EGE ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

2020-2021 Güz Yarıyılı Proje 1

Redis Tutorial – User Guide

Hazırlayanlar:

Yaren Gündüz - 91200000186

Ünburak Öznur - 91200000205

Ocak, 2021

İZMİR

İÇİNDEKİLER

1 REDIS KURULUMU

1.1 Windows İşletim Sisteminde

- Redis aslında Windows desteği barındırmaz fakat Microsoft Open Tech 64 bitlik Windows sürümleri için portlamış oldukları bir redis sürümü bulunmaktadır.Bunun kurulumu şu şekilde gerçekleşir.
- https://github.com/MSOpenTech/redis/releases adresinden son çıkan redis sürümünü indirin.
- Bu zip dosyasına tıklayıp.Bilgisaayrınızda uygun bir path vererek içindeki redis-server exe'sini çalıştırın.
- Command prompt'u açıp redis-cli yazın ve redisin çalışabilir olduğunu test edin.
- Daha sonra https://redisdesktop.com/download adres üzerinden veya internet üzerinde para ödemedn alınan redis desktop manager indirip redis sunucusu için iyi bir arayüzle komutlarımızı kontrol edebiliriz.
- Redis dektop manager açılınca connect to redis'e tıklanır ve sunucuya bir isim,port ve local'de çalışıyorsa 127.0.0.1 yazılıp. Gerekli IP adresi sağlanınca redis sunucusuna bağlanılır.

1.2 Linux İşletim Sisteminde

- Aşağıdaki kod satırları ile terminale bu komutlar girilip redis bulunup,Ubuntu tarafından indirilir.

```
sudo apt-get update

sudo apt-get install redis-server
```

- Bu komutlar redisi makinemize indirecektir.
- redis-server komutu ile redis sunucusunu başlatırız.
- redis-cli komutu ile redisin çalışır olma durumu kontrol edilir.Bu komut sonrasında redisin localhostumuzda çalıştığına dair bir dönüt alırız. redis 127.0.0.1:6379 aslında redisin localimizde genelde 6379 portunda değiştirmediğimiz durumda çalıştığını bize bildirir. Ping atarak bu sunucudan cevap gelip-gelmediğini kontrol etmiş oluruz.
- https://redisdesktop.com/download bu link üzerinden redis'i kolayca yönetebileceğimiz uygulamayı indirerek key-value DB mizi rahatlıkla kullnabiliriz.

1.3 Macos İşletim Sisteminde

- 1.2 Linux işletim sistemi kurulumundaki komutlar yerine alttaki komutlar kullanılır. Geri kalan yapılandırma aynı şekildedir. Komutlar:

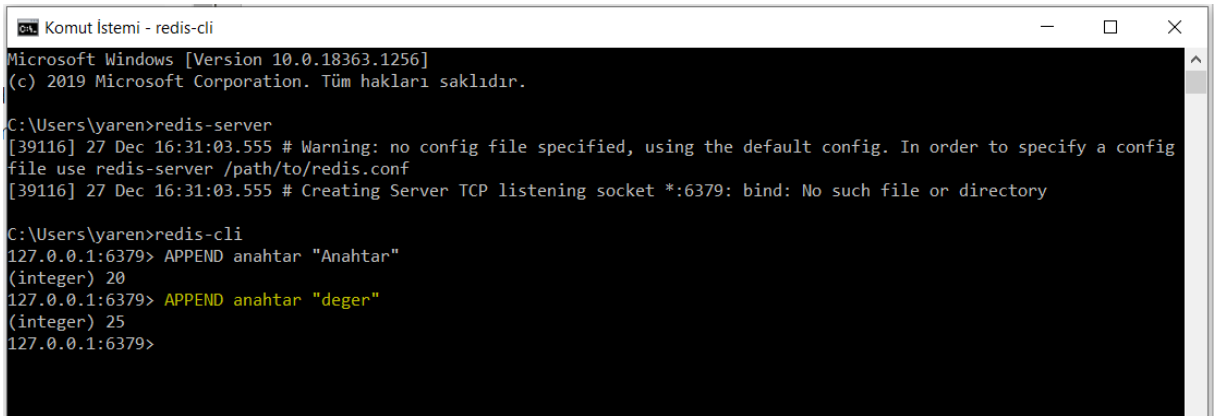
```
brew update

brew install redis
```

2 STRINGS VERİ TİPİ KOMUTLARI

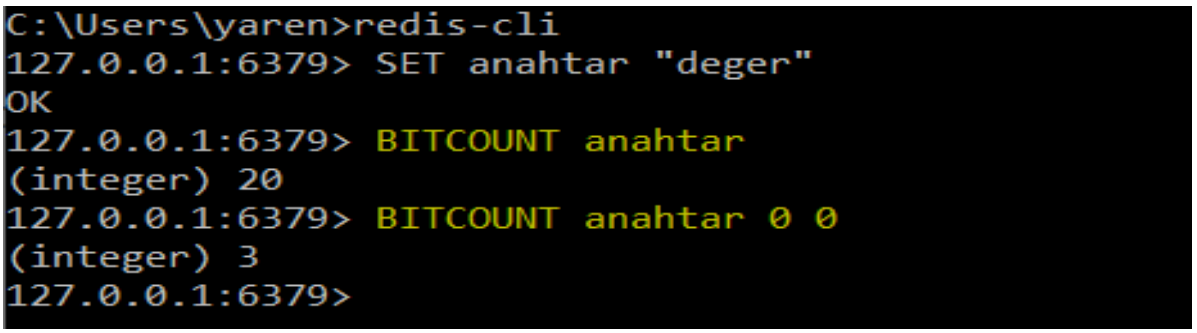
2.1 Append

Karmaşıklık O(1). Anahtar zaten varsa ve bir dizeyse, bu komut dizenin sonuna değeri ekler. Anahtar yoksa, oluşturulur ve boş bir dize olarak ayarlanır.



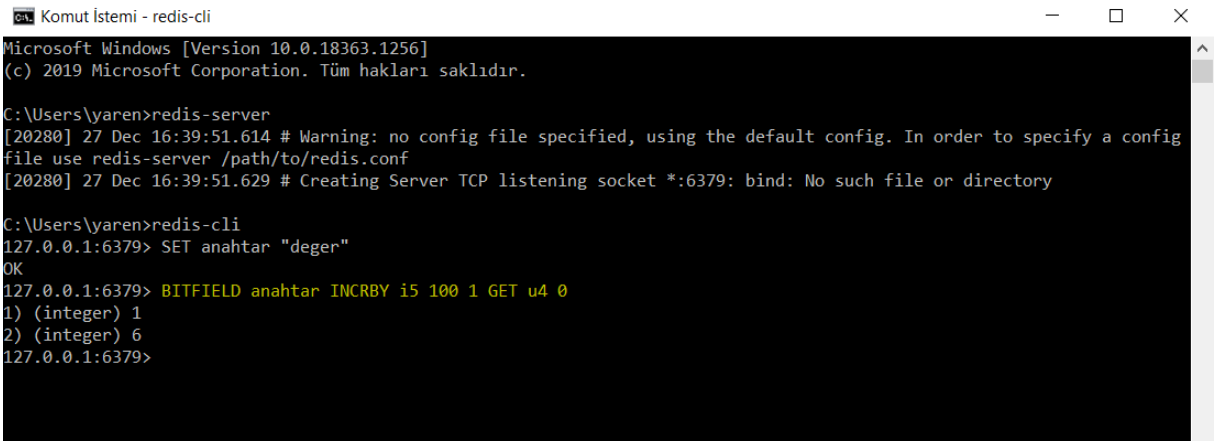
2.2 Bitcount

Karmaşıklık O(N) . Bir dizedeki set bitlerinin sayısını (popülasyon sayımı) sayar. Varsayılan olarak dizede bulunan tüm baytlar incelenir. Sayma işlemini yalnızca ek bağımsız değişkenlerin başlangıcını ve sonunu geçen bir aralıkta belirtmek mümkündür.



2.3 BitField

Komut, bir Redis dizesini bir bit dizisi olarak ele alır ve değişen bit genişliklerine sahip belirli tam sayı alanlarını ve keyfi olmayan (gerekli) hizalı ofseti adresleme yeteneğine sahiptir. Bu komutu kullanarak pratik anlamda, örneğin, 1234 bit ofsetinde işaretli 5 bitlik bir tamsayıyı belirli bir değere ayarlayabilir, 4567 ofsetinden 31 bitlik işaretsiz bir tamsayı alabilirsiniz. Benzer şekilde komut, belirtilen tamsayıların artışlarını ve azalmalarını işler. Örneğin, aşağıdaki komut, bit ofseti 100'de 5 bitlik işaretli bir tamsayıyı artırır ve bit ofseti 0'da 4 bitlik işaretsiz tamsayının değerini alır.



2.4 Bitop

Birden çok anahtar arasında (dize değerleri içeren) bit düzeyinde bir işlem gerçekleştirir ve sonucu hedef anahtarda saklar. AND,OR,XOR,NOT operasyonlarını destekler.

```
127.0.0.1:6379> SET yaren "YAREN"
OK
127.0.0.1:6379> SET burak "BURAK"
OK
127.0.0.1:6379> BITOP AND dest yaren burak
(integer) 5
127.0.0.1:6379> GET dest
"@ARAJ"
127.0.0.1:6379>
```

2.5 Bitpos

Bir dizede 1 veya 0 olarak ayarlanan ilk bitin konumunu döndürür. Diziyi soldan sağa bir bit dizisi olarak düşünerek konum döndürülür, burada birinci baytın en önemli biti 0 konumunda, ikinci baytın en önemli biti konum 8'de vb.

```
C:\Users\yaren>redis-cli
127.0.0.1:6379> SET anahtar "deger"
OK
127.0.0.1:6379> BITFIELD anahtar INCRBY i5 100 1 GET u4 0
1) (integer) 1
2) (integer) 6
127.0.0.1:6379> BITOP AND anahtar srckey
(integer) 0
127.0.0.1:6379> SET yaren "YAREN"
OK
127.0.0.1:6379> SET burak "BURAK"
OK
127.0.0.1:6379> BITOP AND dest yaren burak
(integer) 5
127.0.0.1:6379> GET dest
"@ARAJ"
127.0.0.1:6379> BITPOS anahtar 1 2
(integer) -1
127.0.0.1:6379>
```

2.6 Decr

Anahtarda saklanan sayıyı birer birer azaltır. Anahtar yoksa, işlemi gerçekleştirmeden önce 0'a ayarlanır.

```
127.0.0.1:6379> SET anahtar "Anahtar"
OK
127.0.0.1:6379> SET intanahtar 97
OK
127.0.0.1:6379> DECR intanahtar
(integer) 96
127.0.0.1:6379>
```

2.7 Decrby

Anahtarda saklanan sayıyı azaltarak azaltır. Anahtar yoksa, işlemi gerçekleştirmeden önce 0'a ayarlanır.

```
127.0.0.1:6379> SET anahtar "Anahtar"
OK
127.0.0.1:6379> SET intanahtar 97
OK
127.0.0.1:6379> DECR intanahtar
(integer) 96
127.0.0.1:6379> DECRBY intanahtar 5
(integer) 91
127.0.0.1:6379>
```

2.8 Get

Anahtarın değerini döndürür.

```
127.0.0.1:6379> GET intanahtar
"91"
127.0.0.1:6379>
```

2.9 Getbit

Anahtarda depolanan dizi değerinde ofsetteki bit değerini döndürür.

```
127.0.0.1:6379> GET intanahtar
"91"
127.0.0.1:6379> GETBIT intanahtar 7
(integer) 1
127.0.0.1:6379>
```

2.10 Getrange

Anahtarda saklanan dize değerinin alt dizesini döndürür, başlangıç ve bitiş uzaklıklarına göre belirlenir (her ikisi de dahildir).

```
127.0.0.1:6379> SET yaren "ben yaren,merhaba"
OK
127.0.0.1:6379> GETRANGE yaren 0 -1
"ben yaren,merhaba"
127.0.0.1:6379> GETRANGE yaren 0 3
"ben "
127.0.0.1:6379>
```

2.11 Getset

Atomik olarak anahtarı değere ayarlar ve anahtarda saklanan eski değeri döndürür. Anahtar varsa, ancak bir dize değeri taşımadığında bir hata döndürür.

```
127.0.0.1:6379> GETSET yaren
(error) ERR wrong number of arguments for 'getset' command
127.0.0.1:6379> GETSET yaren "0"
"ben yaren,merhaba"
127.0.0.1:6379> SET unburak "ben unburak,merhaba"
OK
127.0.0.1:6379> GETSET unburak
(error) ERR wrong number of arguments for 'getset' command
127.0.0.1:6379> GETSET unburak "1"
"ben unburak,merhaba"
127.0.0.1:6379>
```

2.12 Incr

Anahtarda saklanan sayıyı birer birer artırır. Anahtar yoksa, işlemi gerçekleştirmeden önce 0'a ayarlanır. Anahtar yanlış türde bir değer veya tamsayı olarak temsil edilemeyen bir dize içeriyorsa bir hata döndürülür. Bu işlem 64 bitlik işaretli tamsayılarla sınırlıdır.

```
127.0.0.1:6379> SET yaren "22"
OK
127.0.0.1:6379> INCR yaren
(integer) 23
127.0.0.1:6379>
```

