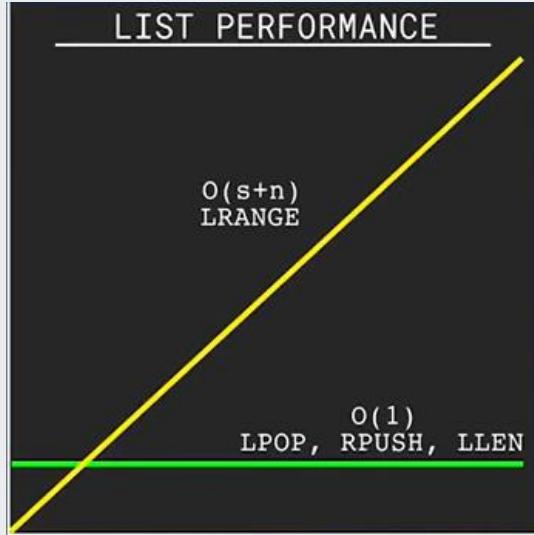
List commands



With a List of 1000 elements,
what is the time complexity for
the following command?

List commands

دستورات مهم توضیحات بیشتر :



Time Complexity

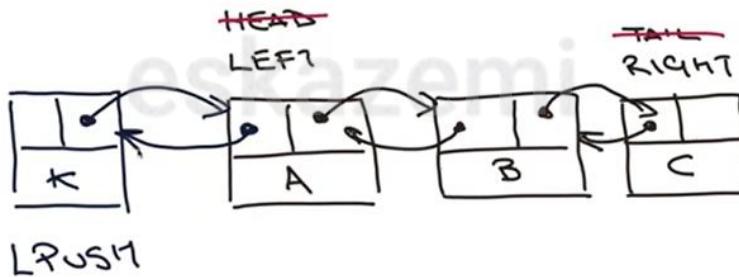
LPUSH RPUSH

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 اهمیت این است که زمان رو به عنوان ورودی میگیرد که اگر زمان صفر است 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 است شود بعد از تایم مشخص شده غیر فعال می شود(بی خیال حذف می شود)

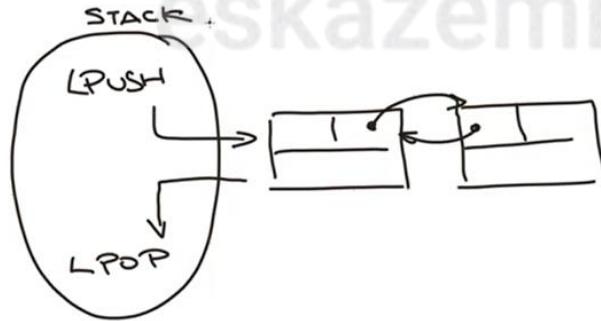
doubly linked list



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.

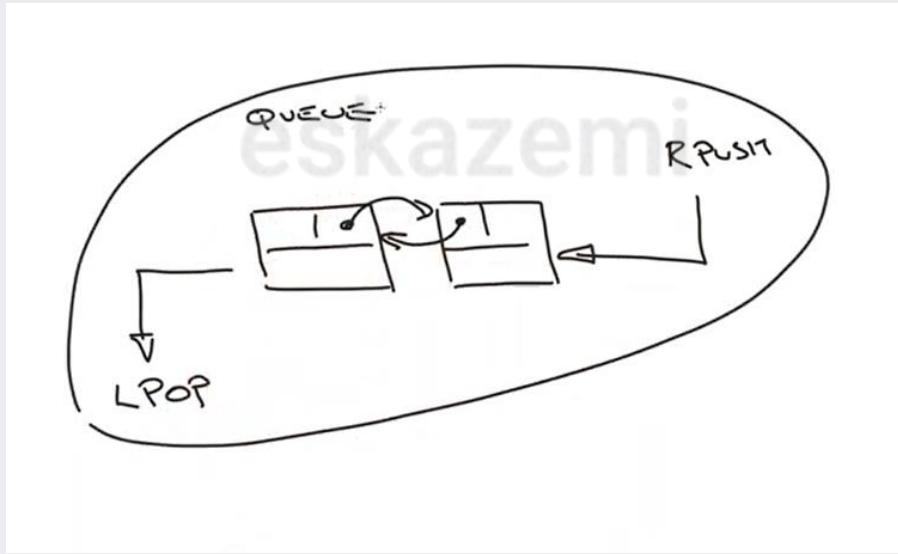
Stack

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

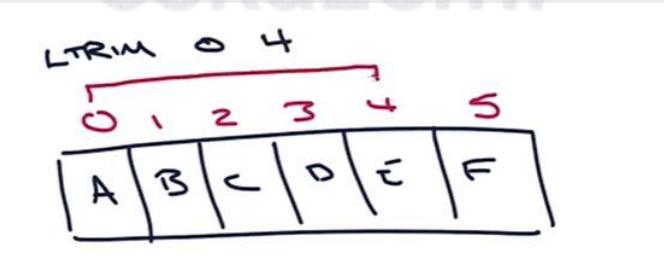
If new elements are added to the **right of the list** and removed from the **left of the list** we've implemented a **Queue**.



“

- LTRIM key start stop

"LTRIM" allows you to specify the range of elements you want to retain. Trimming can be specified from the left with a positive index, or from the right with a negative index.



”

Use Cases -1

Use Cases:

- Activity Stream
 - lpush stream one two three four five
 - lrange stream 0 2
 - ltrim stream 0 3

- زمانی که نیاز به دانستن چدیدترین 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 ها می توان از LRANGE 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. 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 -2

Use Cases:

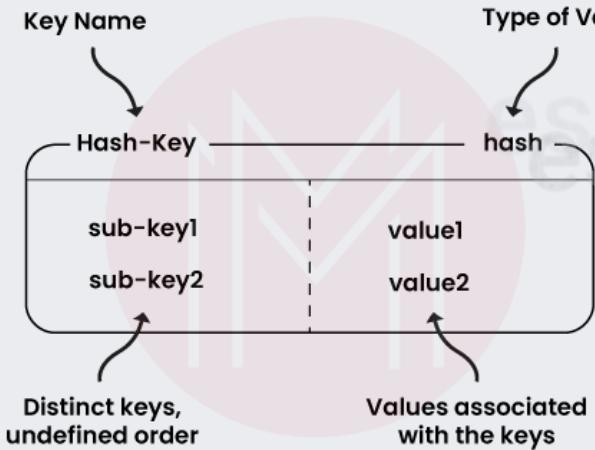
eskazemi

- Inter process communication
 - Produce
 - rpush queue “event1”
 - Consume
 - lpop queue





Hash



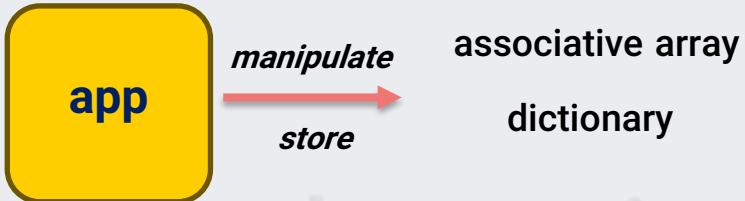
نوعی از داده هستند که می توانید در یک کلید چندین مقدار به شکل کلید/مقدار ذخیره کنید.

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 Hashes 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.

hashes and how they can be utilized



چگونه باید این ساختار را در Redis ذخیره کنید؟

ساده ترین راه برای ذخیره و دسترسی به داده می توانید شی مرتب سازی شده را در Redis به صورت یک binary یا storing ذخیره کنید و سپس زمانی که آن را از Redis می خواهیم، ساختار داده را مرتب سازی کنید. اما چه اتفاقی می افتد اگر تنها به یک بخش از آن ساختار دسترسی داشته باشید یا یک مقدار را در آن ساختار افزایش دهید؟ شما باید کل ساختار را از پایگاه داده بخوانید، و به ساختار خاص زبان خود تبدیل کنید. ساختار پس از تغییر اندازه ساختار و یا انجام افزایش باید شی مورد نظر شما serializer شود، قبل از اینکه در پایگاه داده تغییر کند(ذخیره شود)، Redis کل رشته را ذخیره می کند نه فقط تفاوت ها.

