**Title:** Redis HEXISTS

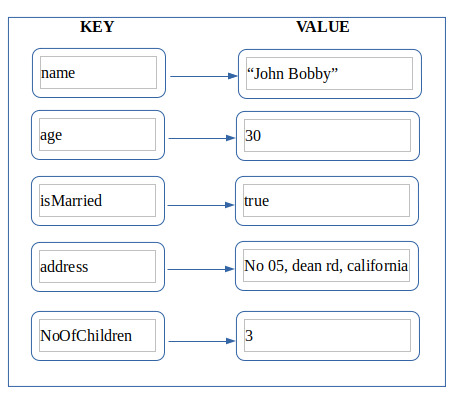
**Excerpt:** Redis hashes are one of the most popular data structures to use in Redis databases. Therefore, Redis supports different operations to manipulate hashes. One of the important commands is the HEXISTS command which checks for the availability of the given hash field at a given key.

**Permalink:** redis-hexists

**Category:** Redis

# Redis Hashes

The map is a widely-used data structure in the context of programming. It is a widely used technique to store key-value pairs. Each key maps to a certain value as shown in the following.



Redis hashes follow the same internal structure. But they will not allow you to store complex value types such as set, list, or hash. There are some ways to point the hash field values to other complex data types but it is not straightforward.

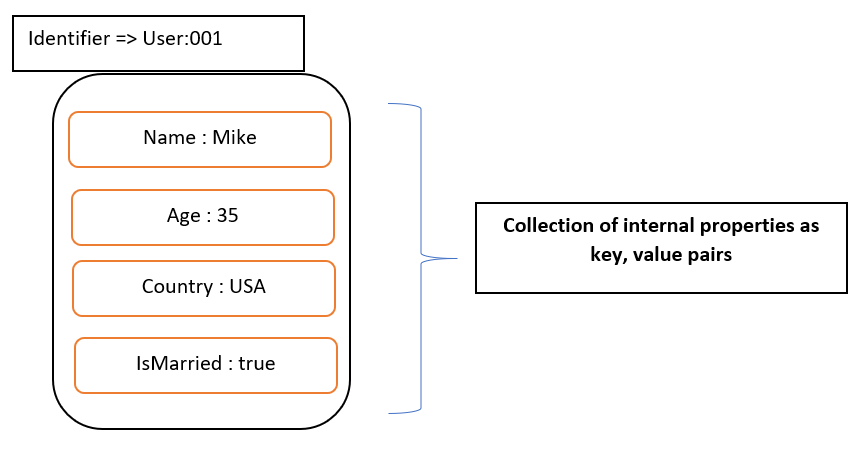
## Practical usage of Redis hashes

Redis hashes are very effective in two different scenarios.

1. Object representation
2. Store small data values compactly

### Object representation using Redis hashes

In programming, objects represent real-world entities with their states and behaviors. Usually, an object has an identifier(name) and internal properties as shown in the following.



Redis hashes are capable of storing this type of object in its storage. We know that Redis is key-value pair storage. Hence, the above object can be represented as shown in the following.

**Key** : User:001

**Value** : Internal properties as a Redis hash

The more detailed representation is shown in the following.

**Key** : User:001

**Value** : “Name” => “Mike”

“Age” => “35”

“Country” => “USA”