2.13 Incrby

Anahtarda saklanan sayıyı adım adım artırır. Anahtar yoksa, işlemi gerçekleştirmeden önce 0'a ayarlanır. Anahtar yanlış türde bir değer veya tamsayı olarak temsil edilemeyen bir dize içeriyorsa bir hata döndürülür. Bu işlem 64 bitlik işaretli tamsayılarla sınırlıdır.

```
127.0.0.1:6379> SET yaren "22"
OK
127.0.0.1:6379> INCR yaren
(integer) 23
127.0.0.1:6379> INCRBY yaren 7
(integer) 30
127.0.0.1:6379>
```

2.14 Incrbyfloat

Anahtarda saklanan kayan nokta sayısını temsil eden dizeyi belirtilen artışla artırır. Negatif bir artış değeri kullanarak, sonuç, anahtarda depolanan değerin azalmasıdır (eklemenin bariz özellikleri ile). Anahtar yoksa, işlemi gerçekleştirmeden önce 0'a ayarlanır. Aşağıdaki koşullardan biri meydana gelirse bir hata döndürülür.

```
127.0.0.1:6379> SET burak 24.5
OK
127.0.0.1:6379> INCRBYFLOAT burak 2.0e2
"224.5"
127.0.0.1:6379>
```

2.15 Mget

Belirtilen tüm anahtarların değerlerini döndürür. Bir dizge değeri taşımayan veya var olmayan her anahtar için özel null değeri döndürülür. Bu nedenle operasyon asla başarısız olmaz.

```
127.0.0.1:6379> MGET burak
1) "224.5"
127.0.0.1:6379> MGET burak yaren nonexisting
1) "224.5"
2) "30.399999999999999"
3) (nil)
127.0.0.1:6379>
```

2.16 Mset

Verilen anahtarları ilgili değerlerine ayarlar. MSET, tıpkı normal SET gibi mevcut değerleri yeni değerlerle değiştirir. Mevcut değerlerin üzerine yazmak istemiyorsanız MSET'e bakın.

```
127.0.0.1:6379> MSET burak "yaren" yaren "burak"
OK
127.0.0.1:6379> burak
(error) ERR unknown command 'burak'
127.0.0.1:6379> MGET burak
1) "yaren"
127.0.0.1:6379> MGET yaren
1) "burak"
127.0.0.1:6379>
```

2.17 Msetnx

Verilen anahtarları ilgili değerlerine ayarlar. MSETNX, yalnızca tek bir anahtar zaten mevcut olsa bile hiçbir işlem gerçekleştirmez.

```
127.0.0.1:6379> SET yaren "Yaren"
OK
127.0.0.1:6379> SET burak "Burak"
OK
127.0.0.1:6379> MSETNX yaren "Hello Burak" burak "Hello Yaren"
(integer) 0
127.0.0.1:6379> MGET yaren burak damla
1) "Yaren"
2) "Burak"
3) (nil)
127.0.0.1:6379>
```

2.18 Psetex

PSETEX, yalnızca sona erme süresinin saniye yerine milisaniye cinsinden belirtilmesi farkıyla tam olarak SETEX gibi çalışır.

```
127.0.0.1:6379> psetex mykey 1000 "Hello"
OK
127.0.0.1:6379> pttl mykey
(integer) -2
127.0.0.1:6379> PTTL mykey
(integer) -2
127.0.0.1:6379> get mykey
(nil)
127.0.0.1:6379> psetex mykey 100000 "Hello"
OK
127.0.0.1:6379> pttl mykey
(integer) 96528
127.0.0.1:6379> get mykey
"Hello"
127.0.0.1:6379>
```

2.19 Set

Dize değerini tutmak için anahtarı ayarlayın. Anahtar zaten bir değer içeriyorsa, türüne bakılmaksızın üzerine yazılır. Başarılı SET işleminde anahtarla ilişkilendirilmiş önceki herhangi bir yaşam süresi atılır.

```
127.0.0.1:6379> SET hello "Hello"
OK
127.0.0.1:6379> GET hello
"Hello"
127.0.0.1:6379> SET hello "there" EX 60
OK
127.0.0.1:6379>
```

2.20 Setbit

Anahtarda depolanan dizi değerindeki ofsetteki biti ayarlar veya temizler. Bit, değere bağlı olarak ayarlanır veya silinir, bu 0 veya 1 olabilir.

```
127.0.0.1:6379> SET hello "Hello"
OK
127.0.0.1:6379> GET hello
"Hello"
127.0.0.1:6379> SET hello "there" EX 60
OK
127.0.0.1:6379> SETBIT hello 7 1
(integer) 0
127.0.0.1:6379> GET hello
"\x01"
127.0.0.1:6379>
```

2.21 Setex

Dize değerini tutması için anahtarı ayarlayın ve belirli bir saniye sayısından sonra anahtarı zaman aşımına ayarlar.

```
127.0.0.1:6379> SETEX hello 10 "Hello"
OK
127.0.0.1:6379> TTL hello
(integer) 6
127.0.0.1:6379> GET hello
"Hello"
127.0.0.1:6379>
```

2.22 Setnx

Anahtar yoksa, dize değerini tutacak şekilde anahtarı ayarlayın. Bu durumda SET'e eşittir. Anahtar zaten bir değer tuttuğunda, hiçbir işlem gerçekleştirilmez. SETEX, "eXists değilse SET" in kısaltmasıdır.

```
127.0.0.1:6379> SETNX hello "Hello"
(integer) 1
127.0.0.1:6379> SETNX hello "World"
(integer) 0
127.0.0.1:6379> GET hello
"Hello"
127.0.0.1:6379>
```

2.23 SetRange

Değerin tüm uzunluğu için belirtilen ofsetten başlayarak anahtarda depolanan dizenin bir kısmının üzerine yazar. Göreli konum anahtardaki dizenin geçerli uzunluğundan daha büyükse, dizge ofset uydurmak için sıfır bayt ile doldurulur. Var olmayan anahtarlar boş dizeler olarak kabul edilir, bu nedenle bu komut, ofsette değer ayarlayabilecek kadar büyük bir dizge tuttuğundan emin olacaktır.

```
127.0.0.1:6379> SET hello "Hello World"
OK
127.0.0.1:6379> SETRANGE hello 6 "Redis"
(integer) 11
127.0.0.1:6379> GET hello
"Hello Redis"
127.0.0.1:6379>
```

2.24 Stralgo

STRALGO, dizeler üzerinde çalışan karmaşık algoritmalar uygular. Şu anda uygulanan tek algoritma LCS algoritmasıdır (en uzun ortak alt dize). Ancak gelecekte yeni algoritmalar uygulanabilir. Bu komutun amacı, Redis kullanıcılarına hızlı uygulamalara ihtiyaç duyan ve normalde çoğu programlama dilinin standart kitaplığında sağlanmayan algoritmalar sağlamaktır.

```
127.0.0.1:6379> MSET key1 ohmytext key2 mynewtext
OK
127.0.0.1:6379> STRALGO LCS KEYS key1 key2
"mytext"
127.0.0.1:6379>
```

2.25 Strlen

Anahtarda saklanan dize değerinin uzunluğunu döndürür. Anahtar dize olmayan bir değer içerdiğinde bir hata döndürülür.

```
127.0.0.1:6379> SET cumle "Bu uzun bir cumledir"
OK
127.0.0.1:6379> STRLEN cumle
(integer) 20
127.0.0.1:6379> STRLEN yaren
(integer) 0
127.0.0.1:6379> STRLEN unburak
(integer) 1
127.0.0.1:6379>
```

3 Hash Veri Tipi:

3.1 Hdel

Belirtilen alanları anahtarda depolanan karmadan kaldırır. Bu hash içinde bulunmayan belirtilen alanlar yok sayılır. Anahtar yoksa, boş bir karma olarak kabul edilir ve bu komut 0 döndürür.

```
127.0.0.1:6379> HSET yaren field "YAREN"
(integer) 1
127.0.0.1:6379> HDEL yaren field
(integer) 1
```

3.2 Hexists

Alan, anahtarda depolanan hash'te mevcut bir alan ise döndürür. 1, hash alan içeriyorsa. hash alan içermiyorsa veya anahtar yoksa 0 döner.

```
127.0.0.1:6379> HSET burak field1 "unburak"
(integer) 1
127.0.0.1:6379> HEXISTS burak field1
(integer) 1
127.0.0.1:6379> HEXISTS yaren field
(integer) 0
127.0.0.1:6379>
```

3.3 Hget

Anahtarda depolanan hashteki alanla ilişkili değeri döndürür.

```
127.0.0.1:6379> HSET burak field1 "unburak"
(integer) 1
127.0.0.1:6379> HEXISTS burak field1
(integer) 1
127.0.0.1:6379> HEXISTS yaren field
(integer) 0
127.0.0.1:6379> HGET burak field1
"unburak"
127.0.0.1:6379>
```

3.4 Hgetall

Anahtarda saklanan hash'in tüm alanlarını ve değerlerini döndürür. Döndürülen değerde, her alan adından sonra değeri gelir, bu nedenle yanıtın uzunluğu hash'in iki katıdır.

```
127.0.0.1:6379> HSET unburak field "UNBURAK"
(error) WRONGTYPE Operation against a key holding the wrong kind of value
127.0.0.1:6379> HSET burak field1 "unburak"
(integer) 1
127.0.0.1:6379> HEXISTS burak field1
(integer) 1
127.0.0.1:6379> HEXISTS yaren field
(integer) 0
127.0.0.1:6379> HGET burak field1
"unburak"
127.0.0.1:6379> HGETALL burak
1) "field1"
2) "unburak"
127.0.0.1:6379>
```

3.5 Hincrby

Anahtarda depolanan karmadaki alanda depolanan sayıyı artırarak artırır. Anahtar yoksa, hash içeren yeni bir anahtar oluşturulur. Alan yoksa, işlem gerçekleştirilmeden önce değer 0 olarak ayarlanır.

```
127.0.0.1:6379> HSET yas field 34
(integer) 1
127.0.0.1:6379> HINCRBY yas field 4
(integer) 38
127.0.0.1:6379>
```

3.6 Hıncrbyfloat

Anahtarda saklanan ve bir kayan nokta numarasını temsil eden bir karmanın belirtilen alanını belirtilen artışla artırın. Artış değeri negatifse, sonuç, hash alanı değerinin artırılması yerine azaltılmasıdır. Alan yoksa, işlemi gerçekleştirmeden önce 0 olarak ayarlanır.

```
127.0.0.1:6379> HSET yaren field1 16.5
(integer) 0
127.0.0.1:6379> I
(error) ERR unknown command 'I'
127.0.0.1:6379> HINCRBYFLOAT yaren field1 0.7
"17.199999999999999"
127.0.0.1:6379>
```

3.7 Hkeys

Anahtarda saklanan hashteki tüm alan adlarını döndürür.

```
127.0.0.1:6379> HKEYS yaren
1) "field1"
127.0.0.1:6379> HKEYS burak
1) "field1"
127.0.0.1:6379>
```

3.8 Hlen

Anahtarda depolanan hash' in içerdiği alanların sayısını döndürür.

```
127.0.0.1:6379> HSET myhash field1 "Hello"
(integer) 0
127.0.0.1:6379> HSET myhash field2 "World"
(integer) 1
127.0.0.1:6379> HSET myhash field3 "How are you?"
(integer) 1
127.0.0.1:6379> HLEN myhash
(integer) 3
127.0.0.1:6379>
```

3.9 Hmget

Anahtarda depolanan karmada belirtilen alanlarla ilişkili değerleri döndürür. Hash te bulunmayan her alan için bir sıfır değeri döndürülür. Var olmayan anahtarlar boş hashler olarak değerlendirildiğinden, mevcut olmayan bir anahtara karşı HMGET'i çalıştırmak sıfır değerlerin bir listesini döndürür.

```
127.0.0.1:6379> HMGET myhash field1 field2 nofield
1) "Hello"
2) "World"
3) (nil)
127.0.0.1:6379>
```

3.10 Hmset

Belirtilen alanları anahtarda saklanan karmadaki ilgili değerlerine ayarlar. Bu komut, hash'de zaten mevcut olan belirli alanların üzerine yazar. Anahtar yoksa, hash içeren yeni bir anahtar oluşturulur.

```
127.0.0.1:6379> HMSET myhash field1 "Hello" field2 "World"
OK
127.0.0.1:6379> HGET myhash
(error) ERR wrong number of arguments for 'hget' command
127.0.0.1:6379> HGET myhash field2
"World"
127.0.0.1:6379> HGET myhash field1
"Hello"
127.0.0.1:6379>
```

3.11 Hscan

hash'i tarar. Scan komutu öncelikli çalıştırır bu komutu.

```
127.0.0.1:6379> scan 17
1) "0"
2) 1) "key"
   2) "users"
   3) "3"
   4) "myhash"
   5) "burak1"
   6) "YAREN"
   7) "dest"
   8) "anahtar"
   9) "yas"
```

3.12 Hset

Anahtarda depolanan hashteki alanı değere ayarlar. Anahtar yoksa, hash içeren yeni bir anahtar oluşturulur. Alan hashte zaten mevcutsa, üzerine yazılır.

```
127.0.0.1:6379> HSET yaren field "Merhaba"
(integer) 1
127.0.0.1:6379> HSET yaren field1 "Ben Yaren"
(integer) 0
127.0.0.1:6379> HGET yaren field1
"Ben Yaren"
127.0.0.1:6379>
```

3.13 Hsetnx

Anahtarda depolanan hashteki alanı değere ayarlar, yalnızca alan henüz mevcut değilse. Anahtar yoksa, hash içeren yeni bir anahtar oluşturulur. Alan zaten varsa, bu işlemin hiçbir etkisi yoktur.

```
C:\Users\yaren>redis-cli
127.0.0.1:6379> HSET damla "Ben DAMLA"
(error) ERR wrong number of arguments for 'hset' command
127.0.0.1:6379> HSET damla field1 "10 Yasindayim"
(integer) 0
127.0.0.1:6379> HSETNX damla field2 "Merhaba"
(integer) 1
127.0.0.1:6379> HGET damla field2
"Merhaba"
127.0.0.1:6379>
```

