

# 大数据库系统

## 3.3 Redis数据类型及操作

## 3.3 Redis数据类型及操作

### ◆ 主要内容

3.3.1 通用命令

3.3.2 STRING（字符串）

3.3.3 LIST（列表）

3.3.4 SET（无序集合）

3.3.5 ZSET（有序集合）


3.3.6 HASH（哈希表）

## 3.3 Redis数据类型及操作

### ◆ 进入redis官网的Commands页面: [redis.io/commands](https://redis.io/commands)

可以看到redis拥有五中数据类型、繁多的操作命令

常用的命令，深入理解，跟随练习，尤其注意别与hive搞混了

 Commands Clients Documentation Community Download Issues License			
All Keys Strings Hashes Lists Sets Sorted Sets Pub/Sub Transactions Scripting Connection Server			
Asynchronously save the dataset to disk	Get the value of a hash field	Set the expiration for a key as a UNIX timestamp specified in milliseconds	Manages the Redis slow queries log
<b>BITCOUNT</b> key [start] [end] Count set bits in a string	<b>HGETALL</b> key Get all the fields and values in a hash	<b>PING</b> Ping the server	<b>SMEMBERS</b> key Get all the members in a set
<b>BITOP</b> operation destkey key [key ...] Perform bitwise operations between strings	<b>HINCRBY</b> key field increment Increment the integer value of a hash field by the given number	<b>PSETEX</b> key milliseconds value Set the value and expiration in milliseconds of a key	<b>SMOVE</b> source destination member Move a member from one set to another
<b>BLPOP</b> key [key ...] timeout Remove and get the first element in a list, or block until one is available	<b>HINCRBYFLOAT</b> key field increment Increment the float value of a hash field by the given amount	<b>PSUBSCRIBE</b> pattern [pattern ...] Listen for messages published to channels matching the given patterns	<b>SORT</b> key [BY pattern] [LIMIT offset count] ... Sort the elements in a list, set or sorted set
<b>BRPOP</b> key [key ...] timeout Remove and get the last element in a list, or block until one is available	<b>HKEYS</b> key Get all the fields in a hash	<b>PUBSUB</b> subcommand [argument [argument ...]] Inspect the state of the Pub/Sub subsystem	<b>SPOP</b> key Remove and return a random member from a set
<b>BRPOPLPUSH</b> source destination timeout Pop a value from a list, push it to another list and return it; or block until one is available	<b>HLEN</b> key Get the number of fields in a hash	<b>P TTL</b> key Get the time to live for a key in milliseconds	<b>SRANDMEMBER</b> key [count] Get one or multiple random members from a set
<b>CLIENT KILL</b> ip:port Kill the connection of a client	<b>HMGET</b> key field [field ...] Get the values of all the given hash fields	<b>PUBLISH</b> channel message Post a message to a channel	<b>SREM</b> key member [member ...] Remove one or more members from a set

## 3.3.1 通用命令

### ◆ Keys命令

查询数据库中的key名

进入redis后，使用set命令存入“site”和“age”两个key及对应的其value

```
redis 127.0.0.1:6379> set site www.zixue.it
OK
redis 127.0.0.1:6379> set age 29
OK
```

返回“ok”即表示执行成功，这时我们使用keys \*命令查询数据库中的所有key

```
redis 127.0.0.1:6379> keys *
1) "age"
2) "site"
redis 127.0.0.1:6379>
```

成功返回了刚刚存入的“age”和“site”

## 3.3.1 通用命令

### ◆ Keys命令

- Keys命令可以精确地查询key名<sup>1</sup>

例：精确查询 “site” : keys site

```
redis 127.0.0.1:6379> keys site  
1) "site"  
redis 127.0.0.1:6379>
```

- Keys命令可以模糊查询key名：

\*: 通配任意多个字符

?: 通配单个字符

[]: 通配括号内的某1个字符

## 3.3.1 通用命令

例1：模糊查询：\*通配匹配任意多个字符

```
redis 127.0.0.1:6379> keys s*  
1) "site"  
redis 127.0.0.1:6379> keys site*  
1) "site"
```

