

به نام خدا

Redis

ارائه دهنده: نرگس قانع


پایتون-مکتب ۶۱



- Redis چیست؟

- در واقع Redis یک ذخیره ساز ساختارهای داده ای در RAM هست.
- ساختارهای داده ای یا Data Structure می توانند یک رشته ساده و یا یک لیستی از داده ها باشند که Redis اونهارو توی RAM برای ما نگه داری می کنه و می تونیم با سرعت بالا اون ها رو بازیابی کنیم. از آنجایی که داده ها رو ذخیره و بازیابی می کند یک نوع دیتابیس هست و چون توی RAM نگه می داره اصطلاحاً **in-memory Database** نام گذاری می شود.
- اما برای هر داده ای که در Redis ذخیره می کنیم یک **key** یا کلید داریم و زمان بازیابی با استفاده از این **key** هست که به داده خودمون که همان **value** هست می رسیم و بنابراین Redis یک **key value database** هست.
- Redis یک دیتابیس NoSql می باشد

نگهداری دیتاها در Redis



در Ram نکه داشتن دیتا ها چه مزایا و معایبی دارد؟

- آیا از آنجایی که Redis داده ها را روی RAM نگه می دارد، بعد از خاموش و روشن شدن و یا هر اتفاق غیر قابل پیش بینی که بیافتد و RAM سیستم خالی شود داده های ما پاک می شوند؟
- خیر، Redis برای نگه داری دائمی داده ها آنها را با توجه به تنظیماتی که ما برای آن مشخص می کنیم به دیسک اصلی سیستم منتقل می کند و بعد از پاک شدن RAM دوباره می تواند آنها را منتقل کند و کار را از سر بگیرد.
- این ویژگی باعث شده اصطلاحاً به آن **on-disk persistence** بگویند.

RDB

- اگر قصد back up گیری از داده های خود به صورت فایل های جداگانه در فاصله های زمانی مشخص و بعد از تعداد مشخصی تغییر را داشته باشید می توانید از این روش استفاده کنید و کافی است به Redis بگویید که در چه بازه های زمانی به صورت تکراری و تا چه مدت از داده ها back up بگیرد.
- برای مثال هر ۳۰ دقیقه و به مدت ۳۰ روز از داده های خود می توانید back up بگیرید و در هر زمان که مشکلی پیش آمد آنها را برگردانید. در واقع RDB مخفف **Redis Database Backup** می باشد به این معنی که هر بار از کل دیتا بیس یک dump می گیرد و نگه می دارد.
- پس در این روش ممکن هست داده هایی رو از دست بدهید!

AOF

- این روش مخفف **Append Only File** هست به معنی اینکه یک بار فایل RDB را می سازد و با هر تغییر می تواند به آن فایل اضافه کند و به این صورت اگر چه شاید کمی کند تر باشد ولی برای داده های حساس تر گزینه بهتری هست.
- این روش قادر است بعد از هر بار درج و یا تغییر یک key جدید در Redis آن را ذخیره کند.
- روش های دیگر:
- غیر فعال کردن Persistence Mode به طور کلی و back up نگرفتن
- استفاده ترکیبی از RDB و AOF

نصب Redis

لینوکس:

```
wget http://download.redis.io/redis-stable.tar.gz  
tar xvzf redis-stable.tar.gz  
cd redis-stable  
make
```

```
1 | $ sudo apt install redis-server
```

ویندوز:

<https://github.com/microsoftarchive/redis/releases>

Step 2: Extract The ZIP File

Step 3: Move The Redis Folder To C Drive (Optional)

Step 3: Add Redis Path To Windows 10 Environment Variable

Right Click to the My Computer (This PC in Windows 10) icon and go to properties or move to **Control Panel\All Control Panel Items\System**. Then go to **Advanced System Settings > Environment Variables**.

On the System Variables section, double click on the PATH variable and add the path of **redis folder**.

اتصال به Redis

```
$ redis-server
```

```
$ redis-cli
```

```
redis 127.0.0.1:6379> ping
```

```
PONG
```

پیکر بندی

- بعد از نصب برای پیکربندی Redis می توانید به سراغ فایل `redis.conf` بروید که اگر از سیستم عامل `ubuntu` استفاده می کنید در آدرس `etc/redis/redis.conf/` می توانید آن را پیدا کنید.
- شما از طریق اعمال تغییرات در این فایل و یا دستورات `command line` می توانید تمامی تنظیمات Redis از جمله پورت مورد استفاده
- انتخاب روش `persistence`
- و هر نوع تنظیم دیگری را انجام دهید.
- برای مثال برای فعال کردن AOF می توانید مقدار `appendonly` را در فایل خود به `yes` تغییر دهید.

1 config set appendonly yes



منظور از اینکه هر دیتابیس چه دیتاتایپ هایی را ساپورت می کند چیست؟


انواع دیتاتایپ ها

- strings
- Lists
- Sets
- Sorted sets
- Hashes


strings

- ساده ترین دیتا تایپ مورد استفاده که داده ها را به صورت
 - Key/value
 - ذخیره میکند.