3.14 Hstrlen

Anahtarda depolanan hashteki alanla ilişkili değerin dize uzunluğunu döndürür. Anahtar veya alan yoksa, 0 döndürülür.

```
C:\Users\yaren>redis-cli
127.0.0.1:6379> HSET damla "Ben DAMLA"
(error) ERR wrong number of arguments for 'hset' command
127.0.0.1:6379> HSET damla field1 "10 Yasindayim"
(integer) 0
127.0.0.1:6379> HSETNX damla field2 "Merhaba"
(integer) 1
127.0.0.1:6379> HGET damla field2
"Merhaba"
127.0.0.1:6379> HSTRLEN damla
(error) ERR wrong number of arguments for 'hstrlen' command
127.0.0.1:6379> HSTRLEN damla field2
(integer) 7
127.0.0.1:6379> HSTRLEN damla field1
(integer) 13
127.0.0.1:6379> HSTRLEN damla field
(integer) 5
127.0.0.1:6379>
```

3.15 Hvals

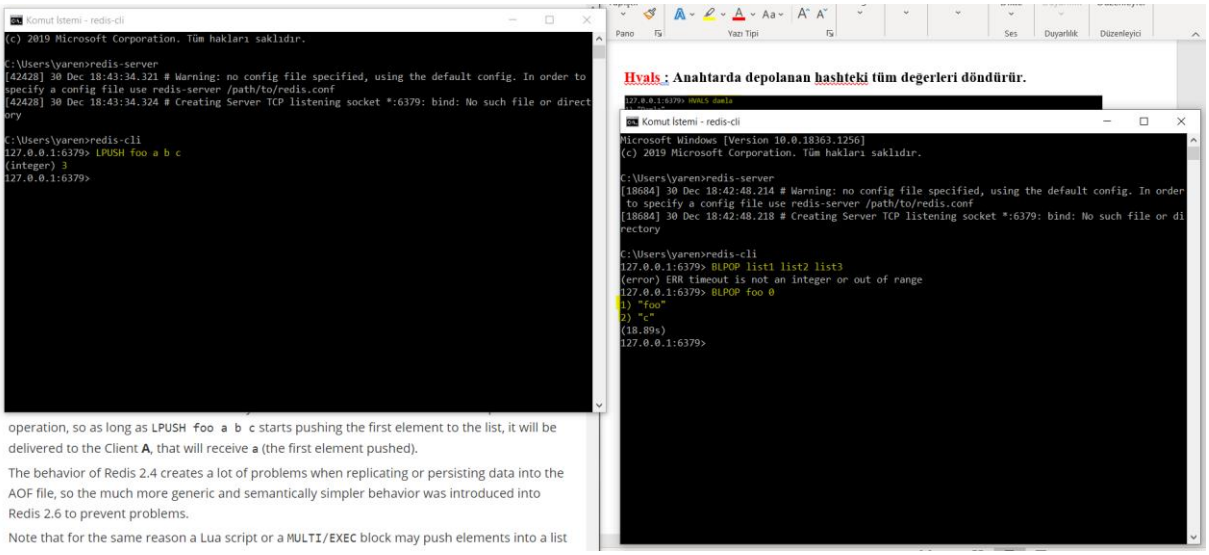
Anahtarda depolanan hashteki tüm değerleri döndürür.

```
127.0.0.1:6379> HVALS damla
1) "Damla"
2) "10 Yasindayim"
3) "Merhaba"
127.0.0.1:6379>
```

4 Liste Veri Tipi

4.1 Blpop

BLPOP, bir engelleme listesi pop ilkelidir. LPOP'nin bloke edici sürümüdür çünkü verilen listelerin hiçbirinden çıkacak öğe olmadığında bağlantıyı engeller. İlk listenin başından boş olmayan bir öğe çıkarılır ve verilen anahtarlar verildikleri sırayla kontrol edilir.



4.2 Brpop

BRPOP, ilkel bir engelleme listesi popülasyonudur. RPOP'un engelleyen sürümüdür çünkü verilen listelerin hiçbirinden çıkacak öğe olmadığında bağlantıyı engeller. İlk listenin kuyruğundan boş olmayan bir öğe çıkarılır ve verilen anahtarlar verildikleri sırayla kontrol edilir.Tam anlamlar için BLPOP belgelerine bakın, çünkü BRPOP, BLPOP ile aynıdır, tek fark, öğeleri baştan çıkarmak yerine bir listenin kuyruğundan çıkarmasıdır.

4.3 Brpopplush

BRPOPLPUSH, RPOPLPUSH'un engelleme varyantıdır. Kaynak elemanlar içerdiğinde, bu komut tam olarak RPOPLPUSH gibi davranır. Bir MULTI / EXEC bloğu içinde kullanıldığında, bu komut tam olarak RPOPLPUSH gibi davranır. Kaynak boş olduğunda Redis, başka bir istemci kendisine itene kadar veya zaman aşımına ulaşılana kadar bağlantıyı engeller. Süresiz olarak engellemek için sıfır zaman aşımı kullanılabilir.

Lindex : Anahtarda saklanan listedeki dizin dizinindeki öğeyi döndürür. Dizin sıfır tabanlıdır, bu nedenle 0 ilk öğe, 1 ikinci öğe vb. Anlamına gelir. Listenin sonundan başlayarak öğeleri belirlemek için negatif indeksler kullanılabilir. Burada -1, son öğe anlamına gelir; -2, sondan bir önceki anlamına gelir ve diğerleri.

```
127.0.0.1:6379> LPUSH list1 "Yaren"
(integer) 3
127.0.0.1:6379> LPUSH list2 "Unburak"
(integer) 1
127.0.0.1:6379> LINDEX list1 0
"Yaren"
127.0.0.1:6379> LINDEX list2 -3
(nil)
127.0.0.1:6379> LINDEX list2 1
(nil)
127.0.0.1:6379> LINDEX list2 3
(nil)
127.0.0.1:6379>
```

4.4 Linsert

Referans değeri pivotundan önce veya sonra anahtarda depolanan listeye öğe ekler. Anahtar olmadığında, boş bir liste olarak kabul edilir ve hiçbir işlem yapılmaz. Anahtar varsa ancak bir liste değerini tutmadığında bir hata döndürülür.

```
127.0.0.1:6379> RPUSH myList "Hello"
(integer) 1
127.0.0.1:6379> RPUSH myList "World"
(integer) 2
127.0.0.1:6379> LINSERT myList BEFORE "World" "There"
(integer) 3
127.0.0.1:6379> LRANGE myList 0 -1
1) "Hello"
2) "There"
3) "World"
127.0.0.1:6379>
```

4.5 Llen

Anahtarda saklanan listenin uzunluğunu döndürür. Anahtar yoksa, boş bir liste olarak yorumlanır ve 0 döndürülür. Anahtarda saklanan değer bir liste olmadığında bir hata döndürülür.

```
127.0.0.1:6379> LPUSH myList "World"
(integer) 4
127.0.0.1:6379> LPUSH myList "Hello"
(integer) 5
127.0.0.1:6379> LLEN myList
(integer) 5
127.0.0.1:6379>
```

4.6 Lmove

Kaynakta depolanan listenin ilk / son öğesini (nereden argümanına bağlı olarak başlık / kuyruk) atomik olarak döndürür ve kaldırır ve depolanan listenin ilk / son öğesindeki (nereye argümanına bağlı olarak başlık / kuyruk) öğeyi iter varış noktasında.

4.7 Lpop

Anahtarda saklanan listenin ilk elemanlarını kaldırır ve döndürür.

```
127.0.0.1:6379> RPUSH mylist "one"
(integer) 1
127.0.0.1:6379> RPUDH mylist "two"
(error) ERR unknown command 'RPUDH'
127.0.0.1:6379> RPUSH mylist "three
Invalid argument(s)
127.0.0.1:6379>  RPUSH mylist "three"
(integer) 2
127.0.0.1:6379> LPOP mylist
"one"
127.0.0.1:6379> LRANGE mylist 0 -1
1) "three"
127.0.0.1:6379>
```

4.8 Lpos

Komut, bir Redis listesindeki eşleşen öğelerin dizinini döndürür. Varsayılan olarak, hiçbir seçenek belirtilmediğinde, listeyi baştan sona tarayacak ve "element" in ilk eşleşmesini arayacaktır. Öğe bulunursa, dizini (listedeki sıfır tabanlı konum) döndürülür. Aksi takdirde, eşleşme bulunmazsa NULL döndürülür.

```
127.0.0.1:6379> RPUSH mylist a b c 1 2 3 4 5 6  yaren unburak ozgur
(integer) 37
127.0.0.1:6379> LPOS mylist c
(error) ERR unknown command 'LPOS'
127.0.0.1:6379> LPOS mylist 3
(error) ERR unknown command 'LPOS'
127.0.0.1:6379>
```

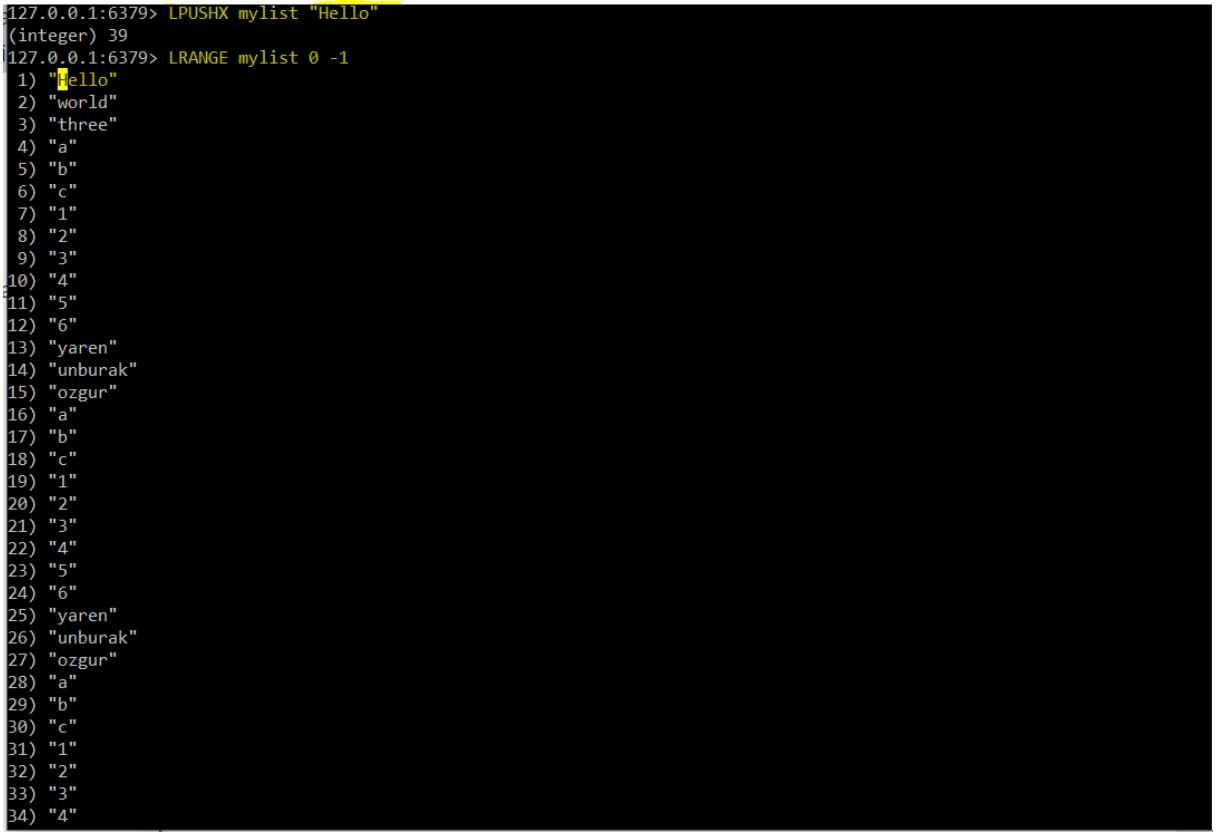
4.9 Lpush

Anahtarda saklanan listenin başına belirtilen tüm değerleri girin. Anahtar yoksa, push işlemlerini gerçekleştirmeden önce boş liste olarak oluşturulur. Anahtar, liste olmayan bir değeri tuttuğunda, bir hata döndürülür.

```
127.0.0.1:6379> LPUSH mylist "world"
(integer) 38
127.0.0.1:6379> LRANGE mylist 0 -1
 1) "world"
 2) "three"
 3) "a"
 4) "b"
 5) "c"
 6) "1"
 7) "2"
 8) "3"
 9) "4"
10) "5"
11) "6"
12) "yaren"
13) "unburak"
14) "ozgur"
15) "a"
16) "b"
17) "c"
18) "1"
19) "2"
20) "3"
21) "4"
22) "5"
23) "6"
24) "yaren"
25) "unburak"
26) "ozgur"
27) "a"
28) "b"
29) "c"
30) "1"
31) "2"
32) "3"
33) "4"
34) "5"
```