例2：模糊查询：[] 通配括号内的某1个字符

```
redis 127.0.0.1:6379> keys sit[ty]  
(empty list or set)  
redis 127.0.0.1:6379> keys sit[ey]  
1) "site"
```

Sit[ey]表示site或者sity，sit[ty]表示sitt或sity，数据库中并没有sitt或sity所以返回空

例3：模糊查询：? 通配单个字符

```
redis 127.0.0.1:6379> keys si?e  
1) "site"  
redis 127.0.0.1:6379>
```

si和e中有一个字符<sup>2</sup>

问：si\*e和si?e有何区别？

## 3.3.1 通用命令

### ◆ RANDOMKEY命令

不指定，随机返回数据库中的key名<sup>3</sup>

```
redis 127.0.0.1:6379> randomkey
"site"
redis 127.0.0.1:6379> randomkey
"site"
redis 127.0.0.1:6379> randomkey
"age"
redis 127.0.0.1:6379> randomkey
"age"
redis 127.0.0.1:6379> randomkey
"site"
redis 127.0.0.1:6379> randomkey
"age"
redis 127.0.0.1:6379> randomkey
"age"
redis 127.0.0.1:6379> randomkey
"age"
redis 127.0.0.1:6379> randomkey
"site"
redis 127.0.0.1:6379> randomkey
"age"
```

数据库中只有site和age，因此随机返回两者之一

## 3.3.1 通用命令

### ◆ EXISTS命令

exists key

判断key是否存在，返回1或0

例：判断是否存在site和age这两个key；

判断是否存在title这个key

```
redis 127.0.0.1:6379> exists age
(integer) 1
redis 127.0.0.1:6379> exists site
(integer) 1
redis 127.0.0.1:6379> exists title
(integer) 0
redis 127.0.0.1:6379> 
```

site和age这两个key在数据库中所以返回1

title不存在于数据库中，返回0



## 3.3.1 通用命令

### ◆ TYPE命令

type key

返回key存储的值的类型

例：分别查看site和age这两个key的值的数据类型

```
redis 127.0.0.1:6379> type age
string
redis 127.0.0.1:6379> type site
string
redis 127.0.0.1:6379>
```

## 3.3.1 通用命令

### ◆ DEL命令

`del key1 key2 ... KeyN`

作用: 删除1个或多个键

返回值: 返回删除的key的数量, 如果key不存在则忽略

例1: 查询数据库中所有key并删除age

```
redis 127.0.0.1:6379> keys *  
1) "age"  
2) "site"  
redis 127.0.0.1:6379> del age  
(integer) 1  
redis 127.0.0.1:6379> 
```

返回1表示删除了1个key, 此时再查询数据库里的key

```
redis 127.0.0.1:6379> del age  
(integer) 1  
redis 127.0.0.1:6379> keys *  
1) "site"  
redis 127.0.0.1:6379> exists age  
(integer) 0  
redis 127.0.0.1:6379> 
```

Exists age返回0, 说明已成功删除

## 3.3.1 通用命令

### ◆ RENAME命令

rename key newkey

作用: 给key赋一个新的key名

注:如果newkey已存在, 则newkey的原值被覆盖<sup>4</sup>

- 例: 将site的key名改成wangzhi

```
redis 127.0.0.1:6379> rename site wangzhi
OK
```

- 改名返回成功, 我们再查询下数据库, 并原来的site是否存在

```
redis 127.0.0.1:6379> exists site
(integer) 0
redis 127.0.0.1:6379> keys *
1) "wangzhi"
redis 127.0.0.1:6379>
```

```
redis 127.0.0.1:6379> get wangzhi
"www.zixue.it"
redis 127.0.0.1:6379>
```

- Site已成功改名为wangzhi

## 3.3.1 通用命令

### ◆ renamenx key newkey

作用: 把key改名为newkey

返回: 发生修改返回1, 未发生修改返回0

注: nx表示not exists, 即newkey不存在时, 才改名;

- 例1: 先删除wangzhi这个key再执行下述改名命令后, site和search会发生什么?

```
redis 127.0.0.1:6379> del wangzhi
(integer) 1
redis 127.0.0.1:6379> set site www.zixue.it
OK
redis 127.0.0.1:6379> set search www.so.com
OK
redis 127.0.0.1:6379> rename site search
```