یک روش کارآمدتر برای ذخیره سازی جداول بدون ناکارآمدی مرتب سازی و از بین بردن کل شی است. از آنجا که هر فیلد ذخیره شده و در دسترس است، تغییر در یک فیلد تنها نیازمند این است که داده های آن فیلد برای Redis ارسال شود. نیازی به خواندن و نوشتن کل شی نیست

”

بنابراین:

Schema-less Hash ها هستند به این معنی که شما می توانید هر زمان که نیاز دارید فیلد ها رو کم و زیاد کنید. ✓

Hash ها اجازه می دهند که فیلد های فردی دستکاری شوند زمانی که نیاز به تغییر فقط یک زیر مجموعه از فیلد ها است این باعث صرفه جویی در CPU ، شبکه و سایر منابع می شود . با این حال **نمی توانید** فیلد های درون یک هش را منقضی کنید انقضا فقط روی کلید تنظیم می شود . ✓

hash

”

Hash

به طور خلاصه بخواهیم بگوییم مزایای ذخیره سازی یک شی در **hash** در مقابل ذخیره سازی در یک **string** چیست؟

1. Individual fields can be Get and Set
2. Individual fields can be Incremented
3. Individual fields can be tested for Existence

Hash

روش های مختلفی برای ذخیره یک شی با روابط و hierarchies در Redis وجود دارد . استفاده از هر کدام از تکنیک ها بستگی دارد به به موارد استفاده و عملیاتی که باید انجام دهید.

- ✓ Serialize the object into a String datatype
- ✓ Flatten the relationships and hierarchy into fields in a Hash
 - ✓ Normalize the structure into multiple Hashes
 - ✓ Normalize the structure into multiple Hashes and Sets



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

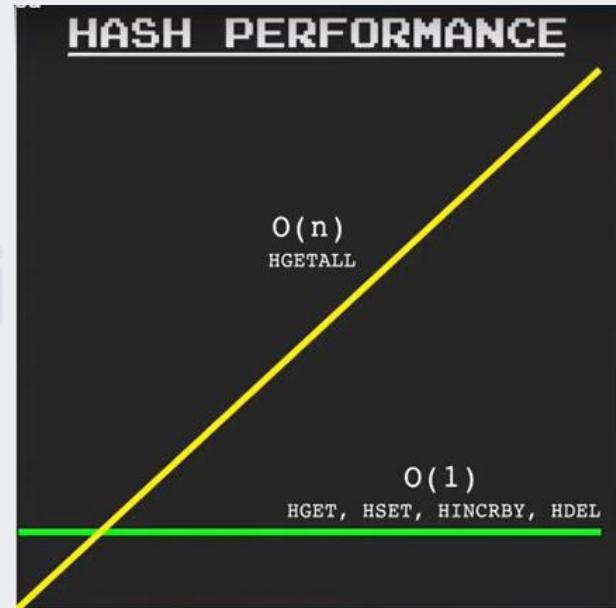
Most Hash commands are $O(1)$, which means they will always perform a task in a constant amount of time, regardless of the size of the Hash.

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

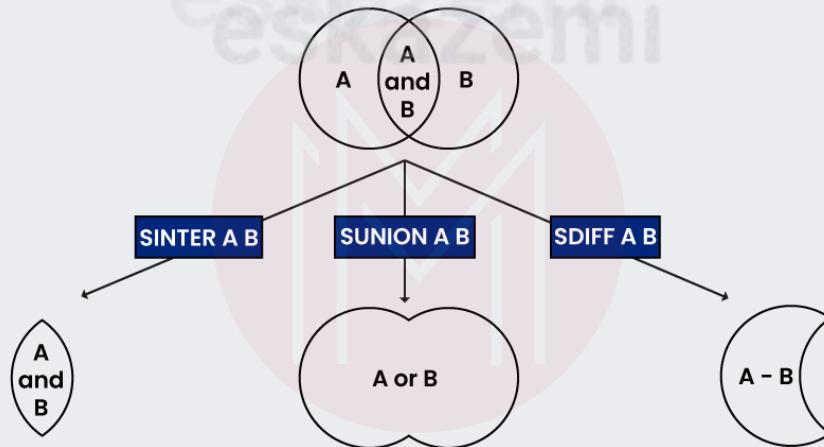
- مجموعه ای از المان هایی String است که به شکل نامنظم در یک کلید ذخیره می شوند که هیچ تکراری ندارند.
- هر set توانایی ذخیره بیشتر از 4 میلیارد المان در خود را دارند.
- از عملیات های استاندارد مجموعه ریاضی مانند اشتراک، اجتماع و تفاضل پشتیبانی می کند.



SINTER و S UNION به چه معناست؟

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

S INTER 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	SDIFF key1 [key2]	Subtracts multiple sets
4	SDIFFSTORE destination key1 [key2]	Subtracts multiple sets and stores the resulting set in a key
5	SINTER key1 [key2]	Intersects multiple sets
6	SINTERSTORE destination key1 [key2]	Intersects multiple sets and stores the resulting set in a key

Set commands

	دستور	توضیح
7	SISMEMBER key member	Determines if a given value is a member of a set
8	SMEMBERS key 1, if the element is a member of the set. 0, if the element is not a member of the set, or if the key does not exist.	Gets all the members in a set
9	SMOVE source destination member	Moves a member from one set to another
10	SPOP key	Removes and returns a random member from a set
11	SREM key member1 [member2]	Removes one or more members from a set
12	SRANDMEMBER key [count]	Gets one or multiple random members from a set

Set commands

SDIFF یک تفريق منطقی از اولين مجموعه با مجموعه های بعدی ارائه شده انجام می دهد. مجموعه های حاوي مجموعه ای از مقادير بى نظم و منحصر به فرد هستند. Redis یک مقایسه بايت را انجام می دهد، بنابراین "A" با "a" و غيره يکی نیست.

Syntax:

```
> sadd set-three A b C  
> sadd set-four a b C  
> sdiff set-three set-four
```

Set commands

Time complexity:

- SINTER key [key ...]

O(N*M) worst case where **N** is the cardinality of the smallest set
and **M** is the number of sets

Set commands

SINTER $O(N \times M)$
SET-SMALL 5 } $N=5$
SET-LARGE 16
M=2

M=2 is numbers set

So as the Cardinality of the smallest set increases or the number of attributes being matched (i.e. the number of Sets to be examined) increases, then the cost of this method increases.

Set commands

- **SADD key member [member ...]**

Time complexity:

O(1) for each element added So O(1) to add N elements when the command is called with multiple arguments.

Index/set

Redis

no native support

Secondary Indexes

how to achieve these constructions with Redis Sets.



Index/set

- Secondary Indexes
- Faceted Search
 - SINTER
 - Data & Cardinality
- Hashed Search

Set ها ساختاری به طور کلی مفید هستند که امکان ذخیره و دستکاری مجموعه های بی نظم و منحصر به فرد را فراهم می کند. ما گفتیم Redis از مفهوم شاخص های ثانویه پشتیبانی نمی کند. این نوع شاخص اغلب برای اجازه دادن به پرس و جوهای کارآمدتر از ویژگی های domain object استفاده می شود.

با این حال، نگهداری شاخص های ثانویه با افزایش CPU، حافظه و پیامدهای ذخیره سازی همراه خواهد بود. در Redis، ما می توانیم از ویژگی هایی که می توانیم آنها را به روشی کارآمد از نظر منابع و راه های جایگزین برای دسترسی به این داده ها حفظ کنیم، بهره برداری کنیم. ما تعدادی مجموعه برای ذخیره مقادیری که می خواستیم بر اساس آن جستجو کنیم، می سازیم. اینها برای پیاده سازی یک **faceted search**، یعنی جستجو با طبقه بندی های مختلف مورد استفاده قرار گرفتند. با انجام یک تقاطع با می توانیم تطابق بین این مقادیر نمایه شده (indexed values) را پیدا کنیم. توزیع داده ها را که بر عملکرد این دستور تأثیر می گذارد و معیارهایی ارائه خواهیم کردیم که باید هنگام انتخاب ساختار داده در نظر بگیرید. در نهایت از هش کردن برای ایجاد یک مقدار ترکیبی برای جستجو، بهینه سازی بیشتر پرس و جو ما (Hashed Search).