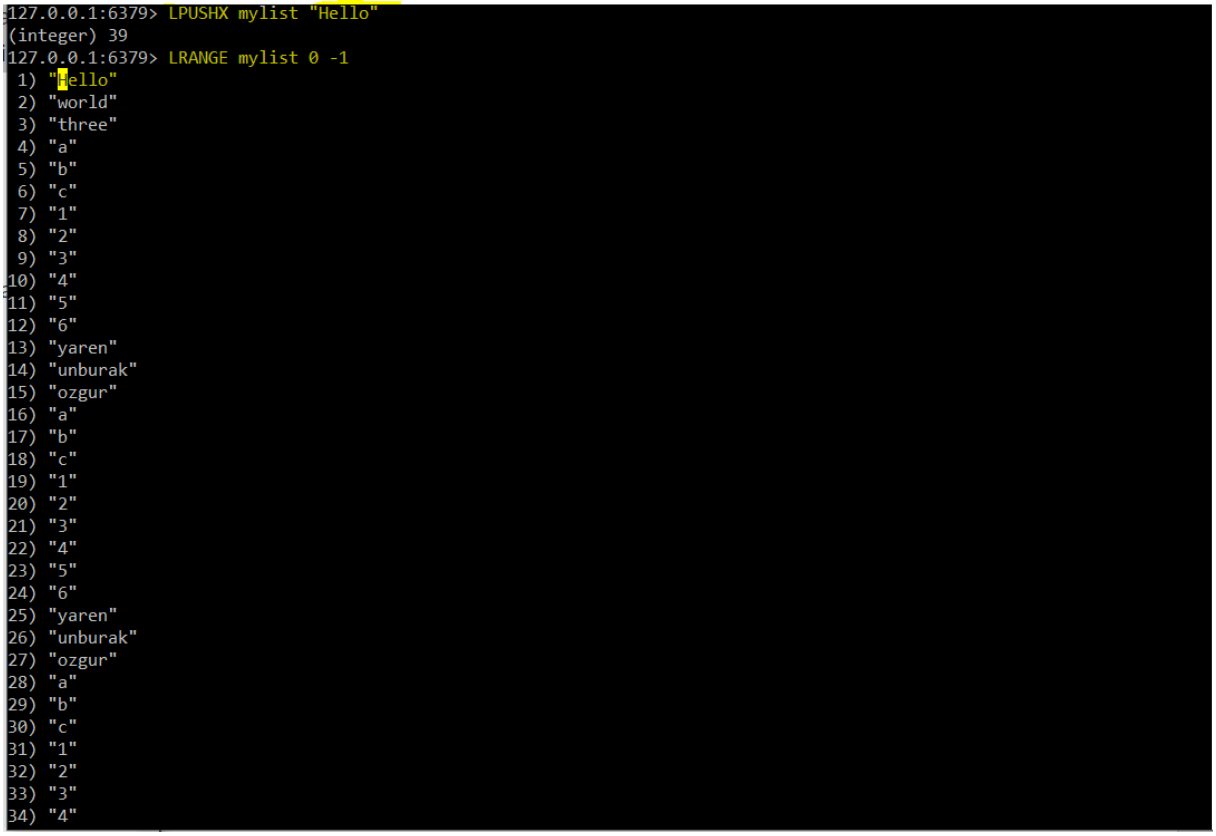
4.10 Lpushx

Yalnızca anahtar zaten varsa ve bir liste tutuyorsa, anahtarda depolanan listenin başına belirtilen değerleri ekler. LPUSH'nin aksine, henüz anahtar olmadığında hiçbir işlem yapılmayacaktır.



4.11 Lrange

Anahtarda depolanan listenin belirtilen öğelerini döndürür. Ofsetler başlangıç ve bitiş sıfır tabanlı indekslerdir; 0, listenin ilk öğesi (listenin başı), 1 sonraki öğe vb.



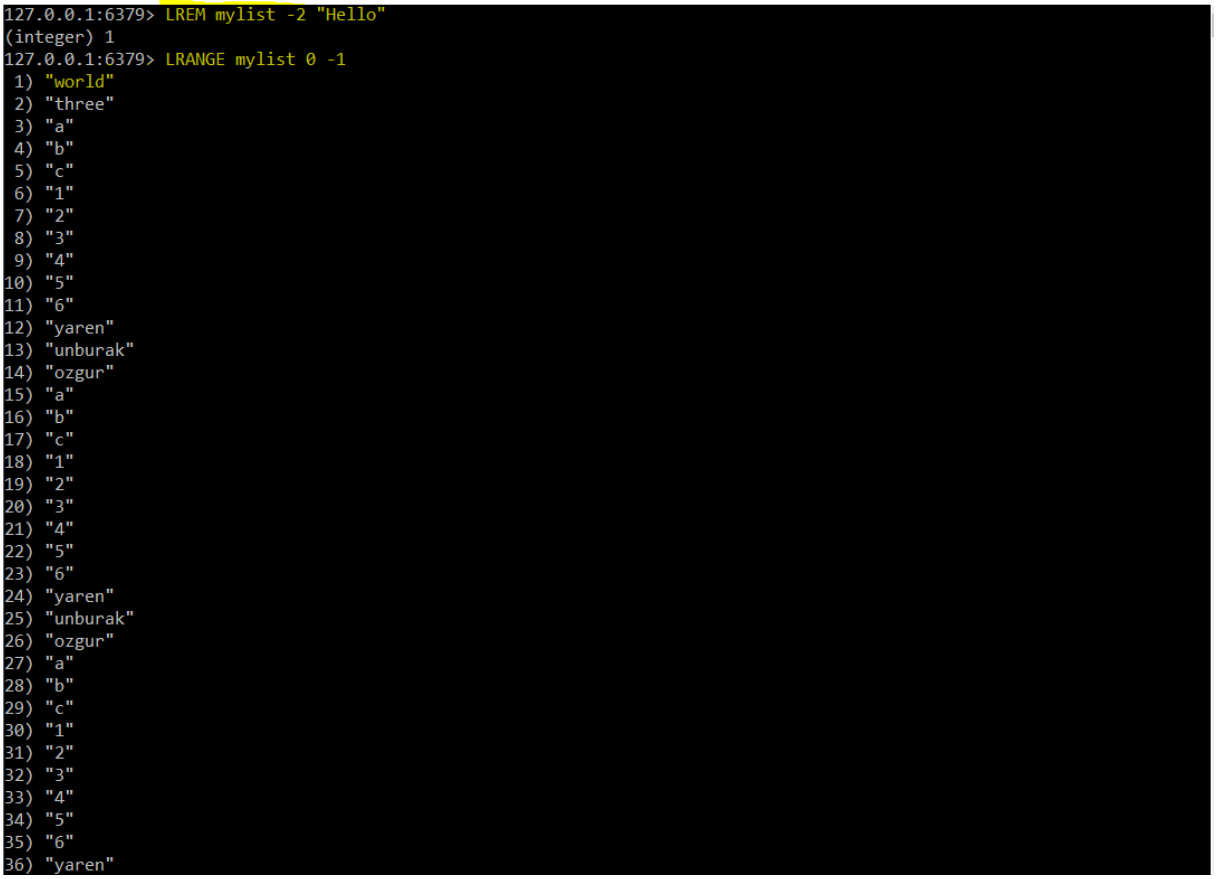
4.12 Lrem

Anahtarda depolanan listeden öğeye eşit öğelerin ilk sayım oluşumlarını kaldırır. Count argümanı işlemi aşağıdaki şekillerde etkiler.

count> 0: Baştan kuyruğa hareket eden öğeye eşit öğeleri kaldırır.

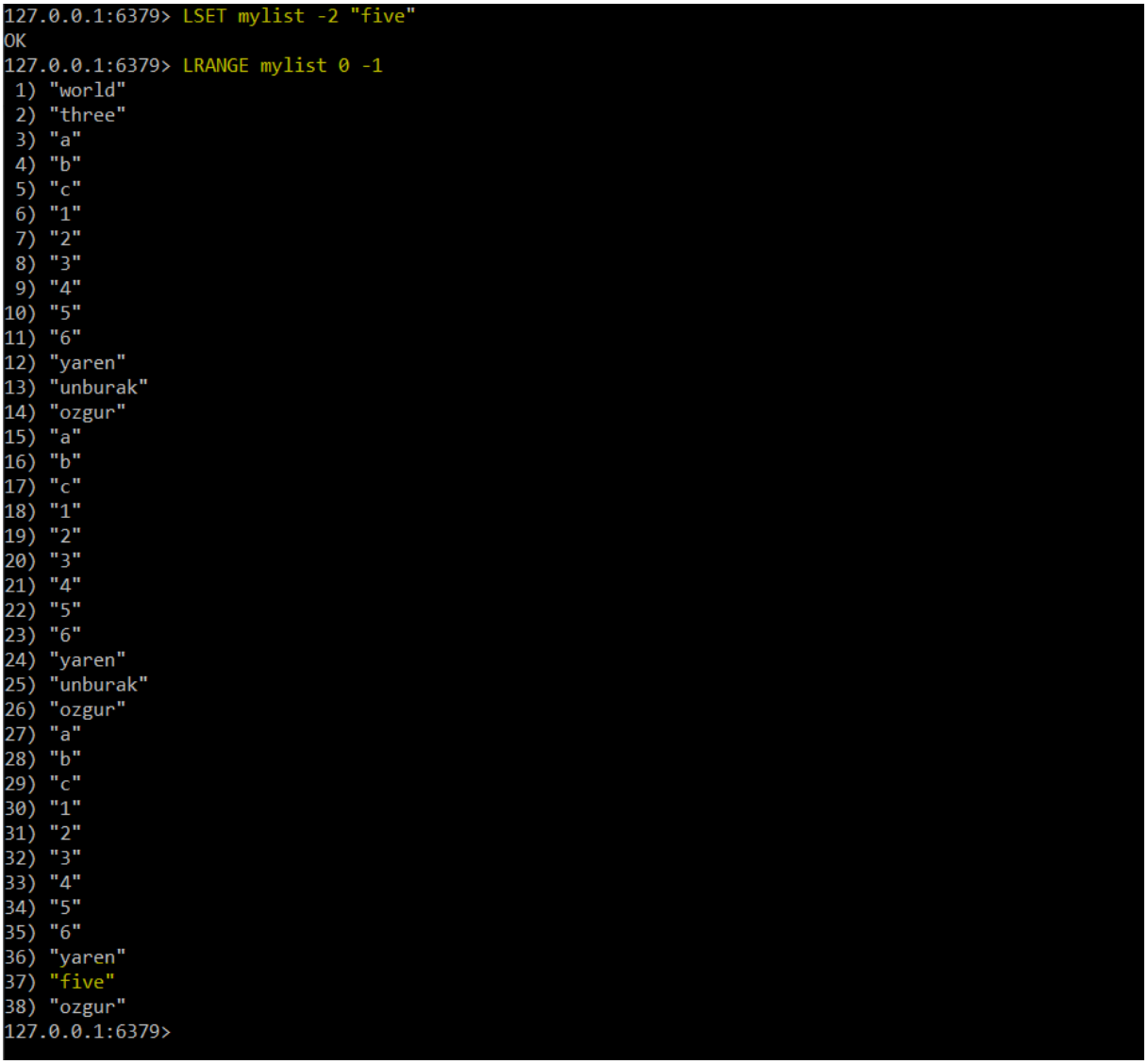
count <0: Kuyruktan başa hareket eden öğeye eşit öğeleri kaldırır.

count = 0: öğeye eşit tüm öğeleri kaldırır.



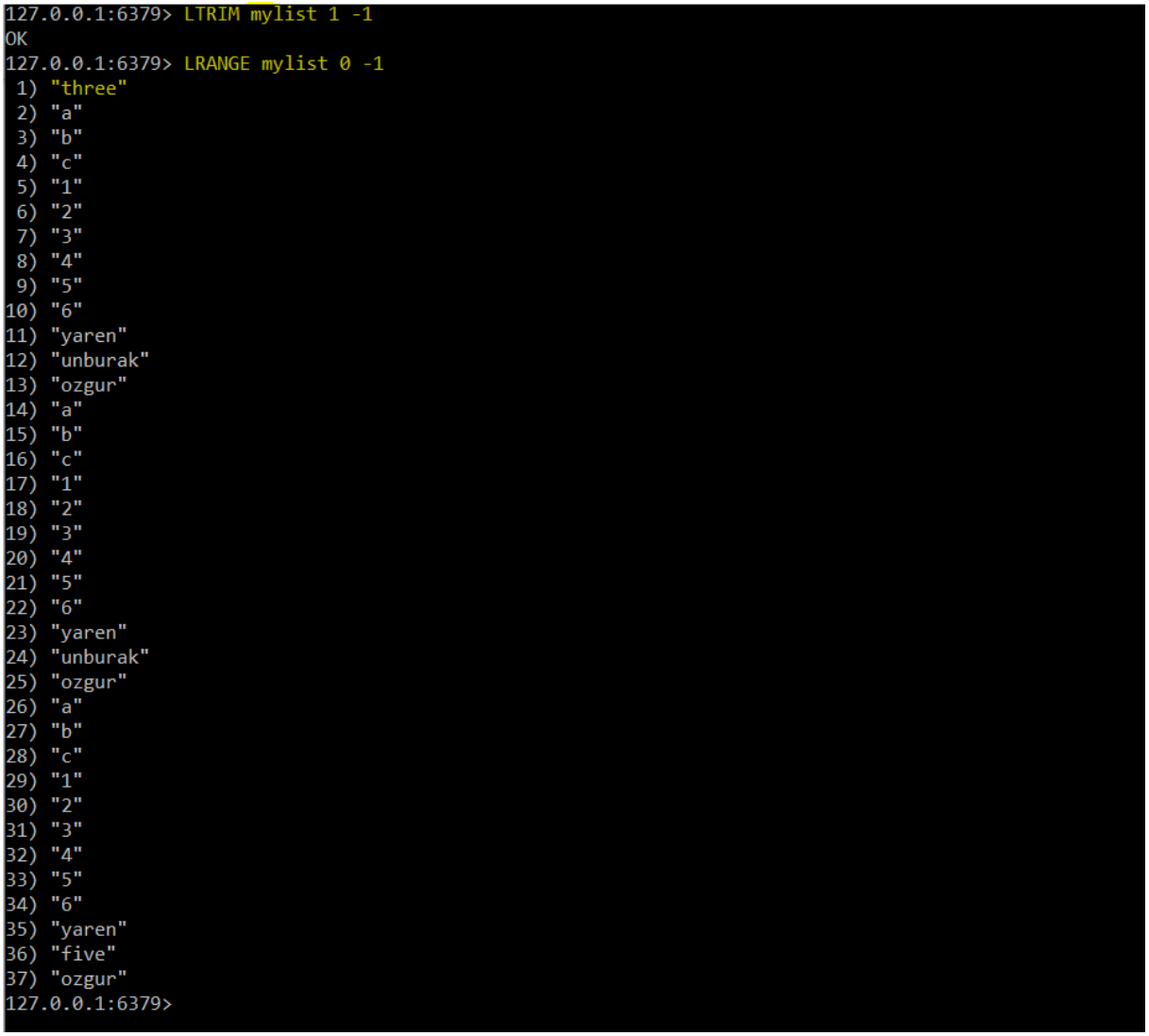
4.13 Lset

Dizindeki liste öğesini öğeye ayarlar.