- 执行“rename site search”的结果如下, site变为search, 然后覆盖掉原来的search

```
redis 127.0.0.1:6379> rename site search
OK
redis 127.0.0.1:6379> get search
"www.zixue.it"
redis 127.0.0.1:6379>
```

## 3.3.1 通用命令

### ◆ renamenx key newkey

- 例2：接上例，将site和search恢复原状，并使用renamenx命令对site重命名为sea观察结果

```
redis 127.0.0.1:6379> set site www.zixue.it
OK
redis 127.0.0.1:6379> set search www.so.com
OK
redis 127.0.0.1:6379> renamenx site sea
(integer) 1
redis 127.0.0.1:6379> get sea
"www.zixue.it"
```

- site重命名为sea成功，接着再将sea重命名为search再观察结果

```
redis 127.0.0.1:6379> renamenx sea search
(integer) 0
redis 127.0.0.1:6379> keys *
1) "search"
2) "sea"
redis 127.0.0.1:6379>
```

- Renamenx返回0，即search已存在，修改失败，因此数据库中为sea和search两个key

## 3.3.1 通用命令

### ◆ SELECT命令<sup>5</sup>

Select dbnum

切换到指定的数据库

- 在redis中，默认数据库数量为16，命名为“0”、“1”、...、“15”
- 默认使用的是“0”数据库
- 如果想改变数据库的数量，可以修改redis.conf文件中的“databases”后的值

```
# Select the number of databases. The default is 16. You can select any number from 1 to 32,767. Increasing the number of databases will cause Redis to allocate more memory, so be careful.
databases 16
```

## 3.3.1 通用命令

### ◆ 使用select db来切换当前数据库

例：查询当前数据库的key，切换到“1”数据，再查询key，观察对比

```
redis 127.0.0.1:6379> keys *  
1) "search"  
2) "sea"
```

当前“0”数据库有两个key，切换到数据库“1”

```
redis 127.0.0.1:6379> select 1  
OK  
redis 127.0.0.1:6379[1]> keys *  
(empty list or set)  
redis 127.0.0.1:6379[1]>
```

数据库“1”没有任何数据，注意提示符上中括号标明了当前数据库名

## 3.3.1 通用命令

### ◆ MOVE命令

MOVE key db

将当前数据库的某个key移动到指定数据库

例:

```
redis 127.0.0.1:6379[1]> select 2
```

OK

```
redis 127.0.0.1:6379[2]> keys *
```

(empty list or set)

```
redis 127.0.0.1:6379[2]> select 0
```

OK

```
redis 127.0.0.1:6379> keys *
```

1) "name"

2) "cc"

3) "a"

4) "b"

```
redis 127.0.0.1:6379> move cc 2
```

(integer) 1

```
redis 127.0.0.1:6379> select 2
```

OK

```
redis 127.0.0.1:6379[2]> keys *
```

1) "cc"



## 3.3.1 通用命令

### ◆ Key的生命周期操作命令

ttl key                      查询key的生命周期

expire key 整型值              设置key的生命周期，以秒为单位

persist key                      把指定key置为永久有效

### ◆ Redis既可以用作**存储**，也可以用作**缓存**<sup>6</sup>

用作存储，key通常为永久生命周期

用作缓存，key通常有生命周期

## 3.3.1 通用命令

### ◆ TTL命令

`ttl key`

查询key的生命周期，返回：秒数

注：对于不存在的key、已过期的key（不存在已过期等价的）、永久存在的key，都返回-1

Redis2.8中，对于不存在、已过期的key,返回-2

- 例：查询search的生命周期，查询dsafsdf的生命周期

```
redis 127.0.0.1:6379> ttl search
(integer) -1
redis 127.0.0.1:6379> ttl dsafsdf
(integer) -1
redis 127.0.0.1:6379>
```

```
127.0.0.1:6379> ttl search
(integer) -1
127.0.0.1:6379> ttl dsafsdf
(integer) -2
127.0.0.1:6379>
```

