Arrays asociativos

Tiempo estimado: 15min

Ésta es la última lección relacionada con los tipos de datos nativos de Redis. Tras presentar las cadenas, las listas y los conjuntos, le ha llegado el momento a los *arrays* asociativos. Un tipo de datos indispensable para almacenar datos estructurados como, por ejemplo, registros u objetos, tal como podemos hacer con otros motores de bases de datos.

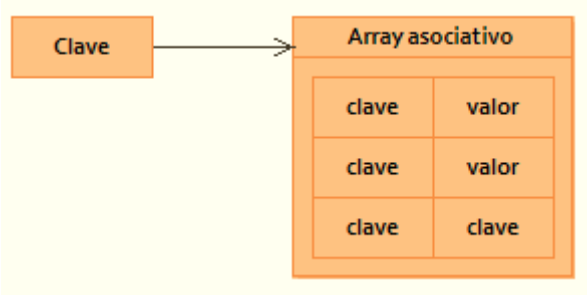
La lección se estructura en tres puntos básicos. Comenzamos presentando el concepto de *array* asociativo. Y a continuación, mostramos las distintas operaciones que podemos llevar a cabo con claves cuyo valor es un *array* asociativo como, por ejemplo, la creación, la supresión, la comprobación de existencia, etc. Finalmente, se presenta el módulo rxhashes que amplia el número de comandos relacionados con este tipo de datos.

Al finalizar la lección, el estudiante sabrá:

- Qué es un *array* asociativo.
- Cómo operar con pares clave-valor de tipo *array* asociativo.

Introducción

Un array asociativo (*hash*) es una estructura de datos formada por campos. Un campo (*field*) no es más que un par clave-valor, donde la clave representa su nombre y el valor, su valor. Tanto las claves como los valores deben ser de tipo cadena. Es ideal para almacenar objetos de datos de tipo registro, pero sin olvidar que el valor siempre debe ser de tipo cadena. Con ellos podemos almacenar información relacionada con un determinado objeto de datos como, por ejemplo, un pedido, una factura, un empleado, un producto, un servicio, un cliente o un proveedor.



Un array asociativo es como una estructura de C/C++, un diccionario de Python, un objeto de JavaScript o un documento de MongoDB o CouchDB.

Entre las operaciones que podemos realizar con pares clave-valor de este tipo, encontramos: la supresión de campos, la comprobación de existencia de campos, la obtención de valores de campos, etc. Todas ellas vamos a presentarlas a continuación.

Creación de arrays asociativos

Para crear un par clave-valor de tipo *array*, hay que utilizar los comandos siguientes:

```
HSET clave campo valor
HSETNX clave campo valor
HMSET clave campo valor campo valor campo valor...
```

HSET añade el campo a la clave especificada de la base de datos. Si la clave no existe, la crea como *array* asociativo y le añade el campo indicado. Si la clave existe y el campo también, cambia su valor al especificado. Por su parte, HSETNX sólo añade el campo si todavía no existe. Al igual que HSET, si la clave no existe, la crea. Finalmente, HMSET es como HSET pero con varios campos.

Veamos unos ejemplos:


```
127.0.0.1:6379> HSET author:1 name "Michael Crichton"
(integer) 1
127.0.0.1:6379> HSET author:1 birthYear 1942
(integer) 1
127.0.0.1:6379> HGETALL author:1
1) "name"
2) "Michael Crichton"
3) "birthYear"
4) "1942"
127.0.0.1:6379> HMSET author:2 name "Robert Harris" birthYear 1957
OK
127.0.0.1:6379> HGETALL author:2
1) "name"
2) "Robert Harris"
3) "birthYear"
4) "1957"
127.0.0.1:6379>
```

Existencia de campos

Para comprobar si un par clave-valor de tipo *array* contiene un determinado campo, se utiliza el comando HEXISTS:

```
HEXISTS clave campo
```

Ejemplo:

```
127.0.0.1:6379> HEXISTS author:1 name
(integer) 1
127.0.0.1:6379> HEXISTS author:1 noexiste
(integer) 0
127.0.0.1:6379>
```

Consulta de arrays asociativos

El acceso a un *array* asociativo se puede hacer a dos niveles: por un lado, a todo el valor, esto es, al contenido de todo el *array* asociativo; por otro lado, a nivel de uno o más campos individualmente.