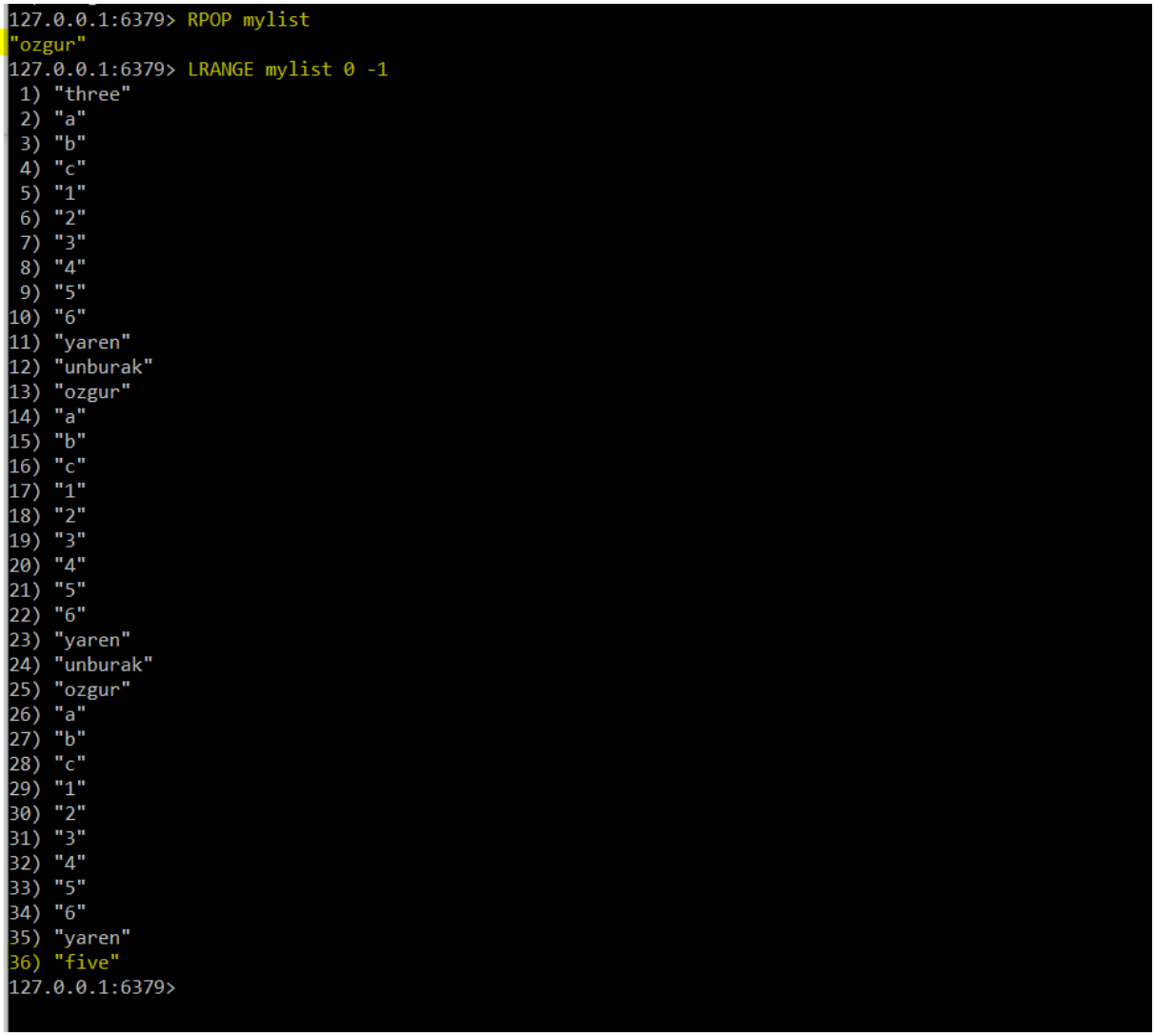
4.14 Ltrim

Mevcut bir listeyi, yalnızca belirtilen öğe aralığını içerecek şekilde kırpın. Hem başlangıç hem de bitiş sıfır tabanlı dizinlerdir; burada 0, listenin ilk öğesi (baş), 1 sonraki öğe vb.



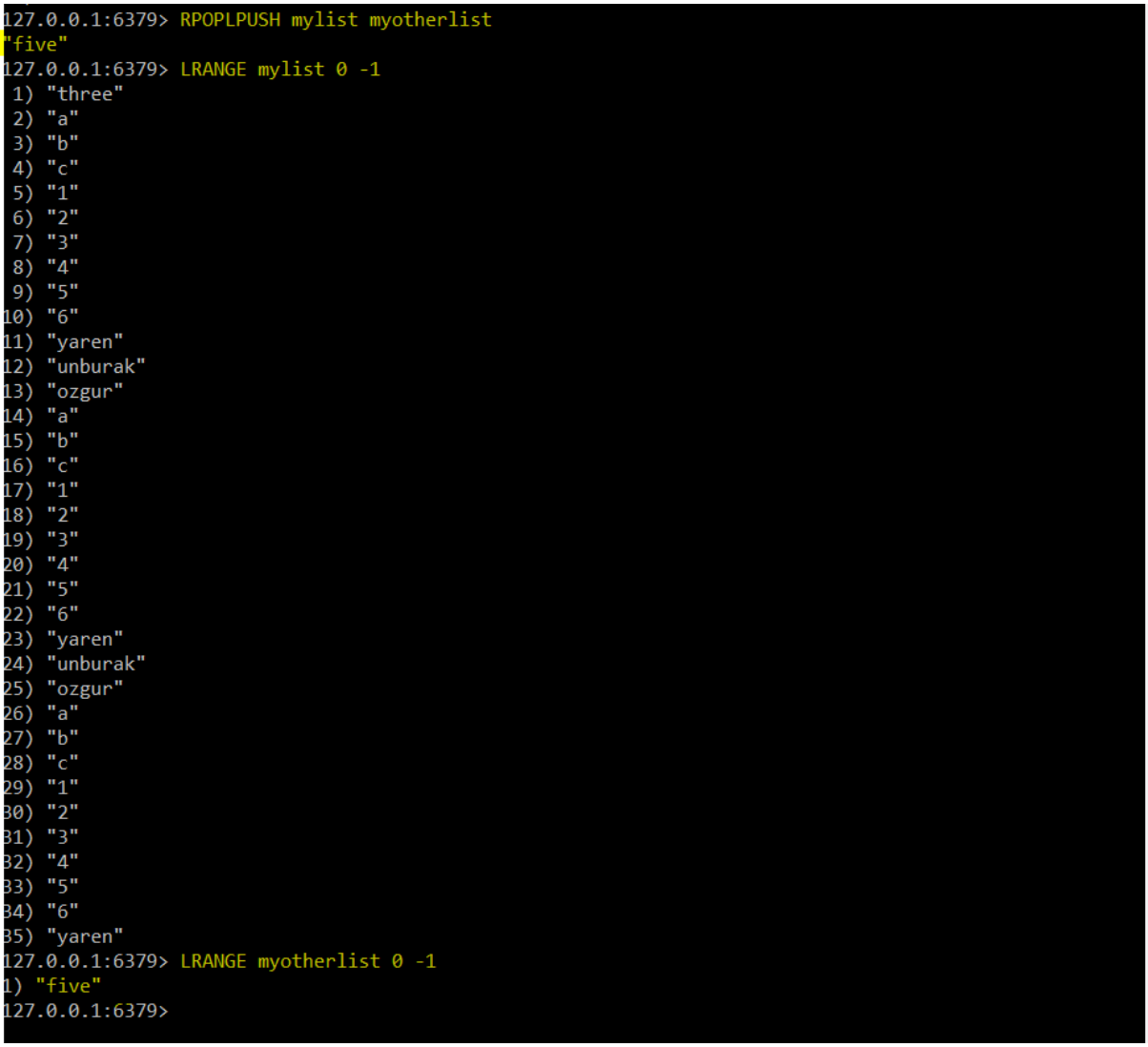
4.15 Rpop

Anahtarda saklanan listenin son elemanlarını kaldırır ve geri döndürür.



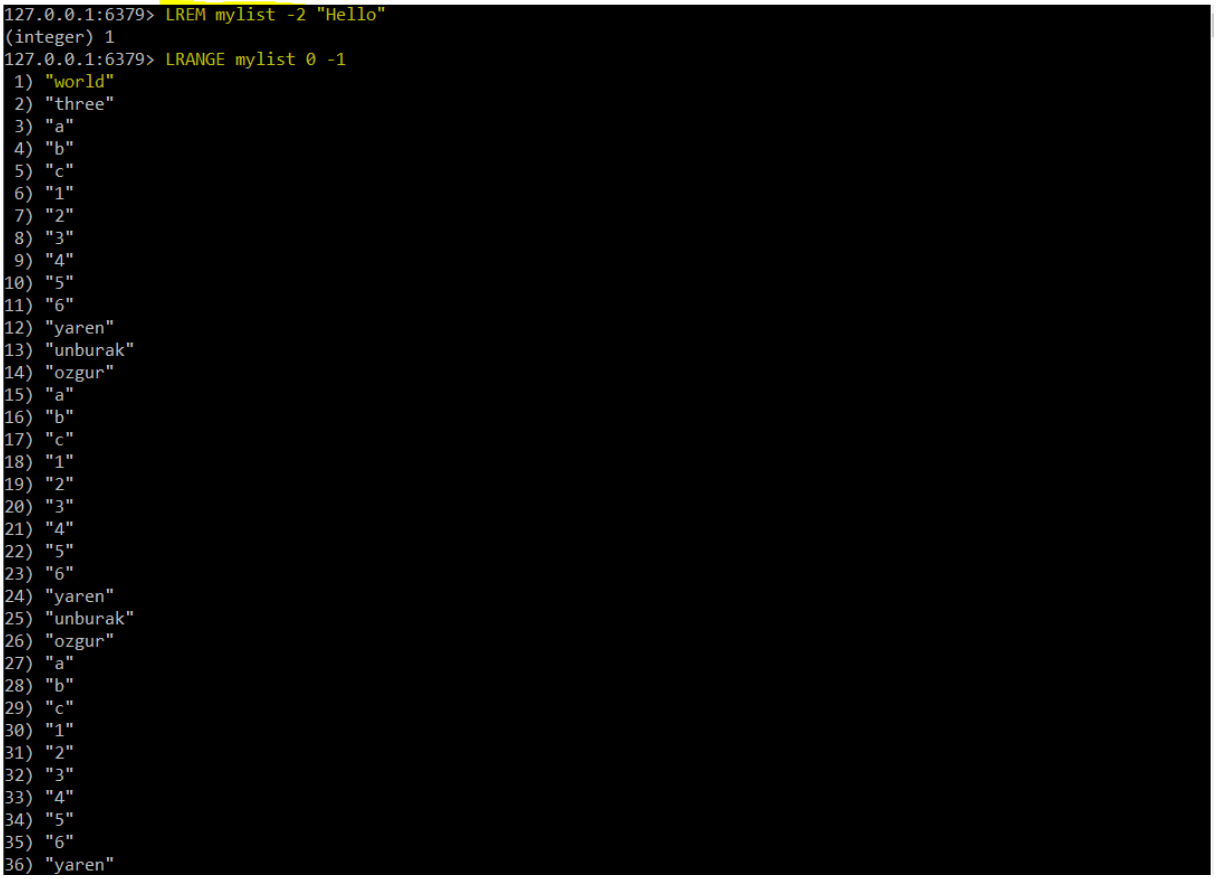
4.16 Rpoplpush

Kaynakta depolanan listenin son öğesini (kuyruk) atomik olarak döndürür ve kaldırır ve öğeyi hedefte depolanan listenin ilk öğesinde (baş) iter.



4.17 Rpush

Anahtarda depolanan listenin sonuna belirtilen tüm değerleri girin. Anahtar yoksa, push işlemini gerçekleştirmeden önce boş liste olarak oluşturulur. Anahtar, liste olmayan bir değeri tuttuğunda, bir hata döndürülür.