返回-1的情况下可以用get、exists进一步判断key是否存在

## 3.3.1 通用命令

### ◆ EXPIRE命令

expire key 整型值

设置key的生命周期，以秒为单位

例：将search的生命周期设为10s，使用ttl跟踪search的生命周期

```
127.0.0.1:6379> set search www.so.com
OK
127.0.0.1:6379> ttl search
(integer) -1
127.0.0.1:6379> ttl dsafsd
(integer) -2
127.0.0.1:6379> expire search 10
(integer) 1
127.0.0.1:6379> ttl search
(integer) 7
127.0.0.1:6379> ttl search
(integer) 6
127.0.0.1:6379> ttl search
(integer) 5
127.0.0.1:6379> ttl search
(integer) 3
127.0.0.1:6379> ttl search
(integer) 1
127.0.0.1:6379> ttl search
(integer) -2
127.0.0.1:6379> ttl search
(integer) -2
```

十秒后search生命周期结束，故ttl search 返回-2，即不存在search

## 3.3.1 通用命令

### ◆ pexpire key 毫秒数

以毫秒为单位设置key的生命周期

### ◆ pttl key

以毫秒返回key的生命周期

### ◆ persist key

把指定key置为永久有效

例:

```
redis 127.0.0.1:6379> set site www.zixue.it
OK
redis 127.0.0.1:6379> expire site 10
(integer) 1
redis 127.0.0.1:6379> ttl site
(integer) 7
redis 127.0.0.1:6379> ttl site
(integer) 6
redis 127.0.0.1:6379> persist site
(integer) 1
redis 127.0.0.1:6379> ttl site
(integer) -1
redis 127.0.0.1:6379> get site
"www.zixue.it"
redis 127.0.0.1:6379>
```

## 3.3 Redis数据类型及操作

### ◆ 通用命令

DEL

RENAME

RENAMENX

MOVE

KEYS

RANDOMKEY

EXISTS

TYPE

SELECT

TTL

EXPIRE

PERSIST

## 3.3 Redis数据类型及操作

### 3.3.1 通用命令

### 3.3.2 STRING（字符串）

### 3.3.3 LIST（列表）

### 3.3.4 SET（无序集合）

### 3.3.5 ZSET（有序集合）

### 3.3.6 HASH（哈希表）

## 3.3.2 STRING（字符串）

### ◆ SET命令

set key value [EX seconds]/[PX milliseconds] [nx] /[xx]

使用SET命令将字符串值value设置到key中。

- EX seconds：用于设置key的过期时间为多少秒（seconds）
- PX milliseconds：用于设置key的过期时间为多少毫秒（milliseconds）
- NX：表示当key不存在时，才对key进行设置操作
- XX：表示当key存在时，才对key进行设置操作

**注意：**如果key已经存在，则在执行SET命令后，将会覆盖旧值

## 3.3.2 STRING（字符串）

◆ set key value [EX seconds]/[PX milliseconds] [nx] /[xx]

- 例：

```
redis 127.0.0.1:6379> flushdb
OK
redis 127.0.0.1:6379> set site www.so.com
OK
redis 127.0.0.1:6379> set site www.baidu.com nx
(nil)
redis 127.0.0.1:6379> get site
"www.so.com"
redis 127.0.0.1:6379> set site www.google.com xx
OK
redis 127.0.0.1:6379> get site
"www.google.com"
redis 127.0.0.1:6379>
```

- Set后加xx表示site存在时才执行set操作，因此site被覆盖为[www.google.com](http://www.google.com)

```
redis 127.0.0.1:6379> set abc www.google.com xx
(nil)
redis 127.0.0.1:6379>
```

- 由于abc不存在，因此该set指令不予生效



## 3.3.2 STRING（字符串）

### ◆ MSET命令

MSET key value [key value...]

