Conjuntos

Tiempo estimado: 25min

En las lecciones anteriores, presentamos los tipos cadena y lista. El tipo de datos cadena es simple y permite almacenar un único valor. Mientras que el tipo de datos lista es compuesto y permite almacenar varios. En esta lección, vamos a introducir el tipo de datos conjunto, que también permite almacenar varios valores de tipo cadena. Redis proporciona dos tipos de datos conjunto: el conjunto propiamente dicho y el conjunto ordenado.

Primero, presentamos el concepto de conjunto, tal como lo aprendimos en la escuela. A continuación, detallamos el tipo conjunto (desordenado) y el módulo rxsets. Finalmente. el tipo conjunto ordenado y el módulo rxzsets.

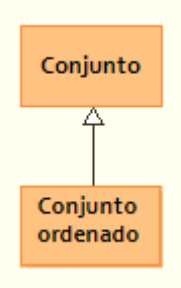
Al finalizar la lección, el estudiante sabrá:

- Cuáles son los dos tipos de datos relacionados con los conjuntos.
- Cómo crear pares clave-valor de tipo conjunto.
- Cómo operar con valores de tipo conjunto.

Introducción

Un conjunto (*set*) es una colección de elementos de tipo cadena. Los conjuntos no presentan valores duplicados. No puede haber dos o más elementos con el mismo valor. Si se añade al conjunto un valor ya existente, no lo duplicará, se quedará con la copia actual y no añadirá el duplicado.

Atendiendo a si los conjuntos mantienen sus elementos ordenados, los clasificamos en ordenados y desordenados.



Un conjunto desordenado o simplemente conjunto es aquel que no tiene ordenados sus elementos. No se puede añadir un elemento en una determinada posición. Redis decide dónde ubicarlo según le sea mejor a él para resolver lo más rápidamente posible las consultas. Por su parte, un conjunto ordenado (*sorted set*) mantiene los valores ordenados según un criterio.

Conjuntos (desordenados)

Recordemos que los conjuntos por definición no mantienen ordenados los valores. A continuación, se presenta las operaciones que podemos realizar con pares clave-valor de tipo conjunto (desordenado).

Añadidura de elementos

Para añadir un nuevo elemento o valor al conjunto, se utiliza SADD:

```
SADD clave valor
SADD clave valor valor valor...
```

El comando devuelve el número de valores añadidos. Recordemos que los conjuntos no pueden tener valores duplicados. Si el valor no existe, SADD devolverá su añadidura; pero si existe, devolverá que no lo ha añadido. Si la clave no existe, la creará vacía y, entonces, le añadirá los valores.

No olvidemos que los conjuntos se encuentran desordenados. La añadidura no significa que el valor se añada al final. Simplemente, se añade. Dónde lo ubique Redis, es algo en lo que no podemos influir.

Acceso a elementos

Como los conjuntos no se encuentran ordenados, no existe ningún comando para acceder al valor de una determinada posición. En su lugar, tenemos que comprobar si el conjunto presenta un determinado valor, conocido en Redis como miembro (*member*):

```
SISMEMBER clave valor
```

El comando SISMEMBER devuelve 1 si el valor se encuentra en el conjunto y 0 en caso contrario.

A continuación, un ejemplo ilustrativo:

```
127.0.0.1:6379> SISMEMBER conjunto uno
(integer) 1
127.0.0.1:6379> SISMEMBER conjunto noexiste
(integer) 0
127.0.0.1:6379>
```

Cuando necesitamos acceder a todos los valores o miembros del conjunto, podemos utilizar SMEMBERS:

```
SMEMBERS clave
```

Ejemplo:

```
127.0.0.1:6379> SMEMBERS conjunto
1) "dos"
2) "uno"
3) "tres"
127.0.0.1:6379>
```

Mientras que si lo que deseamos es que Redis nos devuelva uno o más valores aleatorios, utilizaremos SRANDMEMBER:

```
SRANDMEMBER clave
SRANDMEMBER clave número
```

Longitud del conjunto

Para conocer el número de elementos que tiene el conjunto, se utiliza el comando SCARD:

```
SCARD clave
```

Supresión de elementos

Para suprimir elementos del conjunto disponemos de dos comandos, SREM y SPOP:

```
SREM clave valor
SREM clave valor valor valor...
SPOP clave
SPOP clave número
```

SREM suprime los valores indicados, devolviendo el número de supresiones realizadas. Por su parte, SPOP selecciona uno o más valores aleatoriamente, los suprime y los devuelve al usuario. Cuando no especificamos ningún número de valores, SPOP realizará una única supresión.