4.18 Rpushx

Yalnızca anahtar zaten varsa ve bir liste tutuyorsa, anahtarda depolanan listenin sonuna belirtilen değerleri ekler.

```
127.0.0.1:6379> RPUSHX mylist "Yaren"
(integer) 38
127.0.0.1:6379> RPUSHX mylist "Unburak"
(integer) 39
127.0.0.1:6379> LRANGE mylist 0 -1
 1) "three"
 2) "a"
 3) "b"
 4) "c"
 5) "1"
 6) "2"
 7) "3"
 8) "4"
 9) "5"
10) "6"
11) "yaren"
12) "unburak"
13) "ozgur"
14) "a"
15) "b"
16) "c"
17) "1"
18) "2"
19) "3"
20) "4"
21) "5"
22) "6"
23) "yaren"
24) "unburak"
25) "ozgur"
26) "a"
27) "b"
28) "c"
29) "1"
30) "2"
31) "3"
32) "4"
33) "5"
34) "6"
35) "yaren"
36) "Hello"
37) "World"
38) "Yaren"
39) "Unburak"
127.0.0.1:6379>
```

5 SETS Veri Tipi

5.1 Sadd

Belirtilen üyeleri anahtarda depolanan kümeye ekleyin. Zaten bu grubun üyesi olan belirtilen üyeler göz ardı edilir. Anahtar yoksa, belirtilen üyeleri eklemeden önce yeni bir küme oluşturulur.

```
127.0.0.1:6379> SADD myset "Hello"
(integer) 1
127.0.0.1:6379> SADD myset "Yaren"
(integer) 1
127.0.0.1:6379> SADD myset "Unburak"
(integer) 1
127.0.0.1:6379> SMEMBERS myset
1) "Hello"
2) "Unburak"
3) "Yaren"
127.0.0.1:6379>
```

5.2 Scard

Anahtarda depolanan kümenin ayarlanan önemini (öğe sayısı) döndürür.

```
127.0.0.1:6379> SADD myset "Hello"
(integer) 1
127.0.0.1:6379> SADD myset "Yaren"
(integer) 1
127.0.0.1:6379> SADD myset "Unburak"
(integer) 1
127.0.0.1:6379> SMEMBERS myset
1) "Hello"
2) "Unburak"
3) "Yaren"
127.0.0.1:6379> SCARD myset
(integer) 3
127.0.0.1:6379>
```

5.3 Sdiff

İlk küme ile tüm ardışık kümeler arasındaki farktan kaynaklanan kümenin üyelerini verir.

```
127.0.0.1:6379> SDIFF myset "e"
1) "Hello"
2) "Yaren"
3) "Unburak"
```

5.4 Sdiffstore

Bu komut TMSF'ye eşittir ancak elde edilen setin geri verilmesi yerine hedefte saklanır.

```
127.0.0.1:6379> SADD key1 "a"
(error) WRONGTYPE Operation against a key holding the wrong kind of value
127.0.0.1:6379> SADD key1 a
(error) WRONGTYPE Operation against a key holding the wrong kind of value
127.0.0.1:6379> SADD new1 "a"
(integer) 1
127.0.0.1:6379> SADD new1 "b"
(integer) 1
127.0.0.1:6379> SADD new1 "c"
(integer) 1
127.0.0.1:6379> SADD new2 "c"
(integer) 1
127.0.0.1:6379> SADD new2 "d"
(integer) 1
127.0.0.1:6379> SADD new2 "e"
(integer) 1
127.0.0.1:6379> SDIFFSTORE key new1 new2
(integer) 2
127.0.0.1:6379> SMEMBERS key
1) "b"
2) "a"
127.0.0.1:6379>
```

5.5 Sinter

Verilen tüm kümelerin kesişiminden kaynaklanan kümenin üyelerini döndürür.

```
127.0.0.1:6379> SADD key1 "a"
(integer) 1
127.0.0.1:6379> SADD key1 "b"
(integer) 1
127.0.0.1:6379> SADD key2 "c"
(integer) 1
127.0.0.1:6379> SADD key2 "d"
(integer) 1
127.0.0.1:6379> SADD key2 "e"
(integer) 1
127.0.0.1:6379> SADD key1 "c"
(integer) 1
127.0.0.1:6379> SINTER key1 key2
1) "c"
127.0.0.1:6379>
```

5.6 Sinterstore

Bu komut SINTER'a eşittir, ancak sonuç kümesini döndürmek yerine hedefte saklanır.

```
127.0.0.1:6379> SADD key1 "a"
(integer) 1
127.0.0.1:6379> SADD key1 "b"
(integer) 1
127.0.0.1:6379> SADD key2 "c"
(integer) 1
127.0.0.1:6379> SADD key2 "d"
(integer) 1
127.0.0.1:6379> SADD key2 "e"
(integer) 1
127.0.0.1:6379> SADD key1 "c"
(integer) 1
127.0.0.1:6379> SINTER key1 key2
1) "c"
127.0.0.1:6379> SINTERSTORE key key1 key2
(integer) 1
127.0.0.1:6379> SMEMBERS key
1) "c"
127.0.0.1:6379>
```

5.7 Sismember

Üye, anahtarda depolanan kümenin bir üyesi ise döndürür.

5.8 Smembers

Anahtarda saklanan ayar değerinin tüm üyelerini döndürür.

5.9 Smove

Üyeyi kaynaktaki kümeden hedefteki kümeye taşıyın. Bu işlem atomiktir. Her verilen anda, öğe diğer istemciler için kaynağın veya hedefin bir üyesi olarak görünecektir.

5.10 Spop

Anahtardaki ayar değeri deposundan bir veya daha fazla rastgele üye kaldırır ve döndürür.

5.11 Srandmember

Yalnızca anahtar bağımsız değişkeniyle çağrıldığında, anahtarda depolanan ayar değerinden rastgele bir öğe döndürür.

```
127.0.0.1:6379> SRANDMEMBER myset -5
1) "Hello"
2) "Unburak"
3) "Unburak"
4) "Hello"
5) "Hello"
127.0.0.1:6379>
```

5.12 Sscan

Scan işlemi ile aynı mantıkta çalışır

5.13 Sunion

Verilen tüm kümelerin birleşiminden kaynaklanan kümenin üyelerini döndürür.

```
127.0.0.1:6379> SUNION key1 key2
1) "e"
2) "d"
3) "c"
```

5.14 Sunionstore

Bu komut SUNION komutuna eşittir, ancak sonuç kümesini geri döndürmek yerine hedefte saklanır.Hedef zaten varsa, üzerine yazılır.

```
127.0.0.1:6379> SUNIONSTORE key key1 key2
(integer) 3
127.0.0.1:6379> SMEMBERS key
1) "e"
2) "d"
3) "c"
127.0.0.1:6379>
```

5.15 Srem

Belirtilen üyeleri anahtarda depolanan kümeden kaldırın. Bu grubun üyesi olmayan belirtilen üyeler göz ardı edilir. Anahtar yoksa, boş bir küme olarak kabul edilir ve bu komut 0 döndürür.

```
127.0.0.1:6379> SREM key1 "three"
(integer) 0
127.0.0.1:6379> SREM key2 "four"
(integer) 0
127.0.0.1:6379> SMEMBERS key2
1) "e"
2) "d"
3) "c"
127.0.0.1:6379>
```

5.16 Smismember

Her üyenin anahtarda depolanan kümenin üyesi olup olmadığını döndürür.

```
redis> SADD myset "one"
(integer) 1
redis> SADD myset "one"
(integer) 0
redis> SMISMEMBER myset "one" "notamember"
1) (integer) 1
2) (integer) 0
redis>
```

6 SCRIPT Yapısı

6.1 Eval

EVAL'ın ilk argümanı bir Lua 5.1 betiğidir. Komut dosyasının bir Lua işlevi tanımlaması gerekmez (ve olmamalıdır). Redis sunucusu bağlamında çalışacak bir Lua programıdır.EVAL'ın ikinci argümanı, Redis anahtar adlarını temsil eden komut dosyasını (üçüncü argümandan başlayarak) izleyen argümanların sayısıdır. Argümanlara Lua tarafından tek tabanlı bir dizi biçiminde KEYS global değişkeni kullanılarak erişilebilir (yani KEYS [1], KEYS [2], ...).Tüm ek argümanlar anahtar adlarını temsil etmemelidir ve Lua tarafından, anahtarlarla olana çok benzer şekilde ARGV global değişkeni kullanılarak erişilebilir (yani ARGV [1], ARGV [2], ...)

```
C:\Users\yaren>redis-cli
127.0.0.1:6379> eval "return redis.call('set','foo','bar')" 0
OK
127.0.0.1:6379>
```

6.2 EvalSHA

Sunucu tarafında SHA1 özeti tarafından önbelleğe alınan bir komut dosyasını değerlendirir. Komut dosyaları, SCRIPT LOAD komutu kullanılarak sunucu tarafında önbelleğe alınır. Aksi takdirde komut EVAL ile aynıdır.

6.3 ScriptDebug

EVAL ile yürütülen sonraki komut dosyaları için hata ayıklama modunu ayarlayın. Redis, karmaşık komut dosyaları yazma görevini çok daha basit hale getirmek için kullanılabilen eksiksiz bir Lua hata ayıklayıcı, kod adı LDB içerir. Hata ayıklama modunda Redis, uzaktan hata ayıklama sunucusu görevi görür ve redis-cli gibi bir istemci komut dosyalarını adım adım çalıştırabilir, kesme noktaları ayarlayabilir, değişkenleri inceleyebilir ve daha fazlasını yapabilir - LDB hakkında ek bilgi için Redis Lua hata ayıklayıcı sayfasına bakın.

6.4 ScriptExists

Komut dosyası önbelleğindeki komut dosyalarının varlığı hakkında bilgi verir.

6.5 ScriptFlush

Lua komut dosyası önbelleğini temizler.

6.6 ScriptKill

Betik tarafından henüz bir yazma işlemi gerçekleştirilmediğini varsayarak, şu anda çalıştırılan Lua betiğini öldürür.

6.7 ScriptLoad

Komut dosyası önbelleğine, yürütmeden bir komut dosyası yükleyin. Belirtilen komut betik önbelleğine yüklendikten sonra, EVAL'ın ilk başarılı çağrılmasından sonra olduğu gibi, komut dosyasının doğru SHA1 özetiyle EVALSHA kullanılarak çağrılabilir olacaktır.

7 Connection (Client/Server) Yapıları

7.1 Auth

Auth komutu mevcut bağlantının kimliğini iki durumda doğrular: Redis sunucusu, Requepass seçeneği aracılığıyla parola korumalıysa. Redis 6.0 örneği veya üstü Redis ACL sistemini kullanıyorsa. Redis 6'dan önceki Redis sürümleri, komutun yalnızca bir bağımsız değişken sürümünü anlayabiliyordu:

```
AUTH <username> <password>
```

7.2 Echo

Mesaj döndürür.

```
127.0.0.1:6379> ECHO "Merhaba Yaren ve Unburak"
"Merhaba Yaren ve Unburak"
127.0.0.1:6379>
```

7.3 Hello

Bağlantıyı farklı bir protokole geçirir.

```
127.0.0.1:6379> HELLO 3
1# "server" => "redis"
2# "version" => "6.0.9"
3# "proto" => (integer) 3
4# "id" => (integer) 63
5# "mode" => "standalone"
6# "role" => "master"
7# "modules" => (empty array)
```

7.4 Ping

Bağımsız değişken sağlanmadıysa PONG döndürür, aksi takdirde bağımsız değişkenin bir kopyasını toplu olarak döndürür. Bu komut genellikle bir bağlantının hala canlı olup olmadığını test etmek veya gecikmeyi ölçmek için kullanılır.

```
127.0.0.1:6379> PING
PONG
127.0.0.1:6379> PING "Yaren"
"Yaren"
127.0.0.1:6379> PING "Unburak"
"Unburak"
127.0.0.1:6379>
```

7.5 Quit

Sunucudan bağlantıyı kapatmasını isteyin. Bekleyen tüm yanıtlar istemciye yazılır yazılmaz bağlantı kapatılır.

7.6 Reset

Bu komut, bağlantının kesilmesi ve yeniden bağlanmanın etkisini taklit ederek, bağlantının sunucu tarafı bağlamında tam bir sıfırlama gerçekleştirir.

7.7 Select

Belirtilen sıfır tabanlı sayısal dizine sahip Redis mantıksal veritabanını seçin. Yeni bağlantılar her zaman 0 veritabanını kullanır.

8 Keys Yapısı İçin Gerekli Komutlar

8.1 Copy

Bu komut, kaynak anahtarda depolanan değeri hedef anahtara kopyalar.

