**Title:** Redis BITCOUNT

**Excerpt:** This article explained the Redis Bitmap data structure and how we can use the BITCOUNT command against a bitmap or string to count the number of set bits(1s).

**Permalink:** redis-bitcount

**Category:** Redis

# Redis Data Structures

Redis is a next-level implementation of traditional key-value stores. It is not limited to holding string values for a given key. Redis can store more complex data structures(types) such as Lists, Hashes, Sets, and Bitmaps(Bit arrays). The string data type is still available in Redis. It has been used to implement some of these complex data structure operations.

## What are Redis Bitmaps?

The Bitmap is not a native data type in Redis. Its underlying implementation is based on the string data type. It is a set of functionalities built on the String data type. The easiest way to understand a Bitmap is to think of it as an array of bits.



As mentioned above, This is a string representation with bit operation capabilities. The bit is the smallest storage size inside a computer. Hence, each bit can store 1 or 0 at a given time.



The maximum length of a Redis string is 512MB. If we convert this value to bits, it is about 4 Billion bits which is more than enough to use in a real-world application. We call the array index ‘offset’ in Redis bitmaps. Let’s have a look at the following example.



## Bitmap Operations

There are two main types of operations associated with Redis bitmaps. The single-bit operations are performed on a specific bit such as get value or set value. There is another type of operation that is performed on a group of bits such as BITCOUNT.

### The BITCOUNT Command

The BITCOUNT command is a batch-type operation. It counts the number of occurrences of ‘1’s in a given bitmap or string. We call this ‘population counting’ or count of set bits.

#### Syntax

| BITCOUNT your\_key [ interval\_start interval\_end ] [BYTE | BIT] |
| --- |

Your\_key: This is the key of the string or bitmap. It is a mandatory parameter.

Interval\_start, interval\_end: These two parameters specify the range using the start and end indexes. These two parameters are optional.

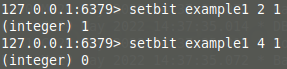
**BYTE or BIT**: This parameter specifies the interval start and end as a BYTE index or BIT index. This parameter is optional. By default, BYTE is used.

#### Example 01

Let’s create a key “example1” and set the values of the second and fourth bits to 1. We’ll be using the Redis **setbit** command here.

| setbit example1 2 1  setbit example1 4 1 |
| --- |

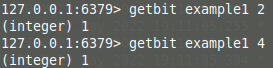
Output:



Let’s verify the created bitmap index value using Redis **getbit** command.

| getbit example1 2  getbit example1 4 |
| --- |

Output:



The 2nd and 4th bits are set to 1 as expected.

Let’s use the **BITCOUNT** command to count the number of set-bits or 1s in the example1 bitmap.

| **bitcount** example1 |
| --- |

Output:



The example1 bitmap looks like the following.

| 0 | 0 | 1 | 0 | 1 |
| --- | --- | --- | --- | --- |

As you can see, the 2nd and the 4th offsets are set to 1. Hence the **bitcount** command output should be 2 as above.

#### Example 02

Let’s create a new key example2 and assign the string “A”. The string A is represented by 8 bits(1 byte) as shown in the following.

| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| --- | --- | --- | --- | --- | --- | --- | --- |

| **set** example2 "A" |
| --- |

Output:



Let’s use the **bitcount** command to check the number of set bits. Since we got the 1st and 7th bit set to 1, the **bitcount** command output should be 2.

| **bitcount** example2 |
| --- |

Output:



Normally, the bitcount command checks all the bytes contained in the array. In some scenarios, it might be a redundant process to examine all the bytes in a bitmap or string. Hence, we can specify a range to perform the bitcount operation as shown in the following.

| **bitcount** key start\_index end\_index |
| --- |

By default, the start\_index and end\_index parameter values are based on byte indexes. Let’s try this out in the following example.

#### Example 03

We will be creating a new key named example3 and assigning the value “AB”.

| **set** example3 "AB" |
| --- |

The example3 string should look like the following.

| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

**|**  **A** **| B**

The first 8 bits represent the letter A and the second 8 bits represent the letter B. The string “AB” takes 2 Bytes. Let’s use the bitcount command to count the number of set bits for a given interval.

| Bitcount example3 0 0 |
| --- |

We have specified both the start and end Byte indexes as 0 in the above command. That means it will count the 1s in the first byte(8 bits) only. Therefore, the output value should be 2.

Output:



If we specify both the start and end indexes as 1, the bitcount command will count set bits only in the second Byte(second 8 bits) which represents the letter B. It should be two again.

| Bitcount example3 1 1 |
| --- |

Output:



We can retrieve all set bits in the string “AB” by specifying the range from 0th byte to 1st byte. The output should be four. Since we got four 1s in the whole string.

| Bitcount example3 0 1 |
| --- |

Output:



The bitcount command allows users to specify the interval using the bit index. The string “AB” has 16 bits as shown in the above illustration. Therefore, the interval minimum and max indexes will be 0 and 15 respectively. We need to specify this explicitly to the Redis command using the **BIT** argument. Then the **bitcount** command will treat start and end indexes as a bit index.

Let’s count the setbits from 1st bit(0th index) to 4th bit(3rd index)

| **bitcount** example3 0 3 **BIT** |
| --- |

Notice the newly passed **BIT** argument. Now it examines the set bits from 0th to 3rd Bit index. The output should be one.

| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

**|<**================>|

**Only 1 set bit is in this range**

Output:



Next we will give the interval from 1st bit(oth bit index) to 10th bit(9th bit index).

| **bitcount** example3 0 9 **BIT** |
| --- |

| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

**|<**===========================================>|

**Only 3 set bits are in this range**

According to the above illustration, the output should be 3.

Output:



# Conclusion

Redis can store different types of data structures for a specific key. Bitmaps are one of the useful data structures that Redis supports. The underlying implementation is a String representation with Bitmap operations supported. The bitcount is a Redis command that can be used to count the number of set bits in a given Bitmap or String.