Mediante HGETALL consultamos todos los campos de una clave *array*:

```
HGETALL clave
```

Para consultar campos específicos, podemos usar HGET y HMGET:

```
HGET clave campo
HMGET clave campo campo campo...
```

HGET sólo accede a un único campo, mientras que HMGET permite el acceso a varios.

A continuación, unos ejemplos ilustrativos:

```
127.0.0.1:6379> HGETALL author:3
1) "name"
2) "Neal Stephenson"
3) "birthYear"
4) "1959"
127.0.0.1:6379> HGET author:3 name
"Neal Stephenson"
127.0.0.1:6379> HGET author:3 birthYear
"1959"
127.0.0.1:6379> HGET author:3 campoInexistente
(nil)
127.0.0.1:6379> HMGET author:3 name birthYear campoInexistente
1) "Neal Stephenson"
2) "1959"
3) (nil)
127.0.0.1:6379>
```

HGET devuelve el valor del campo especificado, el cual recordemos debe ser de tipo cadena. Si el campo no existe, devuelve nil. HMGET devuelve nil en los campos inexistentes. Finalmente, si la clave no existe: HGET devuelve nil; HGETALL, una lista vacía; y HMGET una lista con tantos elementos como campos se ha especificado, con sus valores a nil.

Es posible conocer los nombres de los campos de un *array* e incluso sus valores:


```
HKEYS clave
HVALS clave
```
Ambos comandos devuelven una lista. El primero con los nombres de los campos del *array* y el segundo con los valores de los campos. He aquí unos ejemplos ilustrativos:

```
127.0.0.1:6379> HKEYS author:4
1) "name"
2) "birthYear"
127.0.0.1:6379> HVALS author:4
1) "Frederick Forsyth"
2) "1938"
127.0.0.1:6379>
```
El comando HLEN devuelve el número de campos que tiene el *array*:

```
HLEN clave
```

Supresión de campos

Para suprimir un determinado campo, se utiliza el comando HDEL:

```
HDEL clave campo
HDEL clave campo campo campo...
```
El comando devuelve el número de campos suprimidos. Si el campo no existe, no se contabilizará entre los suprimidos.

Operaciones aritméticas sobre campos

¿Recuerda el comando INCRBY de las cadenas, aquel con el que incrementamos el valor de una clave cadena que contiene como valor un número? Redis proporciona dos comandos para incrementar valores de campos que contienen números, ideales para almacenar contadores como, por ejemplo, el número de accesos a una página web. Estos comandos son:

```
HINCRBY clave campo incremento
HINCRBYFLOAT clave campo incremento
```
Si el valor de incremento es negativo, realizará un decremento. Si el campo no existe, lo creará a cero y, a continuación, le aplicará el incremento. Los comandos devuelven el valor resultante.

Como no podía ser de otra manera, un ejemplo ilustrativo:

```
127.0.0.1:6379> HKEYS web:1
1) "url"
2) "counter"
127.0.0.1:6379> HINCRBY web:1 counter 1
(integer) 124
127.0.0.1:6379>
```

Módulo rxhashes

El módulo rxhashes proporciona un comando para trabajar con pares cuyo valor es un *array* asociativo.

HGETSET

Mediante el comando HGETSET podemos cambiar el valor de un campo, obteniendo el valor que tiene antes del cambio:

```
HGETSET clave campo valor
```
Ejemplo:

```
127.0.0.1:6379> HGETALL band:the-minus-5
1) "name"
2) "The Minus 5"
3) "origin"
4) "US"
5) "year"
6) "1993"
127.0.0.1:6379> HGETSET band:the-minus-5 origin USA
"US"
127.0.0.1:6379> HGETALL band:the-minus-5
```



```
1) "name"
2) "The Minus 5"
3) "origin"
4) "USA"
5) "year"
6) "1993"
127.0.0.1:6379>
```