8.2 Del

Belirtilen anahtarları kaldırır. Anahtar yoksa, yok sayılır.

```
127.0.0.1:6379> SET key1 "Hello"
OK
127.0.0.1:6379> SET key2 "Dolly"
OK
127.0.0.1:6379> DEL key1 key2
(integer) 2
127.0.0.1:6379>
```

8.3 Dump

Anahtarda depolanan değeri Redis'e özgü bir biçimde seri hale getirin ve kullanıcıya iade edin. Döndürülen değer, RESTORE komutu kullanılarak bir Redis anahtarına geri sentezlenebilir.

```
127.0.0.1:6379> SET goat "Dolly"
OK
127.0.0.1:6379> DUMP goat
"\x00\x05Dolly\a\x00/\x04\xea\xb44*\x96\xfe"
127.0.0.1:6379>
```

8.4 Exists

Anahtar varsa döner.

```
127.0.0.1:6379> EXISTS goat
(integer) 1
127.0.0.1:6379> SET yaren "YAREN"
OK
127.0.0.1:6379> EXISTS yaren
(integer) 1
127.0.0.1:6379>
```

8.5 Expire

Anahtar için bir zaman aşımı ayarlayın. Zaman aşımı süresi dolduktan sonra, anahtar otomatik olarak silinecektir. Redis terminolojisinde ilişkili bir zaman aşımına sahip bir anahtarın genellikle geçici olduğu söylenir.

```
127.0.0.1:6379> SET unburak "Unburak"
OK
127.0.0.1:6379> EXPIRE unburak 10
(integer) 1
127.0.0.1:6379> TTL unburak
(integer) 6
127.0.0.1:6379> SET yaren "Yaren"
OK
127.0.0.1:6379> TTL yaren
(integer) -1
127.0.0.1:6379>
```

8.6 Expireat

EXPIREAT, EXPIRE ile aynı etkiye ve anlamsallığa sahiptir, ancak TTL'yi (yaşam süresi) temsil eden saniye sayısını belirtmek yerine, mutlak bir Unix zaman damgası alır (1 Ocak 1970'den bu yana saniye). Geçmişteki bir zaman damgası, anahtarı hemen siler.

```
127.0.0.1:6379> EXPIREAT yaren 123456789
(integer) 1
127.0.0.1:6379> EXISTS yaren
(integer) 0
127.0.0.1:6379>
```

8.7 Keys

Kalıpla eşleşen tüm anahtarları döndürür.Bu işlem için zaman karmaşıklığı O (N) iken, sabit zamanlar oldukça düşüktür. Örneğin, giriş seviyesi bir dizüstü bilgisayarda çalışan Redis, 40 milisaniyede 1 milyon anahtar veritabanını tarayabilir.

```
127.0.0.1:6379> MSET firstname Yaren lastname Gunduz age 22
OK
127.0.0.1:6379> KEYS *name*
1) "firstname"
2) "lastname"
127.0.0.1:6379> KEYS a??
1) "age"
127.0.0.1:6379> KEYS *
 1) "list2"
 2) "foo"
 3) "myList"
 4) "goat"
 5) "anahtar"
 6) "firstname"
 7) "myhash"
 8) "lastname"
 9) "cumle"
10) "mylist"
11) "age"
12) "myset"
13) "intanahtar"
14) "myotherlist"
15) "YAREN"
16) "api/city"
17) "yaren1"
18) "key"
19) "users"
20) "damla"
21) "dolly"
22) "yas"
23) "list1"
24) "burak"
25) "burak1"
26) "3"
27) "dest"
28) "hello"
127.0.0.1:6379>
```

8.8 Migrate

Bir anahtarı kaynak Redis örneğinden hedef Redis örneğine atomik olarak aktarın. Başarı durumunda anahtar orijinal örnekten silinir ve hedef örnekte varlığı garanti edilir.

8.9 Move

Anahtarı şu anda seçili veritabanından (bkz. SEÇ) belirtilen hedef veritabanına taşı. Anahtar hedef veritabanında zaten mevcutsa veya kaynak veritabanında yoksa, hiçbir şey yapmaz. Bundan dolayı, MOVE'u kilitleme ilkeli olarak kullanmak mümkündür.Eğer key taşınırsa 1,taşınmazsa 0 döner.

8.10 Object

OBJECT komutu, tuşlarla ilişkili Redis Nesnelerinin iç kısımlarının incelenmesine izin verir. Hata ayıklama veya anahtarlarınızın yerden tasarruf etmek için özel olarak kodlanmış veri türlerini kullanıp kullanmadığını anlamak için kullanışlıdır.

```
127.0.0.1:6379> SET foo 1000
OK
127.0.0.1:6379> object encoding foo
"int"
127.0.0.1:6379> append foo bar
(integer) 7
127.0.0.1:6379> get foo
"1000bar"
127.0.0.1:6379> object encoding foo
"raw"
127.0.0.1:6379>
```

8.11 Persist

Anahtardaki mevcut zaman aşımını kaldırın, anahtarı geçici durumdan (sona erme setine sahip bir anahtar) kalıcı hale getirin (zaman aşımı ilişkili olmadığı için hiçbir zaman sona ermeyecek bir anahtar).

```
127.0.0.1:6379> SET yaren "YAREN"
OK
127.0.0.1:6379> EXPIRE yaren 10
(integer) 1
127.0.0.1:6379> TTL yaren
(integer) 6
127.0.0.1:6379> PERSIST yaren
(integer) 1
127.0.0.1:6379> TTL yaren
(integer) -1
127.0.0.1:6379>
```

8.12 Pexpire

Bu komut tam olarak EXPIRE gibi çalışır ancak anahtarın geçerlilik süresi saniye yerine milisaniye cinsinden belirtilir.

```
127.0.0.1:6379> SET unburak "Unburak"
OK
127.0.0.1:6379> PEXPIRE unburak 1534
(integer) 1
127.0.0.1:6379> TTL unburak
(integer) -2
127.0.0.1:6379> PTTL unburak
(integer) -2
127.0.0.1:6379>
```

8.13 Pexpireat

PEXPIREAT, EXPIREAT ile aynı etkiye ve anlamsallığa sahiptir, ancak anahtarın süresinin dolacağı Unix zamanı saniye yerine milisaniye cinsinden belirtilir.

```
127.0.0.1:6379> PEXPIREAT unburak 123445
(integer) 0
127.0.0.1:6379> TTL unburak
(integer) -2
127.0.0.1:6379> PTTL unburak
(integer) -2
127.0.0.1:6379>
```

8.14 Pttl

TTL gibi, bu komut, süresi dolan bir anahtarın kalan yaşam süresini döndürür; tek fark, TTL'nin kalan süreyi saniye cinsinden döndürmesi, PTTL'nin ise milisaniye cinsinden döndürmesidir. Redis 2.6 veya daha eski sürümlerde, anahtar yoksa veya anahtar mevcutsa ancak ilişkili bir süresi yoksa komut -1 değerini döndürür.

Redis 2.8 ile başlayarak, hata durumunda dönüş değeri değişti:

1-Anahtar yoksa komut -2 döndürür.

2-Anahtar varsa ancak ilişkili bir süresi yoksa komut -1 döndürür.

```
127.0.0.1:6379> EXPIRE unburak 2
(integer) 0
127.0.0.1:6379> PPTL unburak
(error) ERR unknown command 'PPTL'
127.0.0.1:6379> PTTL unburak
(integer) -2
127.0.0.1:6379>
```

8.15 RandomKey

Seçili veritabanından rastgele bir anahtar döndür.

Rename : Anahtarı yeni anahtar olarak yeniden adlandırır. Anahtar olmadığında hata verir.

```
127.0.0.1:6379> SET yaren "Yaren"
OK
127.0.0.1:6379> RENAME yaren unburak
OK
127.0.0.1:6379> GET unburak
"Yaren"
127.0.0.1:6379>
```

8.16 Renamenx

Henüz newkey yoksa anahtarı newkey olarak yeniden adlandırır. Anahtar olmadığında hata verir.

```
127.0.0.1:6379> SET unburak "Yaren"
OK
127.0.0.1:6379> SET yaren "Unburak"
OK
127.0.0.1:6379> RENAMEX yaren unburak
(error) ERR unknown command 'RENAMEX'
127.0.0.1:6379> RENAMENX yaren unburak
(integer) 0
127.0.0.1:6379> GET unburak
"Yaren"
127.0.0.1:6379>
```

8.17 Restore

Sağlanan serileştirilmiş değerin serisini kaldırarak elde edilen bir değerle ilişkili bir anahtar oluşturur (DUMP aracılığıyla elde edilir).

```
redis> DEL mykey
0
redis> RESTORE mykey 0 "\n\x17\x17\x00\x00\x00\x12\x00\x00\x00\
                        x00\xc0\x01\x00\x04\xc0\x02\x00\x04\xc0
                        xff\x04\x00u#<\xc0;.\xe9\xdd"
OK
redis> TYPE mykey
list
redis> LRANGE mykey 0 -1
1) "1"
2) "2"
3) "3"
```

8.18 Scan

SCAN komutu ve yakından ilişkili komutlar SSCAN, HSCAN ve ZSCAN, bir dizi öğe üzerinde aşamalı olarak yineleme yapmak için kullanılır.

```
127.0.0.1:6379> scan 0
1) "26"
2)  1) "list2"
    2) "yaren1"
    3) "firstname"
    4) "list1"
    5) "burak"
    6) "myotherlist"
    7) "yaren"
    8) "myList"
    9) "damla"
   10) "myset"
   11) "intanahtar"
127.0.0.1:6379>
```

8.19 Sort

Anahtar üzerinde liste, küme veya sıralı kümede bulunan öğeleri döndürür veya saklar. Varsayılan olarak sıralama sayısaldır ve öğeler, çift duyarlıklı kayan noktalı sayı olarak yorumlanan değerlerine göre karşılaştırılır.

```
SORT mylist
```

8.20 Touch

Bir anahtarın son erişim zamanını değiştirir. Anahtar yoksa, yok sayılır.

```
127.0.0.1:6379> SET burak "Unburak"
OK
127.0.0.1:6379> SET yaren "Yaren"
OK
127.0.0.1:6379> TOUCH burak yaren
(integer) 2
127.0.0.1:6379>
```

8.21 Ttl

Zaman aşımı olan bir anahtarın kalan yaşam süresini döndürür. Bu iç gözlem yeteneği, bir Redis istemcisinin belirli bir anahtarın kaç saniye veri kümesinin parçası olmaya devam edeceğini kontrol etmesini sağlar.Eğer key yoksa -2,var fakat sona ermesine bağlantısı yoksa -1 döner.

```
127.0.0.1:6379> SET unburak "Unburak"
OK
127.0.0.1:6379> EXPIRE unburak 10
(integer) 1
127.0.0.1:6379> TTL unburak
(integer) 6
127.0.0.1:6379>
```

8.22 Type

Anahtarda depolanan değerin türünün dize temsilini döndürür. Döndürülebilecek farklı türler şunlardır: string, list, set, zset, hash ve stream.

```
127.0.0.1:6379> SET nursel "Nursel"
OK
127.0.0.1:6379> LPUSH computer "CPU"
(integer) 1
127.0.0.1:6379> SADD IP "192.168.1.45"
(integer) 1
127.0.0.1:6379> TYPE nursel
string
```

8.23 Unlink

Bu komut DEL'e çok benzer: belirtilen anahtarları kaldırır. Tıpkı DEL gibi, bir anahtar yoksa yoksayılır.

```
redis> SET key1 "Hello"
"OK"
redis> SET key2 "World"
"OK"
redis> UNLINK key1 key2 key3
(integer) 2
redis>
```

8.24 Wait

Bu komut, önceki tüm yazma komutları başarıyla aktarılıncaya ve en azından belirtilen kopya sayısı kadar onaylanana kadar mevcut istemciyi engeller. Milisaniye cinsinden belirtilen zaman aşımına ulaşılırsa, belirtilen replika sayısına henüz ulaşılmamış olsa bile komut geri döner. Komut her zaman WAIT komutundan önce gönderilen yazma komutlarını kabul eden replikaların sayısını döndürür. belirtilen çoğaltma sayısına ulaşıldığında veya zaman aşımına ulaşıldığında.

```
127.0.0.1:6379> SET foo bar
OK
127.0.0.1:6379> WAIT 1 0
```

9 Sorted Sets Yapısı

9.1 Bzpopmin

Engelleme sürümüdür, çünkü verilen sıralı kümelerin hiçbirinden çıkacak üye olmadığında bağlantıyı engeller.

```
redis> DEL zset1 zset2
(integer) 0
redis> ZADD zset1 0 a 1 b 2 c
(integer) 3
redis> BZPOPMIN zset1 zset2 0
1) "zset1"
2) "a"
3) "0"
```

9.2 Bzpopmax

Engelleyen sürümdür çünkü verilen sıralanmış kümelerin hiçbirinden çıkacak üye olmadığında bağlantıyı engeller.

```
redis> DEL zset1 zset2
(integer) 0
redis> ZADD zset1 0 a 1 b 2 c
(integer) 3
redis> BZPOPMAX zset1 zset2 0
1) "zset1"
2) "c"
3) "2"
```

9.3 Zadd

Anahtarda depolanan sıralanmış kümeye, belirtilen puanlara sahip belirtilen tüm üyeleri ekler. Birden fazla puan / üye çifti belirtmek mümkündür.

```
127.0.0.1:6379> ZADD yarenset 1 "yaren"
(integer) 1
127.0.0.1:6379> ZRANGE yarenset 0 -1 WITHSCORES
1) "yaren"
2) "1"
127.0.0.1:6379>
```

9.4 Zcard

Anahtarda saklanan sıralı kümenin sıralı küme önemini (öğe sayısı) döndürür.

```
127.0.0.1:6379> ZCARD yarenset
(integer) 1
127.0.0.1:6379>
```

9.5 Zcount

Anahtar üzerindeki sıralı kümedeki öğe sayısını minimum ve maksimum arasında bir puanla döndürür.

```
127.0.0.1:6379> ZCOUNT yarenset -inf +inf
(integer) 1
127.0.0.1:6379>
```

9.6 Zdiff

Bu komut ZDIFFSTORE'a benzer, ancak sonuçta ortaya çıkan sıralı kümeyi depolamak yerine istemciye geri döndürülür.

```
redis> ZADD zset1 1 "one"
(integer) 1
redis> ZADD zset1 2 "two"
(integer) 1
redis> ZADD zset1 3 "three"
(integer) 1
redis> ZADD zset2 1 "one"
(integer) 1
redis> ZADD zset2 2 "two"
(integer) 1
redis> ZDIFF 2 zset1 zset2
1) "three"
redis> ZDIFF 2 zset1 zset2 WITHSCORES
1) "three"
2) "3"
redis>
```