一次性设置多个键值

**注意：**如果某个key已经存在，那么MSET命令会用新值覆盖旧值

- 例：使用一条命令设置多个键值：

a aman

b bold

c controller

d diamond

```
redis 127.0.0.1:6379> mset a aman b bold c controller d diamond
OK
redis 127.0.0.1:6379> keys *
1) "a"
2) "d"
3) "b"
4) "c"
5) "site"
redis 127.0.0.1:6379> get a
"aman"
redis 127.0.0.1:6379> get b
"bold"
redis 127.0.0.1:6379>
```

## 3.3.2 STRING（字符串）

### ◆ GET key

获取key的值，当key存在时，返回key所对应的值；如果key不存在，则返回nil；如果key不是字符串类型的，则返回错误<sup>8</sup>；

### ◆ MGET key [key...]

同时返回多个给定key的值，key之间使用空格隔开

例：使用一条指令获得a、b和c的value

```
redis 127.0.0.1:6379> keys *
1) "a"
2) "d"
3) "b"
4) "c"
5) "site"
redis 127.0.0.1:6379> get a
"aman"
redis 127.0.0.1:6379> get b
"bold"
redis 127.0.0.1:6379> mget a b c
1) "aman"
2) "bold"
3) "controller"
redis 127.0.0.1:6379>
```

## 3.3.2 STRING（字符串）

### ◆ SETRANGE命令

SETRANGE key offset value2

把key的value中的第offset位置开始替换成value2，从位置0开始算

“hello” 单词

h	e	l	l	o
offset =0	offset =1	offset =2	offset =3	offset =4

例1：将hello中的两个“l”替换成两个“？”

```
redis 127.0.0.1:6379> flushdb
OK
redis 127.0.0.1:6379> set word hello
OK
redis 127.0.0.1:6379> get word
"hello"
redis 127.0.0.1:6379> setrange word 2 ??
(integer) 5
redis 127.0.0.1:6379> get word
"he??o"
redis 127.0.0.1:6379>
```

## 3.3.2 STRING（字符串）

### ◆ SETRANGE key offset value2

注意: 如果偏移量>字符串长度, 该字符自动补”\x00”

例2:

```
redis 127.0.0.1:6379> set word hello
OK
redis 127.0.0.1:6379> setrange word 6 !
(integer) 7
redis 127.0.0.1:6379> get word
"hello\x00!"
redis 127.0.0.1:6379>
```

由于hello的offset最高为4，所以在offset=5的位置补上”\x00”，在offset=6的位置加上 “！”

## 3.3.2 STRING（字符串）

### ◆ APPEND命令

APPEND key value

如果key存在且是字符串类型的，则将value值追加到key旧值的末尾

如果key不存在，则将key设置值为value

返回值：返回追加value之后，key中字符串的长度

例：word后追加字符

```
redis 127.0.0.1:6379> get word
"hello\x00!"
redis 127.0.0.1:6379> append word @@
(integer) 9
redis 127.0.0.1:6379> get word
"hello\x00!@@"
redis 127.0.0.1:6379>
```

## 3.3.2 STRING（字符串）

### ◆ GETRANGE命令

GETRANGE key start end

使用GETRANGE命令来获取key中字符串值从start开始到end结束的子字符串，下标从0开始（字符串截取）

注意： start和end参数是整数，可以取负值。当取负值时，表示从字符串最后开始计数，-1表示最后一个字符，-2表示倒数第二个字符，以此类推<sup>9</sup>。

h	e	l	l	o
0	1	2	3	4
-5	-4	-3	-2	-1

## ◆ GETRANGE key start end

例：

```
redis 127.0.0.1:6379> set area chinese
OK
redis 127.0.0.1:6379> getrange area 1 4
"hine"
redis 127.0.0.1:6379> set status working
OK
redis 127.0.0.1:6379> getrange status 0 -3
"worki"
redis 127.0.0.1:6379>
```

w	o	r	k	i	n	g
0	1	2	3	4	5	6
-7	-6	-5	-4	-3	-2	-1

假定value长度为length,

- 1、如果start>=length, 则返回空字符串
- 2、end>=length, 则截取至字符结尾
- 3、如果start 所处位置在end右边, 返回空字符串<sup>10</sup>

## 3.3.2 STRING（字符串）

### ◆ GETSET命令

GETSET key value

使用GETSET命令将给定key的值设置为value，并返回key的旧值。

返回值：返回给定key的旧值。如果key不存在，则返回nil；如果key存在但不是字符串类型的，则返回错误。

例：

```
redis 127.0.0.1:6379> set status sleep
OK
redis 127.0.0.1:6379> getset status wakeup
"sleep"
redis 127.0.0.1:6379> get status
"wakeup"
redis 127.0.0.1:6379> getset status working
"wakeup"
redis 127.0.0.1:6379> 
```