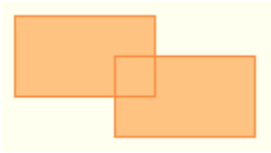
Otra posibilidad es suprimir un valor de un conjunto y llevarlo a otro. Para esta operación, disponemos del comando SMOVE:

```
SMOVE origen destino valor
```

Si el conjunto destino no existe, lo creará como vacío y le añadirá el valor. La operación sólo se lleva a cabo si existe el conjunto origen y contiene el valor indicado.

Unión de elementos

La unión (*union*) es la operación mediante la cual se devuelve un conjunto con los elementos de dos o más conjuntos.



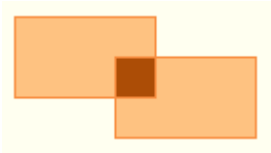
Si tenemos un conjunto formado por 1, 2, 3 y 4 y otro por 2, 4 y 6, la unión devuelve un conjunto formado por 1, 2, 3, 4 y 6. Esta operación puede realizarse mediante los comandos SUNION y SUNIONSTORE:

```
SUNION clave clave
SUNION clave clave clave...
SUNIONSTORE destino clave clave
SUNIONSTORE destino clave clave clave...
```

SUNION devuelve el resultado. En cambio, SUNIONSTORE lo almacena en la clave destino, devolviendo el número de elementos del conjunto resultado.

Intersección de elementos

La intersección (*intersection*) es la operación con dos o más conjuntos que devuelve los elementos o valores comunes en todos ellos.



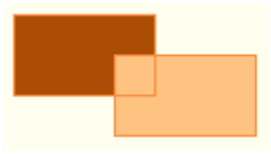
Por ejemplo, si tenemos un conjunto formado por 1, 2, 3 y 4 y otro por 2, 4 y 6, la intersección de ambos resultará en el conjunto formado por los elementos 2 y 4. La operación se puede realizar mediante los comandos SINTER y SINTERSTORE:

```
SINTER clave clave
SINTER clave clave clave...
SINTERSTORE destino clave clave
SINTERSTORE destino clave clave clave...
```

SINTER devuelve el resultado. Mientras que SINTERSTORE lo almacena en la clave destino, devolviendo el número de elementos del conjunto resultado.

Complemento de elementos

El complemento (*complement*) es la operación que devuelve los elementos que se encuentran en un conjunto pero no en otro. El orden de los factores altera el producto. No es lo mismo comprobar A-B que B-A.



Si tenemos un conjunto formado por 1, 2, 3 y 4 y otro por 2, 4 y 6, el complemento del primero menos el segundo resultará en 1 y 3, mientras que el del segundo menos el primero será 6. Se realiza mediante los comandos SDIFF y SDIFFSTORE:

```
SDIFF clave clave
SDIFF clave clave clave…
SDIFFSTORE destino clave clave
SDIFFSTORE destino clave clave clave…
```

Al igual que en los anteriores, SDIFF devuelve el conjunto resultado al usuario, mientras que SDIFFSTORE lo almacena en la clave destino.

Módulo rxsets

El módulo rxsets proporciona un comando adicional para trabajar con conjuntos (desordenados).

MSISMEMBER

Mediante el comando MSISMEMBER se puede comprobar si un determinado elemento (o miembro) se encuentra en varios pares clave-valor de tipo conjunto:

```
MSISMEMBER clave clave … elemento
```

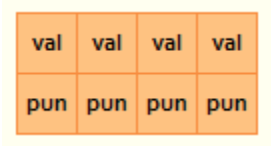
El comando devuelve el número de pares que presentan el elemento.

Veamos un ejemplo ilustrativo:

```
127.0.0.1:6379> SMEMBERS conj1
1) "cuatro"
2) "tres"
3) "dos"
4) "uno"
127.0.0.1:6379> SMEMBERS conj2
1) "cinco"
2) "cuatro"
3) "tres"
4) "seis"
127.0.0.1:6379> MSISMEMBER conj1 conj2 uno
(integer) 1
127.0.0.1:6379> MSISMEMBER conj1 conj2 tres
(integer) 2
127.0.0.1:6379> MSISMEMBER conj1 conj2 cero
(integer) 0
127.0.0.1:6379>
```

Conjuntos ordenados

Un conjunto ordenado (*sorted set*) es aquel que mantiene los valores ordenados atendiendo a un segundo valor conocido formalmente como puntuación (*score*), a diferencia del conjunto que no los mantiene en ningún orden particular.



