**Title:** Redis HINCRBY

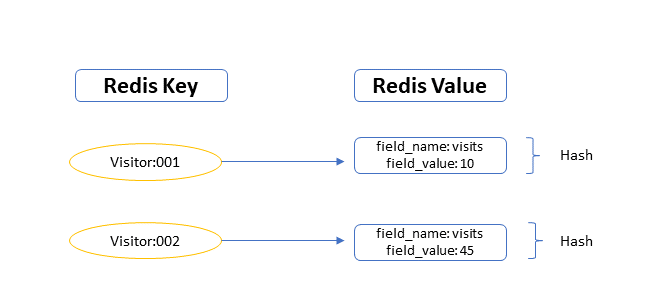
**Excerpt:** Redis hashes are one of the most popular data structures to use in Redis databases. Therefore, Redis supports different operations to manipulate hashes. One of the important commands is the HINCRBY command which increments the hash field value by a given increment.

**Permalink:** redis-hincrby

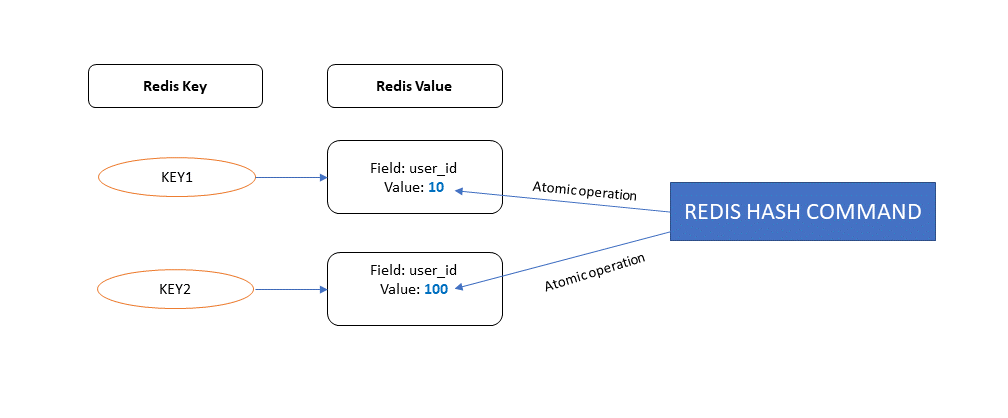
**Category:** Redis

# Redis Hash Manipulation

Redis hash can be identified as a field-value pair data structure. This can be stored at a specified Redis key as shown in the following.



Usually, the hash field values can store string, integer, and float data. One of the atomic operations that we can perform on hashes is increment or decrement of the integer values assigned to a given hash field.



The HINCRBY command is used to perform increment/decrement of an integer value. In the following sections, we will be discussing more the practical usage of the HINCRBY hash command.

# The HINCRBY command

## Syntax

HINCRBY ***redis\_key*** ***hash\_field\_name increment\_by\_value***

**redis\_key**: The top-level key of the Redis store

**hash\_field\_name**: The unique identifier of the hash field

**increment\_by\_value**: The integer value of the increment argument

The return value of the HINCRBY command is an integer. It is the value of the hash field after the operation has been performed. The *increment\_by\_value* argument supports 64-bit signed integers that enable passing both positive and negative numbers.

### When the given hash field doesn’t exist

Let’s say we have a Redis hash key *user:001* with one field named *id*. The value of the *id* field is 10. If we are trying to increment the hash value by 5 for non-existing key *age,* then Redis will create a new field *age* and assign its value to 5.

### When the specified Redis key doesn’t exist

Assume that we pass the key as *user:002* to the HINCRBY command with the field *age* and the increment value *10*. Since the *user:002* key is not available in the Redis store, it will create a new key *user:002* with a field age. The value of the age field will be *10* which is the passed increment argument value.

## Practical usage

There are many real-world use cases of the HINCRBY hash command.

* Visit counters in web applications
* New user-id assignment
* Blocking multiple login attempts
* Track usage and billing status
* Update Shopping cart units

The main advantage of the HINCRBY operation is that it doesn’t need an explicit read operation to update the hash value. You can directly update the Redis store’s hash value. This makes the operations real fast.

### Example 01 - Capturing usage and cost per user

Let’s assume that we need to store usage and cost data per user in a given system. Hence, we will identify each user with a Redis key. The usage and cost will be the hash fields for that particular Redis key.

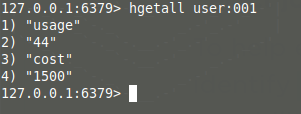
| **HMSET** **user**:001 **usage** 50 **cost** 1500 |
| --- |

We can use the HMSET hash command to set multiple fields for a hash.

Let’s check whether the user:001 hash is stored properly in the data store.

| **HGETALL** **user**:001 |
| --- |

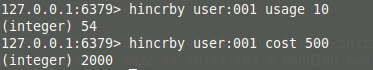
Output:



Assume the user used another 10 units of resources within the system and it would cost another 500 dollars. Now we can easily use the HINCRBY hash command to update the data for *user:001.*

| **HINCRBY** **user**:001 **usage** 10  **HINCRBY user:**001 **cost** 500 |
| --- |

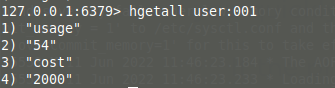
Output:



Let’s check the user:001 hash values.

| **HGETALL** **user**:001 |
| --- |

Output:



### Example 02 - Decrementing the unit count in shopping cart

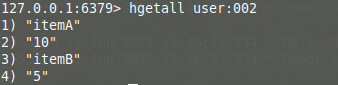
In shopping carts, we always need to perform decrement operations on the count of items when the user removes an item. Let’s store item counts for the *user:002*.

| HMSET user:002 **itemA** 10 **itemB** 5 |
| --- |

In this case, we have assigned itemA and itemB counts as 10 and 5 respectively. Let’s check the user:002 data.

| **HGETALL** **user**:002 |
| --- |

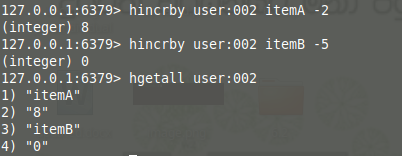
Output:



Let’s assume the *user:002* removes 2 itemAs and 5 itemBs. The HINCRBY command can be used to perform this decrement operation by passing -2 and -5 integers as increment arguments.

| HINCRBY user:002 **itemA** -2 HINCRBY user:002 **itemB** -5 |
| --- |

Output:



The HINCRBY command is powerful and efficient where both the increment and decrement operations can be performed using the same command with a signed increment argument. This command has constant time complexity. Hence, it is real fast to perform increment/decrement operations on hashes.

# Conclusion

Redis can store different types of data structures for a specific key. The hashes are one of the most popular data structures used in Redis databases. Hashes can be used to represent and manipulate simple objects and data effectively. Redis supports several hash operations. The HINCRBY is one of the popular commands to increment and decrement integer values for a given hash field. It supports 64-bit signed integers as the increment value. Hence, both the increment and decrement operations can be performed using the same command.