## 3.3.2 STRING（字符串）

### ◆ STRLEN命令

STRLEN key

使用命令STRLEN统计key的值的字符长度。当key存储的不是字符串时，返回错误。当key不存在时，返回0

```
redis> SET mykey "Hello world"
```

OK

```
redis> STRLEN mykey
```

(integer) 11

## 3.3.2 STRING（字符串）

### ◆ INCR命令、 DECR命令

INCR key、 DECR key

使用INCR/ DECR命令将key中存储的数字值加1/减1。如果key不存在，则key的值先被初始化为0，再执行INCR / DECR操作加1 /减1

返回值：返回执行INCR / DECR命令之后key的新值

注意：只能对数字类型的数据进行操作

常用于抢预约资格等场景，内存先设置剩余数量，用户预约则剩余数减1

例：

```
redis 127.0.0.1:6379> set age 29
OK
redis 127.0.0.1:6379> get age
"29"
redis 127.0.0.1:6379> incr age
(integer) 30
redis 127.0.0.1:6379> decr age
(integer) 29
```

## 3.3.2 STRING（字符串）

### ◆ INCRBY、DECRBY

INCRBY/ DECRBY key number

使用INCRBY / DECRBY命令将key所存储的值加上/减去值number

如果key不存在，则key的值先被初始化为0，再执行命令

返回值：返回加上/减去num之后的key的新值

注意：只能对数字类型的数据进行操作

例：

```
redis 127.0.0.1:6379> decr age  
(integer) 29
```

```
redis 127.0.0.1:6379> incrby age 5  
(integer) 34  
redis 127.0.0.1:6379> incrby age 5  
(integer) 39  
redis 127.0.0.1:6379> decrby age 10  
(integer) 29  
redis 127.0.0.1:6379>
```

## 3.3.2 STRING（字符串）

### ◆ INCRBYFLOAT、DECRBYFLOAT命令

INCRBYFLOAT / DECRBYFLOAT key number

使用INCRBYFLOAT / DECRBYFLOAT命令将key所存储的值加上/减去值  
number，如果key不存在，则key的值先被初始化为0，再执行命令

返回值：返回加上/减去num之后的key的新值

注意：只能对数字类型的数据进行操作

## 3.3.2 STRING（字符串）

### ◆ SETBIT

SETBIT key offset value

设置offset对应二进制位上的值，当key不存在的时候，就创建一个新的字符串value。要确保这个字符串大到在offset处有bit值

返回：该位上的旧值

注意：1、如果offset过大,则会在中间填充0

2、offset最大 $2^{32}-1$ ,可推出最大的的字符串为512M。

### ◆ GETBIT命令

GETBIT key offset

获取值的二进制表示,对应位上的值(从左,从0编号)

返回值：返回字符串值指定偏移量上的位（bit）

## 3.3.2 STRING（字符串）

- ◆ SETBIT K1 1 1 : 第2位上设置为1，即01000000。

按ASCII码表 对应@

- ◆ SETBIT K1 7 1 : 第8位设为1，即01000001。

按ASCII码表 对应A

- ◆ SETBIT K1 9 1 : 第10位设为1。即01000001 01000000

分字节来按ASCII码表 对应 A@

- ◆ SETBIT K1 26 1 : 第27位设为1。即

01000001 01000000 00000000 00100000

例：使用setbit命令改变字符大小写

分析：

字母A ASCII为65 二进制表示：0100 0001

字母a ASCII为97 二进制表示：0110 0001

字母B ASCII为66

字母b ASCII为98

可知，大小写字母ASCII相差32，因此大写变小写，将二进制表示的第3个位置  
(offset=2) 为1，小写变大写，将二进制表示的第3个位置为0

```
redis 127.0.0.1:6379> set char A
OK
redis 127.0.0.1:6379> setbit char 2 1
(integer) 0
redis 127.0.0.1:6379> get char
"a"
redis 127.0.0.1:6379> set char B
OK
redis 127.0.0.1:6379> setbit char 2 1
(integer) 0
redis 127.0.0.1:6379> get char
"b"
redis 127.0.0.1:6379> setbit char 2 0
(integer) 1
redis 127.0.0.1:6379> get char
"B"
redis 127.0.0.1:6379>
```

## 3.3.2 STRING（字符串）

### ◆ BITOP命令

BITOP operation destkey key [key...]