Use Cases

Use Cases:

- Tag Cloud
 - SADD wrench tool metal
 - SADD coin currency metal
 - SSCAN wrench match*
 - SINTER wrench coin
 - SUNION wrench coin

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

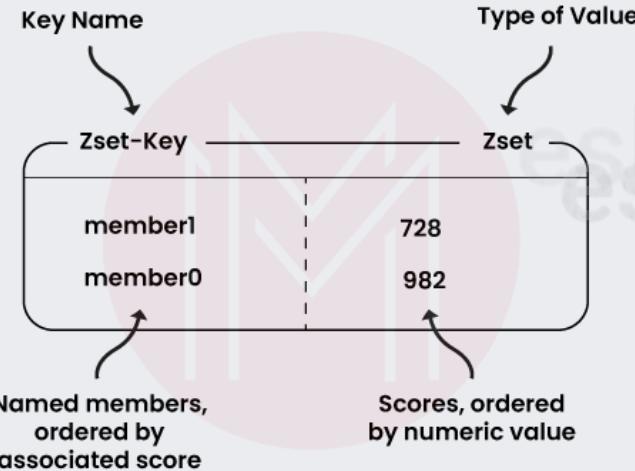
Use Cases:

- Unique visitors
 - URL + Time period
 - SADD about.html:20180210
jim jane john
 - SSCAN about.html:20180210
match *
 - EXPIRE about.html:20180210
86400

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 می‌توان مجموعه‌ای ایجاد کرد. به عنوان مثال "about.html" و 10 فوریه 2018. کوکی هر بازدید کننده را می‌توان 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 می‌توان برای تعریف دوره نگهداری برای متريک، در اين مورد، 86400 ثانие يا يك روز استفاده کرد. از آنجايي که انقضا به کلید مربوط مي شود نه عناصر موجود در مجموعه، پس از انقضاي کلید همه عناصر حذف مي شوند.



ZSET



این نوع داده دقیقاً شبیه set هاست و نمی‌توان المان‌های تکراری در آن ذخیره کرد. تنها تفاوتی که این دو باهم دارند این است که المان‌ها score یا امتیاز دارند

(از نوع float می‌باشد) که Redis المان‌ها را به ترتیب این score‌ها ذخیره می‌کند. در این نوع داده المان‌ها نمی‌توانند موارد تکراری داشته باشند اما امتیاز‌ها می‌توانند تکراری شوند.

اگر دو المان با امتیاز یکسان داشته باشید، تساوی با ترتیب واژگانی المان‌ها شکسته می‌شود. مجموعه‌های مرتب را می‌توان 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 ZSET‌ها اعمال خواندن، اضافه کردن، حذف کردن و آبدیت کردن المان‌ها بسیار سریع است.



Key	Value
h	
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

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

Sorted Set commands

	دستور	توضیح	نکته
1	ZADD key score1 member1 [score2 member2]	Adds one or more members to a sorted set, or updates its score, if it already exists	It is possible to specify multiple score/member pairs. If a specified member is already a member of the sorted set, the score is updated and the element is reinserted at the right position to ensure correct ordering
2	ZCARD key	Gets the number of members in a sorted set	

Sorted Set commands

	دستور	توضیح	نکته
3	ZCOUNT key min max	returns the number of elements in the sorted set at the key with a score between min and max.	
4	ZINCRBY key increment member	Increments the score of a member in a sorted set	If the key does not exist, a new sorted set with the specified member as its sole member is created. An error is returned when the key exists but does not hold a sorted set.
5	ZRANGE key start stop [WITHSCORES]	Returns a range of members in a sorted set, by index	

Sorted Set commands

	دستور	توضیح	نکته
6	ZRANGEBYLEX key min max [LIMIT offset count]	Returns a range of members in a sorted set, by lexicographical range	گستره مشخصی از عناصر در مجموعه مرتب شده ذخیره شده در کلید را برمی گرداند. این عناصر از پایین ترین تا بالاترین امتیاز مرتب شده اند. ترتیب Lexicographical برای عناصر با امتیاز برابر استفاده می شود. هم شروع و هم توقف شاخص های مبتنی بر صفر هستند، که در آن ° عنصر اول، 1 عنصر بعدی وغیره است. آن ها همچنین می توانند اعداد منفی باشند که Offset های انتهایی مجموعه مرتب شده را نشان می دهند، که 1- آخرین عنصر مجموعه مرتب شده است، 2- عنصر ماقبل آخر است، وغیره.
7	ZRANGEBYSCORE key min max [WITHSCORES] [LIMIT]	Returns a range of members in a sorted set, by score	تمام عناصر موجود در مجموعه مرتب شده را با امتیاز بین min و max (شامل عناصری با امتیاز برابر max یا min) بر می گرداند. این عناصر از پایین به بالا مرتب شده اند.

Sorted Set commands

	دستور	توضیح	نکته
8	ZRANK key member	Determines the index of a member in a sorted set	رتبه (یا شاخص) بر پایه ۰ است، به این معنی که عضوی که کم ترین امتیاز را دارد دارای رتبه ۰ است.
9	ZREM key member [member ...]	Removes one or more members from a sorted set	
10	ZREMRANGEBYRANK key start stop	Removes all members in a sorted set within the given indexes	removes all elements in the sorted set stored at the key with the rank between start and stop. Both start and stop are 0-based indexes with 0 being the element with the lowest score

Sorted Set commands

- ZREMRANGEBYRANK key start stop

معادل LTRIM برای Sorted sets می باشد.

This command uses **zero based indexes** for start and stop, but unlike "LTRIM" where you specify the elements to retain, "ZREMRANGEBYRANK" removes the range specified. Both start and stop are 0-based indexes with 0 being the element with the lowest score. These indexes can be negative numbers, where they indicate offsets starting at the element with the highest score.

Sorted Set commands

- ZINTERSTORE

- destination

- numkeys

- key [key ...]

- WEIGHTS weight [weight ...]

- AGGREGATE SUM | MIN | MAX

هنگام انجام یک عملیات set مانند تقاطع(intersection) باید امتیاز حاصله را برای عناصر تطبیقی تعیین کنید زیرا هر مجموعه مرتب شده ممکن است امتیاز متفاوتی در ارتباط با عناصر تطبیقی داشته باشد.

آرگومان Destination کلیدی است که در آن مجموعه مرتب شده حاصل ذخیره خواهد شد. Numkeys تعداد کلیدهایی است که در ادامه آمده است.

وزن ها برای ارایه یک فاکتور ضرب برای هر مجموعه ورودی استفاده می شود جایی است که می توانید نحوه محاسبه Aggregate امتیاز جدید را مشخص کنید.

Sorted Set commands

- **ZUNIONSTORE**
 - **destination**
 - **numkeys**
 - **key [key ...]**
 - **WEIGHTS weight [weight ...]**
 - **AGGREGATE SUM | MIN | MAX**

اتحاد (union) مجموعه های مرتب شده numkeys که توسط کلیدهای مشخص شده داده شده اند را محاسبه می کند و نتیجه را در destination ذخیره می کند. ارائه تعداد کلیدهای ورودی (کلیدهای num) قبل از پاس دادن کلیدهای ورودی و آرگومان های دیگر (اختیاری) **الزامی** است.

به طور خلاصه:

- 
- 1. Ordered collection of unique strings
 - 2. Floating point Score
 - 3. Manipulation by value, position, score or lexicographically
 - 4. Set commands Sorted Sets:
 - Union
 - Intersection
 - 1. Typical Use Cases:
 - Priority Queue
 - Leaderboards

Sorted Set

A Sorted Set is used to store geospatial objects. For each Longitude / Latitude pair, a Geohash is computed and stored as the score for the element in the Sorted Set.

ZINTERSTORE will provide the intersection of elements in source keys and store the results in the destination key.

```
> zinterstore foo 2 geo:events:Football geo:events:Athletics aggregate min
```

The **aggregate min** is required, since by default ZINTERSTORE will sum the score together. Using **aggregate min or aggregate max** will preserve the score and therefore the Geospatial point.

“

Can you create a Union or
Intersection between a Sorted Set
and Set?

Yes

”



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 Unique دارد یا اگر شما نیاز باشید تعداد ایمیل های unique 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



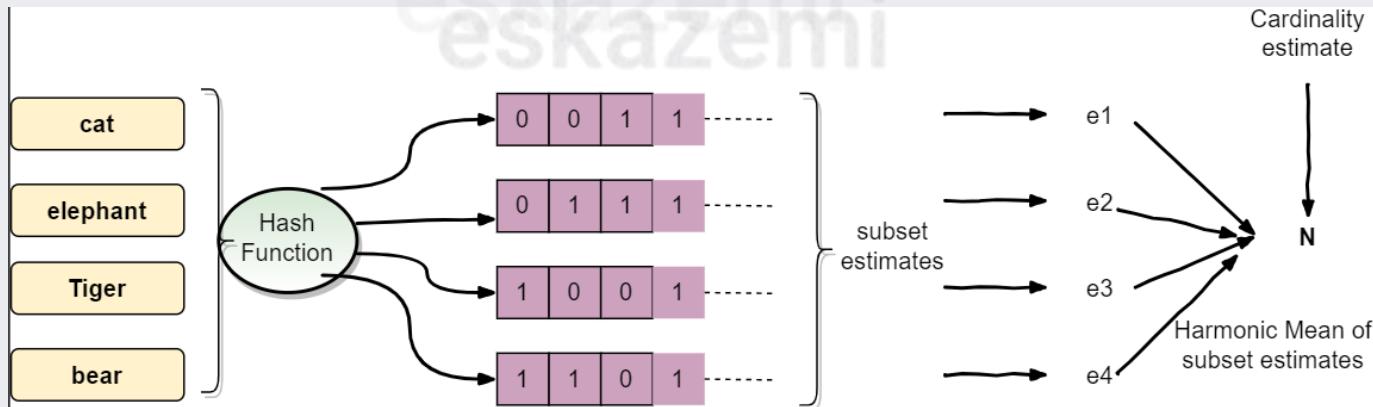
خاطر که در ابتدای دستورات مربوط به این نوع داده از PF استفاده می کنیم.)