اگر یک **key** از قبل تعریف شده باشد و مجدد آن را **set** کنید بر روی اولی **value** را جایگزین می کند. از موارد استفاده این نوع داده ای شاید بتوان ذخیره فایل های **html** را مثال زد تا با **cache** کردن آنها بتوان مجدد به آنها دسترسی داشت و دوباره **render** نشوند. جالب هست که **Redis** می تواند تا مقدار **۵۱۲** مگابایت به ازای هر یک **string** ذخیره کند.



```
redis 127.0.0.1:6379> set mykey somevalue
OK
redis 127.0.0.1:6379> get mykey
"somevalue"
```



```
> set mykey newval nx  
(nil)  
> set mykey newval xx  
OK
```



```
> set counter 100
```

```
OK
```

```
> incr counter
```


```
(integer) 101
```

```
> incr counter
```

```
(integer) 102
```

```
> incrby counter 50
```

```
(integer) 152
```



```
> mset a 10 b 20 c 30
```


```
OK
```

```
> mget a b c
```

```
1) "10"
```

```
2) "20"
```

```
3) "30"
```



```
> set mykey hello
```

```
OK
```

```
> exists mykey
```


```
(integer) 1
```

```
> del mykey
```


```
(integer) 1
```

```
> exists mykey
```

```
(integer) 0
```



```
> set mykey x
OK
> type mykey
string
> del mykey
(integer) 1
> type mykey
none
```

```
> set key some-value
```

```
OK
```

```
> expire key 5
```


```
(integer) 1
```

```
> get key (immediately)
```

```
"some-value"
```

```
> get key (after some time)
```

```
(nil)
```



```
> set key 100 ex 10
```


```
OK
```

```
> ttl key
```


```
(integer) 9
```

lists


- این نوع برای داشتن لیستی از رشته ها هست. دقیقا لیستی که شما بتوانید از ابتدا یا انتهای آن مقادیر خود را اضافه و یا کم کنید.
- برای مثالی از کاربرد آن و وضوح بیشتر، یک **timeline** و یا یک تاریخچه چت رو در نظر بگیرید. باید به ترتیب پیام های رد و بدل شده بین ۲ کاربر را نگه دارید و یا برای **timeline** پست های منتشر شده بر اساس تاریخ وجود دارد.
- می تواند بیش از ۴ میلیارد المان در هر لیست نگه داری کند.



```
> rpush mylist A
(integer) 1
> rpush mylist B
(integer) 2
> lpush mylist first
(integer) 3
> lrange mylist 0 -1
1) "first"
2) "A"
3) "B"
```




```
> rpush mylist 1 2 3 4 5 "foo bar"
(integer) 9
> lrange mylist 0 -1
1) "first"
2) "A"
3) "B"
4) "1"
5) "2"
6) "3"
7) "4"
8) "5"
9) "foo bar"
```



```
> rpush mylist a b c
(integer) 3
> rpop mylist
"c"
> rpop mylist
"b"
> rpop mylist
"a"
```

```
> rpop mylist
(nil)
```



```
> rpush mylist 1 2 3 4 5
(integer) 5
> ltrim mylist 0 2
OK
> lrange mylist 0 -1
1) "1"
2) "2"
3) "3"
```



```
> set foo bar
```


```
OK
```

```
> lpush foo 1 2 3
```

```
(error) WRONGTYPE Operation against a key holding the wrong kind of value
```

```
> type foo
```

```
string
```

```
> del mylist  
(integer) 0  
> llen mylist  
(integer) 0  
> lpop mylist  
(nil)
```

sets

- شاید اگر لیستی از رشته ها را بخواهید بسازید از List استفاده کنید ولی اگر لیستی بخواهید که ترتیب در آن مهم نباشد
- بخواهید که مقادیر تکراری وارد لیست شما نشوند
- از هر کجای لیست که خواستید با سرعت خیلی بالا مقادیر را بگیرید
- بررسی کنید که آیا مقداری که دنبال آن هستید در این لیست هست یا خیر
- باید از Sets استفاده کنید.
- در واقع نوع داده ای Sets در Redis برای داشتن یک collection از رشته ها به عنوان مقادیر هست با این ویژگی مهم که دیگر نیاز به بررسی وجود این مقدار در collection نیست و هر چند بار که یک مقدار تکراری را به Set اضافه کنیم همان یک مقدار را در collection خودمون داریم.
- یکی دیگه از ویژگی های جالب Set این هست که شما می توانید چند تا Set را در Redis با هم Union کنید و از اونها استفاده کنید.
- از موارد پر کاربرد Set می توان به محاسبه ی تعداد کاربران یکتای بازدید کننده از سایت اشاره کرد به این صورت که یک Set از مقادیر ip تمام کاربران بازدید کننده داریم و از آنجایی که این مقادیر تکراری نمی باشد می توان با محاسبه تعداد آنها به بازدید یکتای سایت خود برسیم.
- می توانیم بیش از ۴ میلیارد مقدار متفاوت را در یک Set اضافه کنیم



```
> sadd myset 1 2 3
```

```
(integer) 3
```

```
> smembers myset
```

```
1. 3
```

```
2. 1
```

```
3. 2
```



```
> sismember myset 3
```

```
(integer) 1
```

```
> sismember myset 30
```

```
(integer) 0
```

```
> spop game:1:deck  
"C6"  
> spop game:1:deck  
"CQ"  
> spop game:1:deck  
"D1"  
> spop game:1:deck  
"CJ"  
> spop game:1:deck  
"SJ"
```


```
1 | > SREM myset "World"
```

```
2 | 1
```