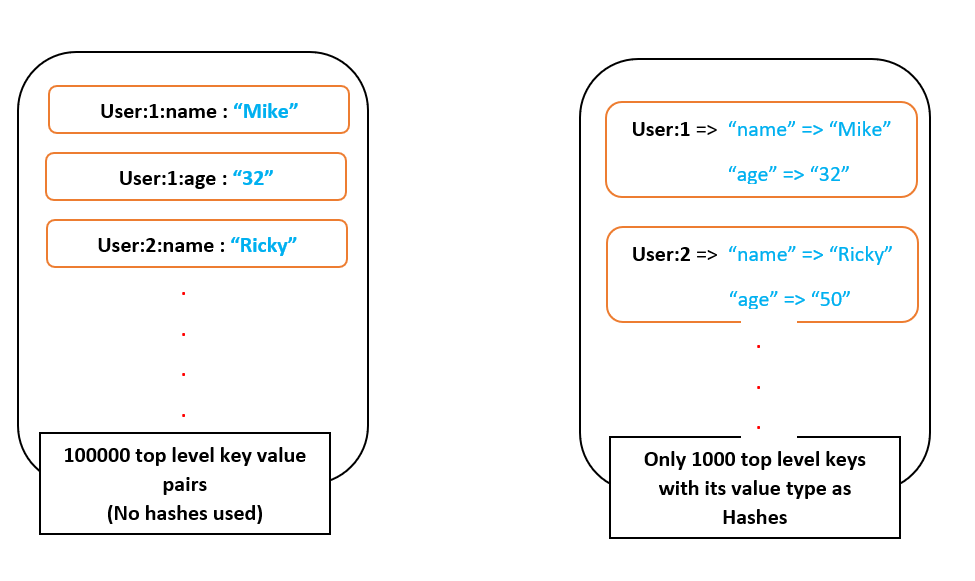
“IsMarried” => “true”

The green colored part is a Redis hash type value. In this case, we got four fields in our hash and the key is User:001.

### Store small data values compactly

Imagine we are going to store 100000 field-value pairs in a Redis database. The most typical way is to create 100000 top-level keys pointing to relevant values. This methodology is not going to be efficient as you might think. Hence, we can use hashes to optimize the performance.

There is a performance gain in hashes whenever a hash has a small number of fields, Redis increases the access efficiency by a considerable amount. We call this Redis’s small hash storage optimization. Therefore, we can keep 1000 hashes each with 100 internal fields as shown in the following.



The best thing about hashes is they can store up to 2^32-1 field-value pairs in plain text format. It is a great amount of data, more than 4 billion.

# Redis hash commands

Redis enables you to perform operations on hashes. There are around 15 hash commands available to use such as HSET, HEXISTS, HGET, HGETALL, etc. There are a couple atomic operations available to perform on numeric hash field values. It is the atomic increment and decrement on numeric values using HINCRBY, and HINCRBYFLOAT commands.

## The HEXISTS command

The HEXISTS command is used to check whether a given field is available in the particular hash at a specified key. This command has constant time complexity which is denoted by Big O(1). Therefore, it is very efficient to use.

### Syntax

| HEXISTS key\_name field\_name |
| --- |

**key\_name**  : This is the Redis top level key of this hash.

**field\_name**  : This is the hash field name.

### Command Return Value

The HEXISTS command returns an integer value of 1 or 0 based on the command evaluation results. It returns 1 if the field is available in the specified hash. If the field or hash key is not available, it returns 0.

### Practical usage of HEXISTS command

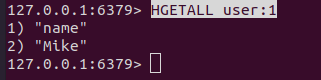
Let’s create a hash with one field and check how the command works. We can use the HSET command to create a hash with one field-value pair.

| HSET user:1 name Mike |
| --- |

Next, we will check whether the hash has been added to the Redis database successfully. The HGETALL command can be used to fetch all the field-value pairs for a given hash key.

| **HGETALL** **user**:1 |
| --- |

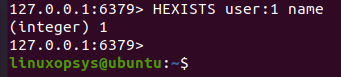
Output:



Now let’s check the HEXISTS command behavior on this hash.

| HEXISTS user:1 name |
| --- |

In this scenario, the user:1 is the hash key. The name is the hash field name. Since the name field is available in the user:1 hash, it should return 1.



Let’s check the command with a field that is not available in the given hash.

| **HEXISTS** **user**:1 **age** |
| --- |

In this case, the age field is not available in the user:1 hash. Hence, the output is 0.



In this way, you can manipulate hash fields for their existence at a given hash key. This is a very time-efficient operation and helpful in object manipulations in Redis databases. It takes constant time to check the existence of the 1st field or 100th field. Therefore, it is recommended to use Redis hashes whenever you need to store objects or simple data.

# 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 HEXISTS is one of the popular commands to check the existence of a given hash field in a hash key.