Las puntuaciones se pueden repetir en el conjunto, no así los valores de los elementos. Los elementos, con valores distintos, pero misma puntuación, se ordenan según el valor.

Añadidura de elementos

Para añadir un nuevo elemento, valor o miembro a un conjunto ordenado, se utiliza ZADD:

```
ZADD clave puntuación valor
ZADD clave puntuación valor puntuación valor puntuación valor...
```

El comando devuelve el número de valores añadidos. Recordemos que los conjuntos no pueden tener valores duplicados. Si el valor no existe, ZADD devolverá su añadidura; pero si existe, devolverá que no lo ha añadido. Si la clave no existe, la creará vacía y, entonces, le añadirá los valores.

No olvidemos que los conjuntos se encuentran ordenados atendiendo a la puntuación, de ahí que tengamos que indicar tanto el valor como la puntuación.

El comando ZADD proporciona varias opciones que se puede especificar entre el nombre de clave y la puntuación del primer elemento a añadir. Podemos indicar cómo debe actuar si ya existe el elemento:

- XX indica que actualice su puntuación, pero que no añada el elemento si éste no existe.
- NX, por su parte, indica que si el elemento ya existe, no haga nada, deje su puntuación como está, omitiendo la indicada en el comando.

La opción XX se utiliza para actualizar elementos existentes, no inserta nuevos; y NX, se utiliza cuando deseamos que simplemente añada, sin actualizar. Si no indicamos ninguna de las dos, el comando se comportará como sigue: si el elemento no existe, lo añadirá; si existe, le cambiará su puntuación.

Lectura de elementos

A diferencia de los conjuntos desordenados, donde los miembros no se pueden acceder por su posición, pues no tienen ninguna específica, los conjuntos ordenados sí pueden. Recordemos que su posición dentro del conjunto viene dada por su puntuación, pero no se accede por su puntuación sino por su posición.

El comando ZRANGE se utiliza para consultar los elementos entre dos posiciones, no así entre dos puntuaciones:

```
ZRANGE clave inicio fin
ZRANGE clave inicio fin WITHSCORES
```

Si no indicamos WITHSCORES, el comando devolverá los valores sin sus puntuaciones.

Para conocer la posición de un determinado elemento, disponemos del comando ZRANK:

```
ZRANK clave valor
```

Para conocer la puntuación de un determinado elemento, de ZSCORE:

```
ZSCORE clave valor
```

Si deseamos acceder a los elementos entre un rango de puntuaciones, usaremos ZRANGEBYSCORE:

```
ZRANGEBYSCORE clave mínima máxima
ZRANGEBYSCORE clave mínima máxima WITHSCORES
ZRANGEBYSCORE clave mínima máxima LIMIT primero cuenta
ZRANGEBYSCORE clave mínima máxima WITHSCORES LIMIT primero cuenta
```

Cuando no indicamos más que el rango de puntuaciones, el comando devuelve sólo los valores de los elementos sin sus puntuaciones. Para obtener también las puntuaciones, hay que indicar WITHSCORES. Por otra parte, la opción LIMIT lo que hace es limitar el número máximo de elementos a devolver, la cuenta, así como a partir de que elemento comenzar, el primero. Los anteriores al primero serán omitidos. Es similar a la cláusula LIMIT de SQL. Se puede utilizar los valores predefinidos -inf y +inf para indicar infinito en los lados negativo y positivo, respectivamente.

Los conjuntos ordenados mantienen ordenados sus valores por la puntuación. Si dos o más elementos tienen la misma puntuación, el conjunto los ordenará según su valor. Así pues, si necesitamos mantener un conjunto ordenado por valor, en vez de por puntuación, bastará con añadir todos sus elementos con la misma puntuación para forzar así a que el conjunto se ordene por valor. Cuando tenemos un conjunto ordenado desde esta perspectiva, podemos utilizar el comando ZRANGEBYLEX para obtener los elementos cuyos valores se encuentren entre los indicados:

```
ZRANGEBYLEX clave inicio fin
ZRANGEBYLEX clave inicio fin LIMIT primero cuenta
```

Ahora, el rango se especifica con respecto a los valores, a diferencia de ZRANGEBYSCORE que lo hace respecto a las puntuaciones. Redis soporta dos especificadores para indicar si debe incluirse los extremos del rango en el resultado: mediante [indicamos que sí; con (que no lo haga. Por ejemplo, [valor indica que si existe el valor, debe incluirse; mientras que con (valor, no debe hacerlo.