Sorted sets

- همان طور که از اسم این نوع داده مشخص هست شبیه Sets هست یعنی یک collection از مقادیر رشته ای بدون تکرار، اما برخلاف Set می توان در Sorted Set یک مقدار به عنوان Score به هر المان داد.
- با این مقدار می توانیم از کمترین به بیشترین score المان های خودمون رو مرتب کنیم و این نکته را هم باید بدونیم که برخلاف مقادیر المان ها، score ها می توانند تکراری هم باشند.
- برای مثالی از کاربرد این نوع داده می توانیم لیستی از task ها را در نظر بگیریم که تکراری نیستند و هر کدام یک درجه اهمیتی دارند. حالا می توانیم این لیست را از هر کجایی که نیاز داریم مرتب کنیم.



```
1  >ZADD myzset 1 "one"
2  1
3  > ZADD myzset 2 "two"
4  1
5  > ZADD myzset 3 "three"
6  1
   ↵
```



```
1 | > ZSCORE myzset "two"
```

```
2 | "2"
```

```
◀
```





```
1 > ZRANGE myzset 2 3
```

```
2 1) "two"
```

```
3 2) "three"
```


```
◀
```



```
1 | > ZREM myzset "two"
```

```
2 | 1
```

```
◀
```



```
> zrevrange hackers 0 -1
```

```
1) "Linus Torvalds"
```

```
2) "Yukihiro Matsumoto"
```

```
3) "Sophie Wilson"
```

```
4) "Richard Stallman"
```


```
5) "Anita Borg"
```

```
6) "Alan Kay"
```

```
7) "Claude Shannon"
```

```
8) "Hedy Lamarr"
```

```
9) "Alan Turing"
```



```
> zrange hackers 0 -1 withscores
```

```
1) "Alan Turing"
```

```
2) "1912"
```

```
3) "Hedy Lamarr"
```

```
4) "1914"
```

```
5) "Claude Shannon"
```

```
6) "1916"
```

```
7) "Alan Kay"
```

```
8) "1940"
```

```
9) "Anita Borg"
```

```
10) "1949"
```

```
11) "Richard Stallman"
```

```
12) "1953"
```

```
13) "Sophie Wilson"
```


```
14) "1957"
```

```
15) "Yukihiro Matsumoto"
```

```
16) "1965"
```

```
17) "Linus Torvalds"
```

```
18) "1969"
```



```
> zrangebyscore hackers -inf 1950
```


```
1) "Alan Turing"
```

```
2) "Hedy Lamarr"
```

```
3) "Claude Shannon"
```

```
4) "Alan Kay"
```

```
5) "Anita Borg"
```




```
> zremrangebyscore hackers 1940 1960  
(integer) 4
```

```
> zrank hackers "Anita Borg"  
(integer) 4
```

Hashes

- اگر می خواهید داخل هر `key` یک `object` نگه دارید `Hash` دقیقا پاسخ نیاز شماست.
- هر `Hash` داخل `Redis` این امکان رو به شما می دهد که به ازای هر `key` یک `map` از رشته هایی به صورت `key value` را نگه دارید یعنی دقیقا همان چیزی که از `object` انتظار دارید.
- از موارد کاربرد آن می توان به نگه داشتن اطلاعات کاربران لاگین شده به سیستم اشاره کرد.
- فرض کنید به ازای هر کاربری که به سیستم شما لاگین می شود یک `key` از نوع `Hash` بسازید و داخل آن اطلاعاتی مثل نام، ایمیل، تلفن و هر چیزی که نیاز دارید نگه داری کنید.



```
> hmset user:1000 username antirez birthyear 1977 verified 1
OK
> hget user:1000 username
"antirez"
> hget user:1000 birthyear
"1977"
> hgetall user:1000
1) "username"
2) "antirez"
3) "birthyear"
4) "1977"
5) "verified"
6) "1"
```





```
> hmget user:1000 username birthyear no-such-field
```

```
1) "antirez"
```

```
2) "1977"
```

```
3) (nil)
```



```
1 > HEXISTS user:1000 age
```

```
2 1
```

```
◀
```

```
> hincrby user:1000 birthyear 10  
(integer) 1987
```

```
> hincrby user:1000 birthyear 10  
(integer) 1997
```