bitcount

- Examples
- Pattern: real-time metrics using bitmaps
- Performance considerations

Count the number of set bits (population counting) in a string.

By default all the bytes contained in the string are examined. It is possible to specify the counting operation only in an interval passing the additional arguments *start* and *end*.

Similar to the `GETRANGE` command, `start` and `end` can contain negative values in order to index bytes starting from the end of the string, where -1 is the last byte, -2 is the penultimate, and so forth.

Non-existent keys are treated as empty strings, so the command will return zero.

By default, the additional arguments `start` and `end` specify a byte index. We can use an additional argument `BIT` to specify a bit index. So 0 is the first bit, 1 is the second bit, and so forth. For negative values, -1 is the last bit, -2 is the penultimate, and so forth.

Examples

```
127.0.0.1:6379> SET mykey "foobar"
OK
127.0.0.1:6379> BITCOUNT mykey
(integer) 26
127.0.0.1:6379> BITCOUNT mykey 0 0
(integer) 4
127.0.0.1:6379> BITCOUNT mykey 1 1
(integer) 6
127.0.0.1:6379> BITCOUNT mykey 1 1 BYTE
(integer) 6
127.0.0.1:6379> BITCOUNT mykey 5 30 BIT
```

```
(integer) 17
```

Pattern: real-time metrics using bitmaps

Bitmaps are a very space-efficient representation of certain kinds of information. One example is a Web application that needs the history of user visits, so that for instance it is possible to determine what users are good targets of beta features.

Using the `SETBIT` command, this is trivial to accomplish, identifying every day with a small progressive integer. For instance day 0 is the first day the application was put online, day 1 the next day, and so forth.

Every time a user performs a page view, the application can register that in the current day the user visited the web site using the `SETBIT` command setting the bit corresponding to the current day.

Later it will be trivial to know the number of single days the user visited the web site simply calling the `BITCOUNT` command against the bitmap.

A similar pattern where user IDs are used instead of days is described in the article called "Fast easy realtime metrics using Redis bitmaps".

Performance considerations

In the above example of counting days, even after 10 years the application is online we still have just `365*10` bits of data per user, that is just 456 bytes per user. With this amount of data `BITCOUNT` is still as fast as any other O(1) Valkey command like `GET` or `INCR`.

When the bitmap is big, there are two alternatives:

- Taking a separated key that is incremented every time the bitmap is modified. This can be very efficient and atomic using a small Lua script.
- Running the bitmap incrementally using the `BITCOUNT` `start` and `end` optional parameters, accumulating the results client-side, and optionally caching the result into a key.

□□□ #1
□ □□□ □□□ 2026-10-31 15:09:51 UTC
□ □□□ □□□ 2026-10-31 15:09:51 UTC