بسیار بهینه تر از set ها هستند. استفاده از منابع نسبت به set ها کمتر .

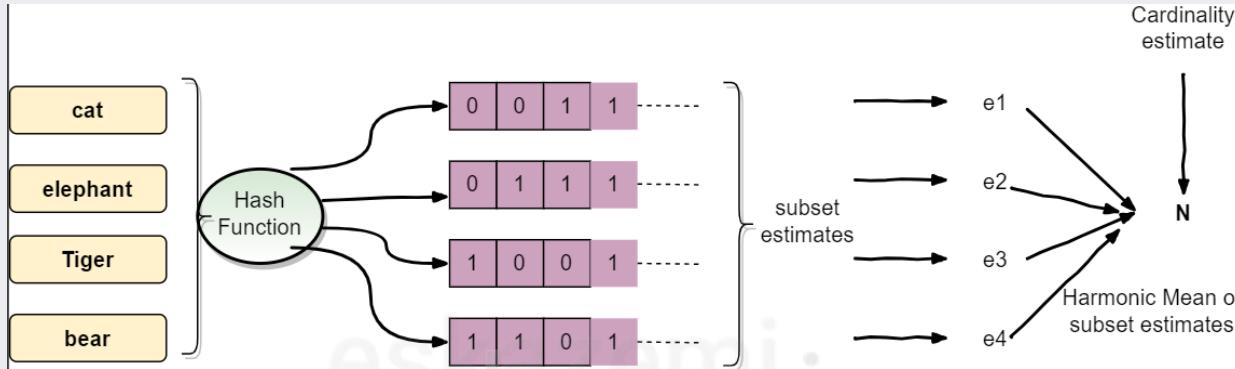


Algorithm Hyporloglog

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 حداقل تعداد صفرهای پیش رو 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 ازتابع 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 حداقل صفرهای پیش رو، در مقادیر آن، در حافظه ذخیره می شوند.

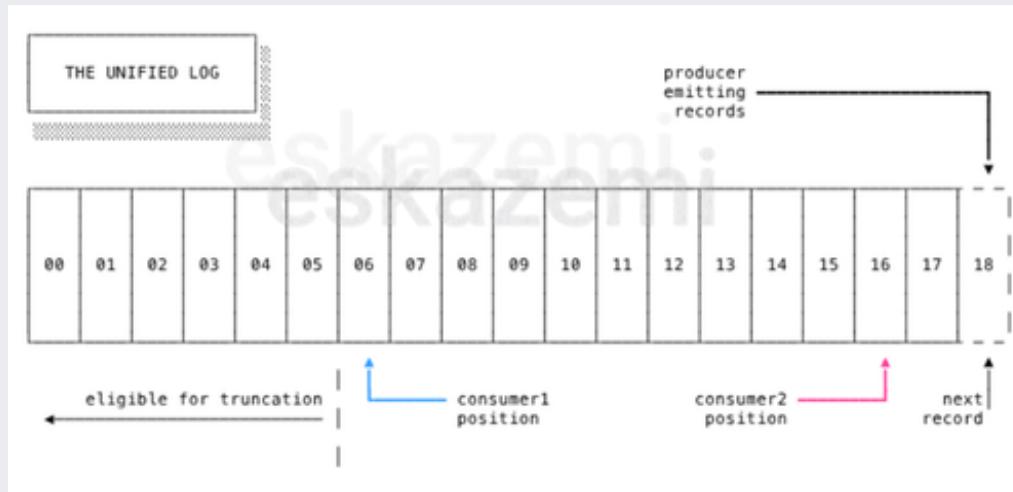
ما میانگین هارمونیک برآوردها را برای تمام زیرمجموعه های محاسبه شده قبلی محاسبه می کنیم.
میانگین هارمونیک محاسبه شده در بالا تخمینی از تعداد بازدید کننده منحصر به فرد در صفحه وب است.

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 صفحه 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.



Bitfield

روشی فشرده و کارآمد برای پیاده سازی شمارنده های چندگانه در یک آرایه واحد ارائه می دهد. 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

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) دارد اما برخلاف دستورهای دیگر، این دستورها قادر نبودند 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 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"

eskazemi

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 ردود را با مختصات ۳۷.۴۸۴ درجه شمالی، ۱۲۲.۲۲۸ درجه غربی پیدا کنیم، می توانیم دستور GEORADIUS را با ASC(برای صعودی) و COUNT را 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**.

011011010110111101101101

Bitmap

eskazemi

{23334}{6634728}{916}

Bitfield

Bit /Byte

بیت (مخفف "رقم دودویی") کوچکترین واحد اندازه گیری است که برای تعیین کمیت کامپیوتر استفاده می‌شود. به عبارتی کوچکترین واحد حافظه بیت (Bit) است و حاوی یک مقدار دودویی (0 یا 1) است.

در حالی که یک بیت می‌تواند مقدار Boolean درست (1) یا نادرست (0) را تعریف کند، یک بیت فردی کاربرد کمی دارد. بنابراین، در فضای ذخیره سازی کامپیوتر، بیت‌ها اغلب با هم در خوش‌های 8 بیتی به نام Byte گروه بندی می‌شوند. از آنجایی که یک بایت حاوی هشت بیت است که هر کدام دارای دو مقدار ممکن است، یک بایت ممکن است 2^8 یا 256 مقدار متفاوت داشته باشد.

بیت‌ها با حروف کوچک "b" خلاصه می‌شوند، در حالی که بایت‌ها با "B" بزرگ مخفف می‌شوند. مهم است که این دو عبارت را اشتباه نگیرید، زیرا هر اندازه گیری در بایت حاوی هشت برابر بیت است. برای مثال، یک متن کوچک (File) با اندازه 4 کیلوبایت حاوی 4000 بایت یا 32000 بیت است.

Bit /Byte

value 8 4 2 1

1 0 1 0

index 3 2 1 0

Decimal

$$8 + 2 = 10$$

Binary	Decimal	Binary	Decimal
0000	0	0001	1
0010	2	0011	3
0100	4	0101	5
0110	6	0111	7
1000	8	1001	9
1010	10	1011	11
1100	12	1101	13
1110	14	1111	15

به طور کلی، فایل‌ها، دستگاه ذخیره سازی (Storage Device) و گنجایش فضای ذخیره (Data transfer) در بایت اندازه گیری می‌شود در حالی که نرخ انتقال داده (Data transfer Capacity) در بیت اندازه گیری می‌شود به عنوان مثال، یک درایو SSD ممکن است دارای ظرفیت ذخیره سازی 240 گیگابایت باشد، در حالی که یک دانلود (Download) ممکن است با سرعت 10 مگابیت در ثانیه منتقل شود

Bit /Byte

”

Bit /Byte

Byte

1 **byte** is group of 8 bits

8 bits can make 256 different patterns

One Byte - 256 Patterns

Start with 0, go up, one pattern per number, until run out of
patterns 0, 1, 2, 3, 4, 5, ... 254, 255

with 256 different patterns, we can store a number in
the range 0..255

Really good for storing characters/letters.