9.7 Zdiffstore

Sıralanan ilk ve tüm ardışık giriş kümeleri arasındaki farkı hesaplar ve sonucu hedefte depolar. Toplam giriş anahtarı sayısı sayı tuşlarıyla belirtilir.

```
redis> ZADD zset1 1 "one"
(integer) 1
redis> ZADD zset1 2 "two"
(integer) 1
redis> ZADD zset1 3 "three"
(integer) 1
redis> ZADD zset2 1 "one"
(integer) 1
redis> ZADD zset2 2 "two"
(integer) 1
redis> ZDIFFSTORE out 2 zset1 zset2
(integer) 1
redis> ZRANGE out 0 -1 WITHSCORES
1) "three"
2) "3"
redis>
```

9.8 Zincrby

Anahtarda saklanan sıralanmış kümedeki üyenin puanını aşamalı olarak artırır.

```
127.0.0.1:6379> ZINCRBY yarenset 2 "One"
"2"
127.0.0.1:6379> ZRANGE yarenset 0 -1 WITHSCORES
1) "yaren"
2) "1"
3) "One"
4) "2"
127.0.0.1:6379>
```

9.9 Zinter

Bu komut ZINTERSTORE'a benzer, ancak sonuçta ortaya çıkan sıralı kümeyi saklamak yerine istemciye geri döndürülür.

```
redis> ZADD zset1 1 "one"
(integer) 1
redis> ZADD zset1 2 "two"
(integer) 1
redis> ZADD zset2 1 "one"
(integer) 1
redis> ZADD zset2 2 "two"
(integer) 1
redis> ZADD zset2 3 "three"
(integer) 1
redis> ZINTER 2 zset1 zset2
1) "one"
2) "two"
redis> ZINTER 2 zset1 zset2 WITHSCORES
1) "one"
2) "2"
3) "two"
4) "4"
redis>
```

9.10 Zinterscore

Belirtilen anahtarlar tarafından verilen sayı anahtarları sıralı kümelerin kesişimini hesaplar ve sonucu hedefte depolar.

```
redis> ZADD zset1 1 "one"
(integer) 1
redis> ZADD zset1 2 "two"
(integer) 1
redis> ZADD zset2 1 "one"
(integer) 1
redis> ZADD zset2 2 "two"
(integer) 1
redis> ZADD zset2 3 "three"
(integer) 1
redis> ZINTERSTORE out 2 zset1 zset2 WEIGHTS 2 3
(integer) 2
redis> ZRANGE out 0 -1 WITHSCORES
1) "one"
2) "5"
3) "two"
4) "10"
redis>
```

9.11 Zlexcount

Sıralı bir kümedeki tüm öğeler aynı puanla eklendiğinde, sözlükbilimsel sıralamayı zorlamak için, bu komut anahtardaki sıralı kümedeki öğe sayısını minimum ve maks.

```
127.0.0.1:6379> ZADD set1 0 a 0 b 0 c 0 d 0 e
(integer) 5
127.0.0.1:6379> ZADD set1 0 f 0 g
(integer) 2
127.0.0.1:6379> ZLEXCOUNT set1 - +
(integer) 7
127.0.0.1:6379> ZLEXCOUNT set1 [b [f
(integer) 5
127.0.0.1:6379>
```

9.12 Zmscore

Anahtarda depolanan sıralanmış kümedeki belirtilen üyelerle ilişkili puanları döndürür.

```
redis> ZADD myzset 1 "one"
(integer) 1
redis> ZADD myzset 2 "two"
(integer) 1
redis> ZMSCORE myzset "one" "two" "nofield"
1) "1"
2) "2"
3) (nil)
redis>
```

9.13 Zpopmax

Anahtarda depolanan sıralı kümedeki en yüksek puana sahip üyeleri kaldırır ve geri döndürür.

```
redis> ZADD myzset 1 "one"
(integer) 1
redis> ZADD myzset 2 "two"
(integer) 1
redis> ZADD myzset 3 "three"
(integer) 1
redis> ZPOPMAX myzset
1) "three"
2) "3"
redis>
```

9.14 Zpopmin

Anahtarda depolanan sıralı kümedeki en düşük puana sahip üyeleri kaldırır ve geri döndürür.

```
redis> ZADD myzset 1 "one"
(integer) 1
redis> ZADD myzset 2 "two"
(integer) 1
redis> ZADD myzset 3 "three"
(integer) 1
redis> ZPOPMIN myzset
1) "one"
2) "1"
redis>
```

9.15 Zrange

Anahtarda saklanan sıralanmış kümedeki belirtilen öğe aralığını döndürür. Öğeler, en düşük puandan en yüksek puana doğru sıralı olarak kabul edilir. Eşit puana sahip elemanlar için sözlüksel sıralama kullanılır.

```
127.0.0.1:6379> ZRANGE set1 0 -1
1) "a"
2) "b"
3) "c"
4) "d"
5) "e"
6) "f"
7) "g"
8) "one"
127.0.0.1:6379>
```

9.16 Zrangebylex

Sıralı bir kümedeki tüm öğeler, sözlükbilimsel sıralamayı zorlamak için aynı puanla eklendiğinde, bu komut anahtardaki sıralı kümedeki tüm öğeleri min. ve maks.

```
127.0.0.1:6379> ZADD myzset 0 a 0 b 0 c 0 d 0 e 0 f 0 g
(integer) 7
127.0.0.1:6379> ZRANGEBYLEX myzset - [c
1) "a"
2) "b"
3) "c"
127.0.0.1:6379>
```

9.17 Zrangebyscore

Anahtar üzerinde sıralanmış kümedeki tüm öğeleri minimum ve maksimum arasında bir puana sahip olarak döndürür (minimum veya maksimum puana eşit olan öğeler dahil). Öğeler, düşükten yükseğe doğru sıralı olarak kabul edilir.

```
127.0.0.1:6379> ZRANGEBYSCORE myzset -inf +inf
1) "a"
2) "b"
3) "c"
4) "d"
5) "e"
6) "f"
7) "g"
127.0.0.1:6379>
```

9.18 Zrank

Anahtarda depolanan sıralı kümedeki üye sıralamasını, düşükten yükseğe doğru sıralanan puanlarla döndürür. Derece (veya dizin) 0 tabanlıdır, yani en düşük puana sahip üye rütbenin 0 olduğu anlamına gelir.

```
127.0.0.1:6379> Zrank myzset "four"
(nil)
127.0.0.1:6379> ZRANK myzset "four"
(nil)
127.0.0.1:6379> ZADD myzset 1 "one"
(integer) 1
127.0.0.1:6379> ZADD myzset 2 "two"
(integer) 1
127.0.0.1:6379>
127.0.0.1:6379> ZADD myzset 3 "three"
(integer) 1
127.0.0.1:6379> ZRANK myzset "three"
(integer) 9
127.0.0.1:6379> ZRANK myzset "four"
(nil)
127.0.0.1:6379>
```

9.19 Zrem

Belirtilen üyeleri anahtarda depolanan sıralanmış kümeden kaldırır. Mevcut olmayan üyeler göz ardı edilir.

```
127.0.0.1:6379> ZREM myzset "two"
(integer) 1
127.0.0.1:6379>
```

9.20 Zremrangebylex

Sıralı bir kümedeki tüm öğeler aynı puanla eklendiğinde, sözlükbilimsel sıralamayı zorlamak için, bu komut minimum ve maksimum ile belirtilen sözlük aralığı arasında anahtarda saklanan sıralı kümedeki tüm öğeleri kaldırır.

```
127.0.0.1:6379> ZREMRANGEBYLEX myzset [alpha [omega
(integer) 6
127.0.0.1:6379>
```

9.21 Zremrangebyrank

Başlatma ve durdurma arasında sırayla anahtarda saklanan sıralı kümedeki tüm öğeleri kaldırır. Hem başlangıç hem de bitiş, 0 tabanlı dizindir ve 0 en düşük puana sahiptir.

```
:\Users\yaren>redis-cli
127.0.0.1:6379> ZADD myzset 1 "one"
(integer) 0
127.0.0.1:6379> ZADD myzset 2 "two"
(integer) 1
127.0.0.1:6379> ZADD myzset 3 "three"
(integer) 0
127.0.0.1:6379> ZREMRANGEBYRANK myzset 0 1
(integer) 2
127.0.0.1:6379> ZRANGE myzset 0 -1 WITHSCORES
1) "two"
2) "2"
3) "three"
4) "3"
127.0.0.1:6379>
```

9.22 Zremrangebyscore

Minimum ve maksimum arasında bir puanla anahtarda saklanan sıralanmış kümedeki tüm öğeleri kaldırır.

```
C:\Users\yaren>redis-cli
127.0.0.1:6379> ZADD myzset 1 "one"
(integer) 0
127.0.0.1:6379> ZADD myzset 2 "two"
(integer) 1
127.0.0.1:6379> ZADD myzset 3 "three"
(integer) 0
127.0.0.1:6379> ZREMRANGEBYRANK myzset 0 1
(integer) 2
127.0.0.1:6379> ZRANGE myzset 0 -1 WITHSCORES
1) "two"
2) "2"
3) "three"
4) "3"
127.0.0.1:6379> ZREMRANGEBYSCORE myzset -inf (2
(integer) 0
127.0.0.1:6379>
```

9.23 Zrevrange

Anahtarda saklanan sıralanmış kümedeki belirtilen öğe aralığını döndürür. Öğeler, en yüksekten en düşüğe doğru sıralı olarak kabul edilir. Eşit puana sahip öğeler için azalan sözlük sıralaması kullanılır.

```
C:\Users\yaren>redis-cli
127.0.0.1:6379> ZADD myzset 1 "one"
(integer) 0
127.0.0.1:6379> ZADD myzset 2 "two"
(integer) 1
127.0.0.1:6379> ZADD myzset 3 "three"
(integer) 0
127.0.0.1:6379> ZREMRANGEBYRANK myzset 0 1
(integer) 2
127.0.0.1:6379> ZRANGE myzset 0 -1 WITHSCORES
1) "two"
2) "2"
3) "three"
4) "3"
127.0.0.1:6379> ZREMRANGEBYSCORE myzset -inf (2
(integer) 0
127.0.0.1:6379> ZREVRANGE myzset -2 -1
1) "three"
2) "two"
127.0.0.1:6379>
```

9.24 Zrevrangebylex

Sıralı bir kümedeki tüm öğeler, sözlükbilimsel sıralamayı zorlamak için aynı puanla eklendiğinde, bu komut anahtardaki sıralı kümedeki tüm öğeleri max ve min arasında bir değerle döndürür.

```
127.0.0.1:6379> ZADD myzset 0 a 0 b 0 c 0 d 0 e 0 f 0 g
(integer) 7
127.0.0.1:6379>
127.0.0.1:6379> ZREVRANGEBYLEX myzset [c -
1) "c"
2) "b"
3) "a"
127.0.0.1:6379>
```

9.25 Zrevrangebyscore

Anahtar üzerinde sıralı kümedeki tüm öğeleri maks. Ve min. Arasında bir puana sahip (maks. Veya min. Puana eşit öğeler dahil) döndürür.

```
127.0.0.1:6379> ZREVRANGEBYSCORE myzset +inf -inf
1) "three"
2) "two"
3) "g"
4) "f"
5) "e"
6) "d"
7) "c"
8) "b"
9) "a"
127.0.0.1:6379>
```

9.26 Zrevrank

Anahtarda depolanan sıralı kümedeki üye sıralamasını, yüksekten düşüğe doğru sıralanmış puanlarla verir. Derece (veya dizin) 0 tabanlıdır, yani en yüksek puana sahip üye rütbesi 0'dır.

```
C:\Users\yaren>redis-cli
127.0.0.1:6379> ZREVRANK myzset "two"
(integer) 1
127.0.0.1:6379>
```

9.27 Zscan

Basit scan işlemidir.Scan komutu ile aynı işlemi yapar.

9.28 Zscore

Anahtardaki sıralı kümedeki üyenin puanını döndürür.

```
127.0.0.1:6379> ZSCORE myzset "one"
(nil)
127.0.0.1:6379> ZADD myzset 1 "on"
(integer) 1
127.0.0.1:6379> ZSCORE myzset "on"
"1"
127.0.0.1:6379>
```

9.29 Zunion

Bu komut ZUNIONSTORE'a benzer, ancak sonuçta ortaya çıkan sıralı kümeyi saklamak yerine istemciye geri döndürülür.

```
redis> ZADD zset1 1 "one"
(integer) 1
redis> ZADD zset1 2 "two"
(integer) 1
redis> ZADD zset2 1 "one"
(integer) 1
redis> ZADD zset2 2 "two"
(integer) 1
redis> ZADD zset2 3 "three"
(integer) 1
redis> ZUNION 2 zset1 zset2
1) "one"
2) "three"
3) "two"
redis> ZUNION 2 zset1 zset2 WITHSCORES
1) "one"
2) "2"
3) "three"
4) "3"
5) "two"
6) "4"
```

9.30 Zunionstore

Belirtilen anahtarlar tarafından verilen sayı anahtarları sıralı kümelerin birleşimini hesaplar ve sonucu hedefte depolar. Giriş anahtarlarını ve diğer (isteğe bağlı) bağımsız değişkenleri geçmeden önce giriş anahtarlarının (sayı tuşları) sayısını sağlamak zorunludur.

```
127.0.0.1:6379> ZUNIONSTORE out 2 zset1 zset2 WEIGHTS 2 3
(integer) 6
127.0.0.1:6379>
```

10 Referanslar

1 - https://redis.io/commands