使用BITOP命令对一个或多个保存二进制位的字符串key进行位元运算，并将运算结果保存到destkey中。

operation表示位元操作符，它可以是AND、OR、NOT、XOR这4种操作中的任意一种

BITOP AND destkey key [key...]: 表示对一个或多个key求逻辑与，并将结果保存到destkey中。

BITOP OR destkey key [key...]: 表示对一个或多个key求逻辑或，并将结果保存到destkey中。

BITOP NOT destkey key: 表示对给定key求逻辑非，并将结果保存到destkey中

BITOP XOR destkey key [key...]: 表示对一个或多个key求逻辑异或，并将结果保存到destkey中。



例：通过BITOP命令进行字母大小写转换

```
redis 127.0.0.1:6379> setbit lower 2 1
(integer) 0
redis 127.0.0.1:6379> set char Q
OK
redis 127.0.0.1:6379> bitop or char char lower
(integer) 1
redis 127.0.0.1:6379> get char
"q"
redis 127.0.0.1:6379>
```

setbit lower 2 1 即指定lower为0010 0000

大写字母与lower作“或”操作变小写

指定一个upper为1101 1111，与小写字母做AND操作可变为大写字母，可以用NOT操作直接把lower变成upper

字母A ASCII为65 二进制表示：0100 0001

字母a ASCII为97 二进制表示：0110 0001

字母B ASCII为66

字母b ASCII为98

可知，大小写字母ASCII相差32，因此大写变小写，将二进制表示的

## 3.3.2 STRING（字符串）

### ◆ 需求：

假定有一个网站，有1亿个用户，如何记录每个用户当日是否登录？每周奖励活跃用户，即连续登录7天给与奖励，如何找出连续七天都登录的用户？

分析：由于每个用户在数据库中都有编号：比如用户1，用户2...

用户每日登录与未登录只有两种状态0或1

因此每个用户用1个位来表示

## 3.3.2 STRING（字符串）

假定只有八个用户，可以用8位二进制表示这8个用户是否登录

	0	1	0	1	0	0	1	0
周一：	id=1今日未登录	id=2今日有登录	id=3今日未登录	id=4今日有登录	id=5今日未登录	id=6今日未登录	id=7今日有登录	id=8今日未登录

求一周连续登录的用户，只需要将每周的值进行AND操作，二进制位为1的对应用户就是一周连续登录的用户

优点：

1: 节约空间, 1亿人每天的登陆情况,用1亿bit, 约1200WByte,约10M的字符就能表示

2: 计算方便

周二： 0110 1111

周三： 1110 0011

.....

周日： 0010 1111

redis 127.0.0.1:6379> setbit mon 100000000 0

redis 127.0.0.1:6379> setbit mon 3 1

redis 127.0.0.1:6379> setbit mon 5 1

redis 127.0.0.1:6379> setbit mon 7 1

.....

redis 127.0.0.1:6379> setbit thur 100000000 0

redis 127.0.0.1:6379> setbit thur 3 1

redis 127.0.0.1:6379> setbit thur 5 1

redis 127.0.0.1:6379> setbit thur 8 1

.....

redis 127.0.0.1:6379> setbit wen 100000000 0

redis 127.0.0.1:6379> setbit wen 3 1

redis 127.0.0.1:6379> setbit wen 4 1

redis 127.0.0.1:6379> setbit wen 6 1

.....

redis 127.0.0.1:6379> bitop and res mon ..... sun

# 总结

## ◆ 通用命令

DEL  
RENAME  
RENAMENX  
MOVE  
KEYS  
RANDOMKEY  
EXISTS  
TYPE  
SELECT  
TTL  
EXPIRE  
PERSIST

## ◆ STRING字符串操作命令：

SET  
MSET  
GET  
MGET  
SETRANGE  
APPEND  
GETRANGE  
GETSET  
INCR  
INCRBY  
INCRBYFLOAT  
DECR  
DECRBY  
GETBIT  
SETBIT  
BITOP