Bit /Byte

Byte

In general: add 1 bit, double the number of patterns

- 1 bit -> 2 patterns
- 2 bits -> 4 patterns
- 3 bits -> 8 patterns
- 4 bits -> 16 patterns
- 5 bits -> 32 patterns
- 6 bits -> 64 patterns
- 7 bits -> 128 patterns
- 8 bits -> 256 - one byte

Number of bits	Different Patterns
1	0 1
2	00 01 10 11
3	000 001 010 011 100 101 110 111

Mathematically: n bits yields 2^n patterns (2 to the nth power)

Bit /Byte

Bytes and Characters - ASCII Code

ASCII is an encoding representing each typed character by a number

Each number is stored in one byte (so the number is in 0..255)

A is 65

Example: B is 66

a is 96

space is 32

"Unicode" is an encoding for mandarin, greek, arabic, etc. languages, typically 2-bytes per "character"

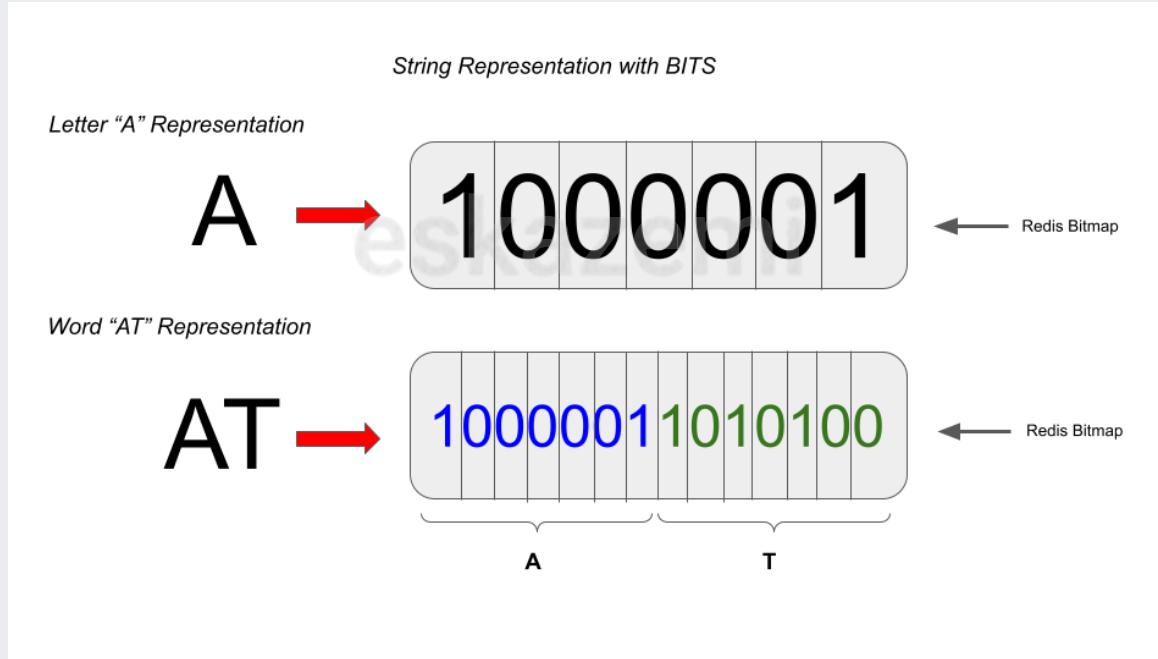
هنگام برخورد با داده های بیتی، دو خانواده از
دستورات وجود دارد:

Bit fields Bit arrays

Redis supports both bit fields and bit arrays

- ✓ Redis تعدادی دستور دارد که اجازه می دهد داده های بیت دستکاری شوند.
- ✓ با این حال یک نوع داده بیت صریح وجود ندارد، این دستورها بر روی نوع داده رشته عمل می کنند.
- ✓ این موضوع شاید در ابتدا عجیب به نظر برسد اما اگر از بخش های قبلی به یاد داشته باشد یک رشته تنها یک مقدار امن باینری است. بنابراین با کمی offset در یک string ، می توانید عملیات بیت را روی مقدار اعمال کنید.

Bitmap (Bit arrays)



Bitmap (Bit arrays)

- ✓ به سادگی آرایه ای از بیت ها است. بنابراین به عنوان آرایه بیت یا بردار بیت شناخته می شود .
- ✓ این یکی از ساختار های داده ای است که Redis را کاملاً انعطاف پذیر و گسترده می کند . اگر نیاز به ذخیره نقشه ای از اطلاعات Boolean در یک فضای فشرده دارید، Bitmap انتخاب پیش فرض شما خواهد بود.
- ✓ از این رو ساختار داده زیر بنایی Bitmap ها رشته ای است که در آن یک رشته با استفاده از آرایه ای از صفر و ویک در حافظه کامپیوتر ذخیره می شود.
- ✓ Redis bitmaps از بازیابی و تغییر مقدار بیت یک offset داده شده در یک bitmap پشتیبانی می کند. علاوه بر این، دستورهای لازم برای انجام عملیات های بیتی مانند **Not**, **OR**, **AND** و **XOR** را بر روی چندین رشته که در کلیدهای داده شده ذخیره شده اند، فراهم می کند.

Bitmap (Bit arrays)

✓ قبل از فرو رفتن در دستورات واقعی و ذخیره بیت مپ ها،
به چند نکته توجه کنید:

یک نوع داده بومی در Redis نیست. در واقع، آنها مجموعه ای از عملیات بیت گرا هستند که بر روی String ساخته شده اند.

عملیات بیت به دو گروه اصلی دسته بندی می شوند:
عملیات تک بیتی با زمان ثابت.

نمونه ای از عملیات تک بیتی تنظیم یک بیت از 1 به 0 یا بازیابی مقدار یک بیت است.

عملیات بیت گروهی

عملیات بیت گروهی می تواند شامل فرآیندی مانند دریافت تعداد بیت ها در یک محدوده خاص باشد.

Bitmap (Bit arrays) commands

- **GETBIT key offset**
- **SETBIT key offset value**
- **BITCOUNT key [start end]**
- **BITOP operation destkey key [key ...]**
- **BITPOS key bit [start] [end]**

Bitmap (Bit arrays)

Create Bitmap

To create a key holding a bitmap in Redis:

SETBIT command

The command takes the name of the key, offset value,
and the actual bit as arguments

The syntax is as shown:

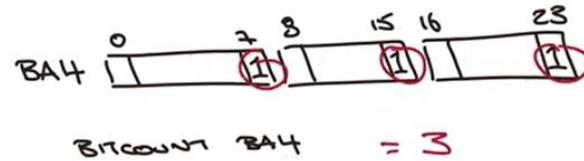
```
127.0.0.1:6379> SETBIT key offset bit
```

The offset value must be greater than or equal to 0 but less than 2^{32} . This is because bitmaps
are limited to 512MB.

Bitmap

Count the number of set bits

BITCOUNT



Bitmap

Count the number of set bits

BITCOUNT

Syntax

```
BITCOUNT key [start end [BYTE | BIT]]
```

مانند دستور GETRANGE شروع و پایان می‌تواند حاوی مقادیر منفی باشد تا bytes را که از انتهای رشته شروع می‌شوند، فهرست کنند، جایی که 1- آخرین بایت، 2- ماقبل آخر و غیره است.

به طور پیش فرض، آرگومان های اضافی شروع و پایان یک شاخص byte را مشخص می کنند. می‌توانیم از یک آرگومان اضافی BIT برای تعیین یک شاخص BIT استفاده کنیم. بنابراین 0 بیت اول، 1 بیت دوم و غیره است. برای مقادیر منفی، -1 بیت آخر، -2 بیت ماقبل آخر است و غیره.

Count Number of Set Bits

Bitmap

To determine the number of set bits in Redis, use the **BITCOUNT** command.

```
127.0.0.1:6379[10]> SETBIT bitkey 3 1
127.0.0.1:6379[10]> SETBIT bitkey 4 1
127.0.0.1:6379[10]> SETBIT bitkey 5 1
127.0.0.1:6379[10]> SETBIT bitkey 6 1
127.0.0.1:6379[10]> SETBIT bitkey 7 1
```

Example:

To get the number of set bits, run:

```
127.0.0.1:6379[10]> BITCOUNT bitkey
(integer) 5
```

NOTE: A set bit refers to any bit whose value is set to 1.

Bitmap

Example:

```
redis> SET mykey "foobar"
"OK"
redis> BITCOUNT mykey
(integer) 26
redis> BITCOUNT mykey 0 0
(integer) 4
redis> BITCOUNT mykey 1 1
(integer) 6
redis> BITCOUNT mykey 1 1 BYTE
(integer) 6
redis> BITCOUNT mykey 5 30 BIT
(integer) 17
```

Bitmap

BITCOUNT

نمونه استفاده از آن:

Bitmaps نمایش بسیار کارآمدی از انواع خاصی از اطلاعات هستند. یک مثال یک برنامه وب است که به تاریخچه بازدیدهای کاربر نیاز دارد، به طوری که برای مثال می‌توان تعیین کرد که چه کاربرانی اهداف خوبی برای ویژگی‌های بتا (beta) هستند.