Los comandos anteriores tienen un equivalente para devolver los resultados en orden inverso:

Comando	Comando inverso
ZRANK	ZREVRANK
ZRANGE	ZREVRANGE
ZRANGEBYSCORE	ZREVRANGEBYSCORE
ZRANGEBYLEX	ZREVRANGEBYLEX

Longitud del conjunto

Para conocer el número de elementos que tiene el conjunto ordenado, se utiliza el comando ZCARD:

```
ZCARD clave
```

Existe un segundo comando que indica el número de elementos que tienen su puntuación dentro de un rango:

```
ZCOUNT clave inicio fin
```


Supresión de elementos

La supresión de elementos la podemos hacer a partir de valores específicos con ZREM, de su posición en el conjunto mediante ZREMRANGEBYRANK, de rangos de valores con ZREMRANGEBYLEX o de rango de puntuaciones mediante ZREMRANGEBYSCORE:

```
ZREM clave valor
ZREM clave valor valor valor...
ZREMRANGEBYRANK clave inicio fin
ZREMRANGEBYLEX clave inicio fin
ZREMRANGEBYSCORE clave mínimo máximo
```

Unión de elementos

Para realizar la unión de conjuntos ordenados, utilizar ZUNIONSTORE:

```
ZUNIONSTORE destino númeroClaves clave clave clave...
```

Intersección de elementos

La intersección se realiza mediante ZINTERSTORE:

```
ZINTERSTORE destino númeroClaves clave clave clave…
```

Módulo rxzsets

Por su parte, el módulo rxzsets añade comandos adicionales para trabajar con conjuntos ordenados.

ZPOP y ZREVPOP

Los comandos ZPOP y ZREVPOP suprimen elementos de un conjunto desordenado atendiendo a la puntuación. ZPOP suprime el de menor puntuación; mientras que ZREVPOP, el de mayor puntuación. Sus sintaxis son las siguientes:

```
ZPOP clave
ZPOP clave WITHSCORE

ZREVPOP clave
ZREVPOP clave WITHSCORE
```

Los comandos devuelve el elemento suprimido sin su puntuación. Si indicamos la cláusula WITHSCORE devolverá su puntuación. Si no existe el par clave-valor, devuelve nil.

Ejemplos ilustrativos:

```
127.0.0.1:6379> ZRANGE conj 0 -1 WITHSCORES
1) "uno"
2) "1"
3) "dos"
4) "2"
5) "tres"
6) "3"
7) "cuatro"
8) "4"
9) "cinco"
10) "5"
127.0.0.1:6379> ZPOP conj
1) "uno"
127.0.0.1:6379> ZPOP conj WITHSCORE
1) "dos"
2) "2"
127.0.0.1:6379> ZRANGE conj 0 -1 WITHSCORES
1) "tres"
2) "3"
3) "cuatro"
4) "4"
5) "cinco"
6) "5"
127.0.0.1:6379>
```


MZRANK y MZREVRANK

Recordemos que el comando ZRANK devuelve la posición de un determinado elemento. Mediante MZRANK y MZREVRANK se puede consultar varios elementos:

```
MZRANK clave elemento elemento elemento …
MZREVRANK clave elemento elemento elemento …
```

Veamos unos ejemplos:

```
127.0.0.1:6379> ZRANGE conj 0 -1
1) "uno"
2) "dos"
3) "tres"
4) "cuatro"
127.0.0.1:6379> ZRANK conj dos
(integer) 1
127.0.0.1:6379> ZRANK conj tres
(integer) 2
127.0.0.1:6379> MZRANK conj dos tres
1) (integer) 1
2) (integer) 2
127.0.0.1:6379> MZREVRANK conj dos tres
1) (integer) 2
2) (integer) 1
127.0.0.1:6379>
```

Observe que MZREVRANK devuelve las posiciones de los elementos en orden inverso a su aparición en el comando.