هر بار که کاربر یک نمای صفحه را می‌بیند، برنامه می‌تواند ثبت کند که در روز جاری کاربر از وب سایت بازدید کرده است و با استفاده از دستور SETBIT بیت مربوط به روز جاری را تنظیم می‌کند.

در مثال بالا از شمارش روز، حتی پس از 10 سال آنلاین بودن برنامه، ما هنوز فقط $10 * 365$ بیت داده به ازای هر کاربر داریم، یعنی فقط 456 بایت برای هر کاربر. با این مقدار داده، BITCOUNT همچنان به سرعت هر دستور دیگر Redis (O(1) مانند INCR یا GET است.

Bitmap

Retrieve Bit Value

برای بدست آوردن مقدار بیت ذخیره شده در یک offset خاص از دستور **GETBIT** و سپس offset هدف استفاده کنید.

```
127.0.0.1:6379[10]> GETBIT bitkey 3
```

Count the number of set bits

(integer) 1

اگر بیت در offset مشخص شده تعیین نشده باشد دستور مقدار صفر را بر می گرداند.

```
127.0.0.1:6379[10]> GETBIT bitkey 200
```

(integer) 0

Bitmap

BITPOS command

get the index of the smallest false/true bit

Integer reply

1. the position of the first bit set to 1 or 0 according to the request
2. -1. In case the bit argument is 1 and the string is empty or composed of just zero bytes
3. Non-existent keys are treated as empty strings.

Bitmap

BITPOS command

به طور پیش فرض، **تمام** بایت های موجود در رشته بررسی می شوند. این امکان وجود دارد که تنها در یک بازه مشخص با گذر از آرگومان های اضافی شروع و پایان به دنبال بیت ها بگردیم (این امکان وجود دارد که فقط شروع را پشت سر بگذاریم، در این عملیات فرض بر این خواهد بود که پایان آخرین Byte رشته است. به طور پیش فرض، دامنه به عنوان محدوده ای از بایت ها تفسیر می شود و نه محدوده ای از بیت ها، بنابراین شروع ° (start=0) و پایان ² (end=2) به معنی نگاه کردن به سه بایت اول است.

شما می توانید از آپشن اختیاری BIT استفاده کنید تا مشخص شود که محدوده باید به عنوان محدوده ای از بیت ها تفسیر شود. پس شروع ° و پایان ² به معنای نگاه کردن به سه بیت اول است.

مانند دستور GETRANGE شروع و پایان می تواند حاوی مقادیر منفی به منظور نمایه کردن بایت هایی باشد که از انتهای رشته شروع می شوند، که در آن ۱- آخرین بایت، ۲- ماقبل آخر . وقتی BIT مشخص می شود، ۱- آخرین بیت، ۲- ماقبل آخر.

Bitmap

BITPOS command

BITPOS key bit [end [BYTE | BIT]]

0	1	0	0	0	0	0	1
---	---	---	---	---	---	---	---

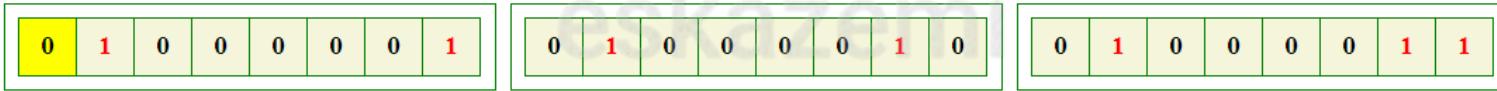
0	1	0	0	0	0	1	0
---	---	---	---	---	---	---	---

0	1	0	0	0	0	1	1
---	---	---	---	---	---	---	---

Bitmap

BITPOS command

BITPOS key 0



Command>BITPOS key 0
Result>0

Bitmap

BITPOS command

BITPOS key 1 2

0	1	0	0	0	0	0	1
---	---	---	---	---	---	---	---

0	1	0	0	0	0	0	1	0
---	---	---	---	---	---	---	---	---

0	1	0	0	0	0	0	1	1
---	---	---	---	---	---	---	---	---

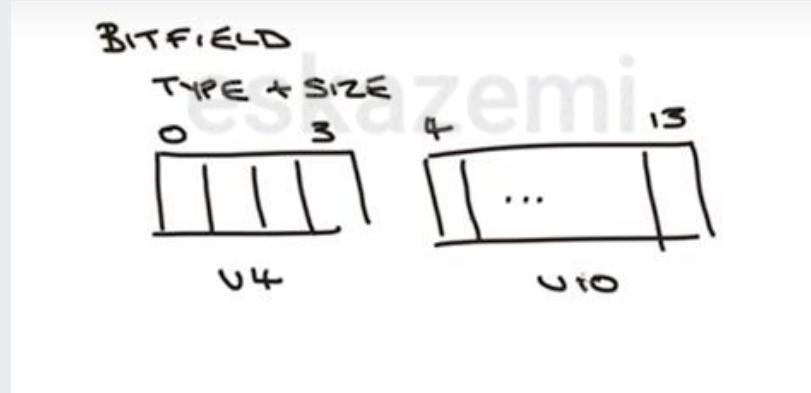
Command>BITPOS key 1 2
Result>17

bit byte

Bitmap

BITFIELD command

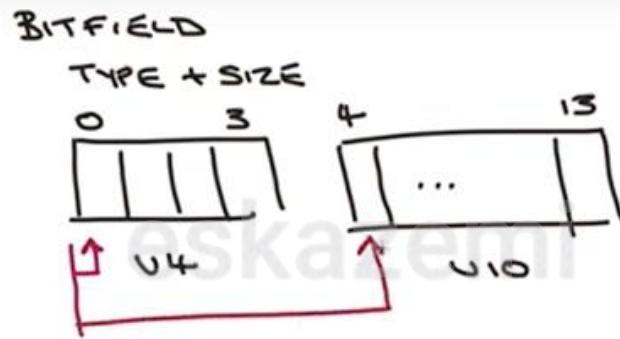
این دستور امکان دستکاری یک یا چند اعداد صحیح با طول متغیر را در نوع داده رشته ای فراهم می کند



در اینجا دو فیلد وجود دارد که هر کدام یک نوع داده و اندازه دارند. اولی یک "u4" یک عدد صحیح بدون علامت با اندازه چهار بیت است. دومی یک "u10" یک عدد صحیح بدون علامت با اندازه ده بیت است.

Bitmap

BITFIELD command



عملیات هایی مانند GET و SET و INCR را می توان در فیلد های جداگانه اعمال کرد. فیلد ها را می توان با offset ارجاع داد. 0 Index اولترين بیت از بايت اول در سمت چپ رشته است. این يك شاخص مبتنی بر صفر است همانطور كه می بینید اين شاخص در فیلد اول مشخص شده است. فیلد های بیتی نیز امکان نمایه سازی بر اساس موقعیت را فراهم می کنند.

Bitmap

BITFIELD

تعدادی دستور فرعی را برای دستکاری فیلدهای درون رشته ارائه می کند.

- BITFIELD key

[GET type offset]

[SET type offset value]

[INCRBY type offset increment]

[OVERFLOW WRAP | SAT | FAIL]

اگر نگاهی عمیق تر به BITFIELD بیندازیم، می بینید که GET، SET و INCRBY نیاز به تعیین نوع (type) دارند.

نوع (type) مشخص می کند که آیا مقدار با علامت "l" و بدون علامت با علامت "L" مشخص می شود .
بعدی تعداد بیت ها (offset) است.

overflowing, wrapping :The command provides several options to control when you write a value that is too large to fit

Bitmap

Sizes do not need to be byte aligned, so don't have to be a multiple of 8. It's perfectly valid to use "i5", "u11" etc.

BITFIELD command

Type

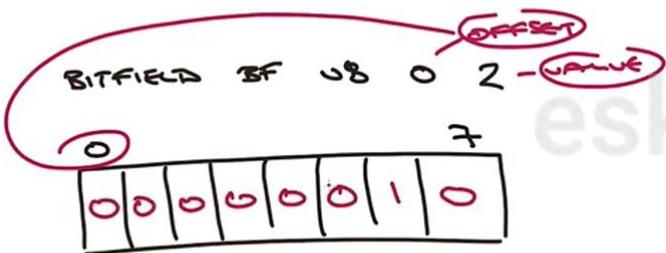
- Type
 - Signed (i) or Unsigned (u)
 - Size
- Limits
 - i64
 - u63
- No Schema

You can specify up to 64 bits of signed and 63 bits for unsigned. Finally Redis not record or maintain a mapping or scheme of these bit fields. This requires the developers to understand the correct offsets and datatype sizes in order to store and retrieve values correctly



Bitmap

BITFIELD



```
redis-enterprise:6379>
redis-enterprise:6379> bitfield bf1 set u8 0 2
1) (integer) 0
redis-enterprise:6379> bitfield bf1 get u8 0
1) (integer) 2
redis-enterprise:6379> get bf1
"\x02"
redis-enterprise:6379>
```

با نگاهی به این مثال "u8اندازه ساختار را در مجموع 8 بیت توصیف می کند."ا" نشان می دهد که بدون علامت است. همان طور که در نمودار می بینید، Offset 0 مهم ترین بیت از بایت اول است. تنظیم مقدار به 2 یا 10 در باینری منجر به تنظیم مقدار می شود همانطور که مشاهده می شود. افست اجازه می دهد تا چندین مقدار ذخیره و دستکاری شود.

Bitmap

BITFIELD

نمونه دوم از افزودن یک مقدار 8 بیتی بدون علامت

در این مورد از علامت هش یا پوند برای تعیین offset با موقعیت استفاده می کنیم و بازیابی مقدار اما با استفاده از offset و موقعیت وجود دارد.

```
redis-enterprise:6379>
redis-enterprise:6379> bitfield bf2 set u8 #1 5
1) (integer) 0
redis-enterprise:6379> bitfield bf2 get u8 #1 get u8 8
1) (integer) 5
2) (integer) 5
redis-enterprise:6379> get bf2
"\x00\x05"
redis-enterprise:6379>
```

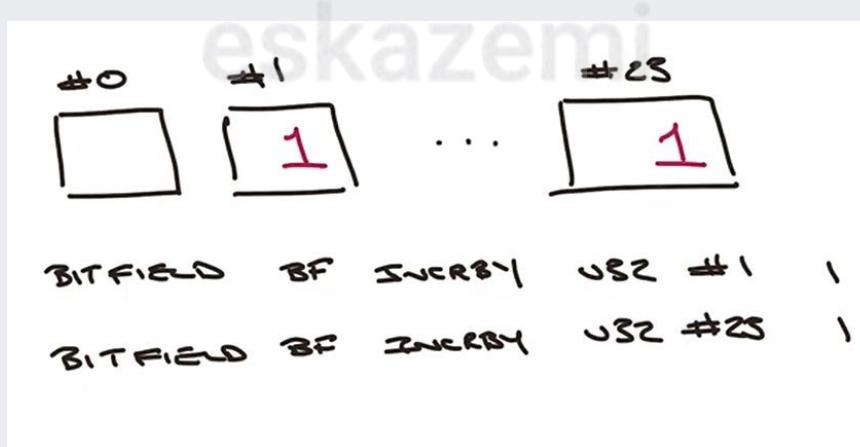
BITFIELD is
variadic, so
multiple GETs and
SETs can be
specified.

instead of calculating the bit offset yourself Redis will compute the bit position
based on the size of the type given.

Bitmap

BITFIELD

می توانید از انها برای نمایش هیستوگرام مقادیر استفاده کنید. در اینجا ما 24 شمارنده برای هر ساعت از روز داریم که در یک مقدار 32 بیتی بدون علامت ذخیره می شود. ما می توانیم از INCRBY و فیلد افست برای افزایش مقادیر استفاده کنیم



Bitmap

BITFIELD

Example:

```
redis-enterprise:6379>
redis-enterprise:6379> bitfield mykey set u8 0 42
1) (integer) 0
redis-enterprise:6379> bitfield mykey get u8 0
1) (integer) 42
redis-enterprise:6379> bitfield mykey incrby u8 0 1
1) (integer) 43
redis-enterprise:6379> type mykey
string
redis-enterprise:6379> object encoding mykey
"raw"
redis-enterprise:6379> █
```

Bitmap

BITFIELD

Example:

```
redis-enterprise:6379>
redis-enterprise:6379> bitfield ba4 set u1 7 1 set u1 15 1 set u1 23 1
1) (integer) 0
2) (integer) 0
3) (integer) 0
redis-enterprise:6379> bitcount ba4
(integer) 3
redis-enterprise:6379> bitcount ba4 1 2
(integer) 2
redis-enterprise:6379> bitcount ba4 0 -2
(integer) 2
redis-enterprise:6379> █
```

Byte 1, 2

Byte 1, 2

Bitmap

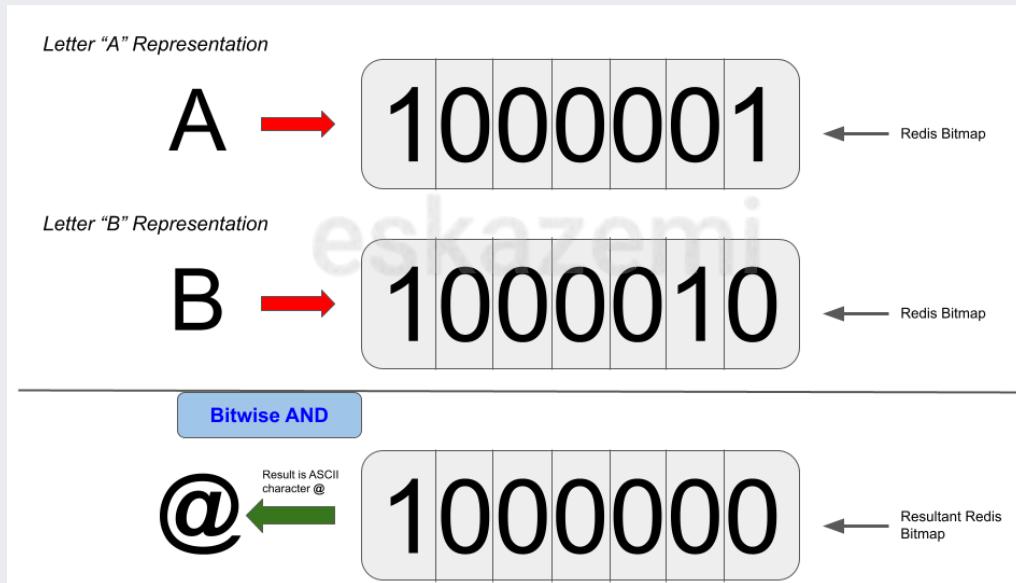
BITOP Command

Bitwise Operations

ما می توانیم عملیات های بیتی (bitwise operations) را روی رشته های Redis متعدد ذخیره شده در کلید های داده شده به صورت زیر انجام دهیم:

Bitmap

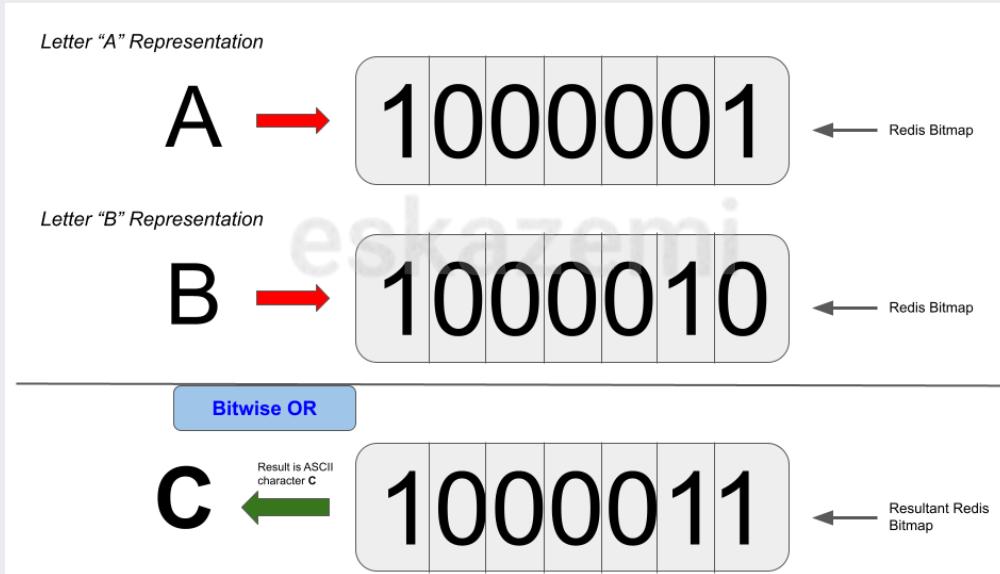
1- AND Operation



ما از رشته های A و B برای انجام عملیات بیتی AND استفاده کردیم

Bitmap

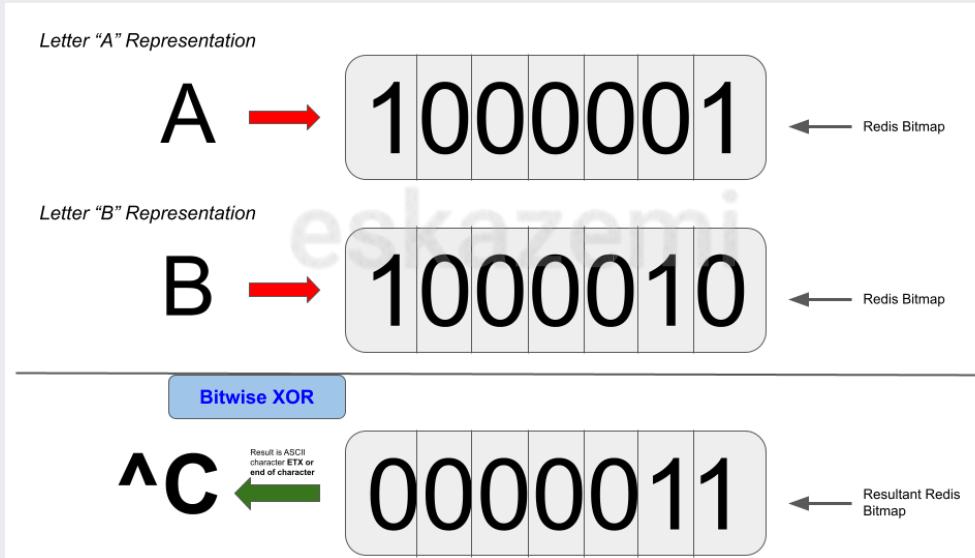
2- OR Operation



ما از رشته های A و B برای انجام عملیات بیتی OR استفاده کردیم

Bitmap

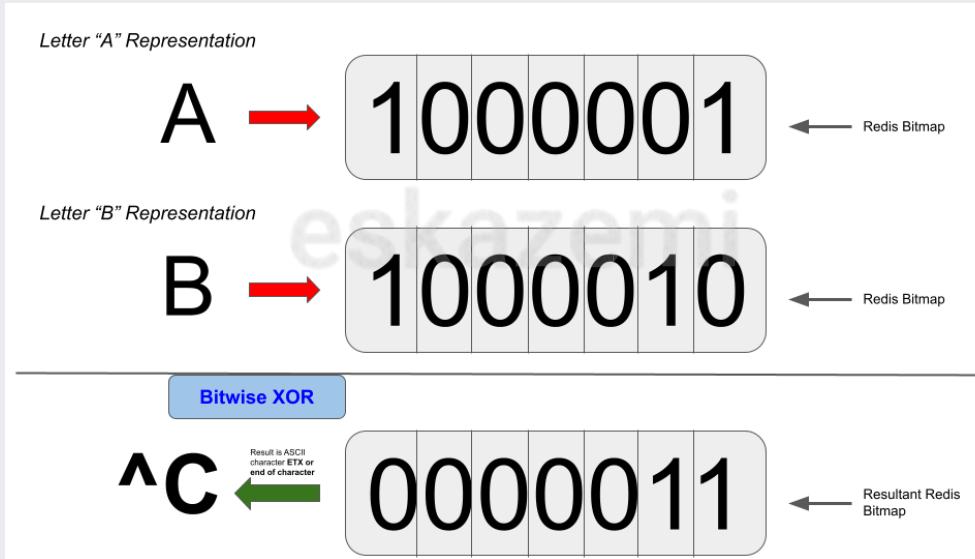
3- XOR Operation



ما از رشته های A و B برای انجام عملیات بیتی XOR استفاده کردیم

Bitmap

3- XOR Operation

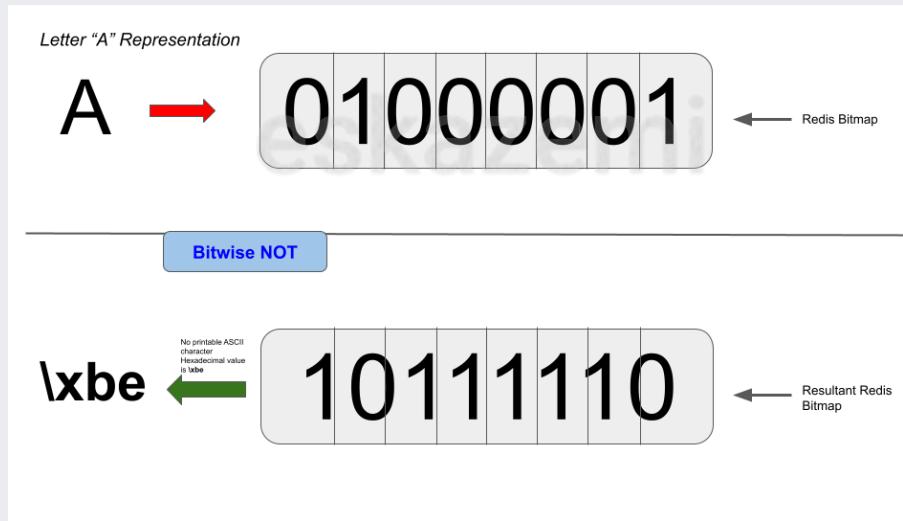


ما از رشته های A و B برای انجام عملیات بیتی XOR استفاده کردیم

Bitmap

4- NOT Operation

عملگر NOT به عنوان یک عملگر unary در عملیات بیتی Redis استفاده می شود از این رو تنها یک مقدار رشته را به عنوان ورودی می گیرد.



عمل بیتی NOT بر روی رشته A مقدار \xbe hexadecimal ASCII قابل چاپ مرتبط را بر نمی گرداند.

Bitmap

BITOP Command

برای انجام عملیات بیتی روی یک یا چند رشته (bitmaps) ذخیره شده در کلید های داده شده انجام می شود.

Time complexity:

این دستور با پیچیدگی زمانی ($O(N^2)$) عمل می کند که در آن N طول طولانی ترین رشته در مقایسه است در نتیجه بسیار کنتر از سایر عملیات bitmap است.

این دستور عملیات بیتی مشخص شده را انجام می دهد دستور BITOP یک مقدار صحیح به اندازه bitmap حاصل را بر می گرداند. اندازه bitmap حاصل برابر با اندازه طولانی ترین bitmap ورودی است. (رشته) حاصل را در **کلید مقصود** مشخص شده ذخیره می کند.

```
BITOP bitwise_operation destination_key bitmap_key [bitmap_key ...]
```

```
BITOP AND destkey srckeys
```

```
BITOP OR destkey srckeys
```

```
BITOP XOR destkey srckeys
```

```
BITOP NOT destkey srkey
```

همانطور که مشاهده می کنید NOT خاص است چرا که تنها یک کلید ورودی را می گیرد، زیرا **وارونگی بیت ها** را انجام می دهد در برخی موارد bitmap های ورودی دارای رشته های در اندازه های مختلف هستند.

“

Bitmap

- ✓ هنگامی که عملیاتی بین رشته هایی با طول های متفاوت انجام می شود، تمام رشته های کوتاه تر از طولانی ترین رشته در مجموعه طوری رفتار می شوند که گویی تا طول طولانی ترین رشته صفر شده اند.
- ✓ همین مساله در مورد کلیدهای ناموجود (non-existent keys) نیز صادق است که به عنوان جریانی از بایت های صفر تا طول طولانی ترین رشته در نظر گرفته می شوند.

BITOP Command

”

Bitmap

Use Cases

- Histograms
 - BITFIELD hist INCRBY u32 #23 1
 - BITFIELD hist GET u32 #23
- Permission bits & masks
 - BITOP XOR file1 request file1

Bit Data

- Bitfields & Bit Arrays
- Compact, optimized structures
- No explicit Bit data type
- Commands to manipulate Bits
- Use cases:
 - Histogram of counters
 - Permission bits & masks

```
{id1=time1.seq({ a: "foo", a: "bar"}) }
```

Stream

```
[A>B>C>C]
```

List

```
{A: (50.1, 0.5)}
```

Geospatial

```
{A<B<C}
```

Set

```
011011010110111101101101
```

Bitmap

```
hello world
```

String

```
{a: "hello", b: "world"}
```

Hash

```
01101101 01101111 01101101
```

Hyperlog

```
{23334}{6634728}{916}
```

Bitfield

```
{A:1, B:2, C:3}
```

Sorted set

“

چگونه اخراج داده ها را مدیریت می کند؟ Redis

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.

”

Things to Consider:

- Calculation for Time Complexity of the Command
- Cardinality of the data
- Multiplying factors
- Clock times are not O time

Thanks!



Any questions?

You can find me at:

- @eskazemi
- m.esmaeilkazemi@gmail.com



eskazemi

