Bases de Datos No SQL

1. Introducción a las BD NoSQL

Universidad
Rey Juan Carlos

Índice

Bibliografía

- Maxwell Dayvson Da Silva; Hugo Lopes Tavares (2015). **Redis Essentials.** Packt Publishing
- Vinoo Das (2015). **Learning Redis.** Packt Publishing
- Josiah L. Carlson(2013). **Redis in Action.** Manning Publications

- Xun Wu et al (2018). **Seven NoSQL Databases in a Week.** Pack Publishing.
- Dan Sullivan (2015). **NoSQL for Mere Mortals.** Addison-Wesley Professional
- Dan McCreary & Ann Kelly (2014). **Making Sense of NoSQL. A guide for managers and the rest of us.** Manning Publications
- Gaurav Vaish (2013). **Getting Started with NoSQL. Your guide to the world and technology of NoSQL.** PACKT Publishing
- Joe Celko (2014). **Joe Celko's Complete guide to NoSQL. What every SQL professional needs to know about Non- Relational Databases.** Morgan Kaufmann

Bases de Datos Clave-Valor

- Son las BD NoSQL **más simples.**
- Se basan en almacenar **datos** con identificadores conocidos como **claves** (similares a arrays).
- En una BD Clave-Valor, **valor** tiene un identificador único , que es la **clave**.
- Las **claves son únicas** dentro del espacio de nombres (que puede denominarse agrupación de datos, base de datos, ...)

Bases de Datos Clave-Valor

- **Claves:**
  - Identificadores asociados con valores.
  - Análogos a las etiquetas de los equipajes:
    - Permiten encontrar tu equipaje.
    - Si los equipajes se organizan por número de etiqueta, estos son fácil de encontrar.
    - La etiqueta se genera al facturar, y podría tener un número de vuelo y un número secuencial para cada equipaje (simplificación, vuelos iguales).

Bases de Datos Clave-Valor

- **Diseño de Claves**

Ejemplo:

- Sitio web en el que se necesita almacenar CLIENTES:
  - Para **cada cliente** : número de cuenta, nombre, número de productos, tipo de cliente.
- Se podría asignar **a cada cliente un número secuencial**, que se incluye en la clave (p.e. Cliente -> 1)
- Se generarían las claves para la BD Clave – Valor de la siguiente forma:

Bases de Datos Clave-Valor

- Diseño de Claves
  - La aproximación se complica si queremos guardar también información de **otras entidades**
    - Por ejemplo: clientes (cli), almacenes (alm)
  - Dado que podemos querer recoger características con el mismo nombre, añadiremos a las claves un **prefijo** con el **nombre de la entidad**
    - Por ejemplo: cli, alm
      - cli1.NúmerodeCuenta, cli1.Nombre, cli1.Dirección
      - cli2.NúmerodeCuenta, cli2.Nombre, cli2.Dirección
      - alm1.Número, alm1.Dirección

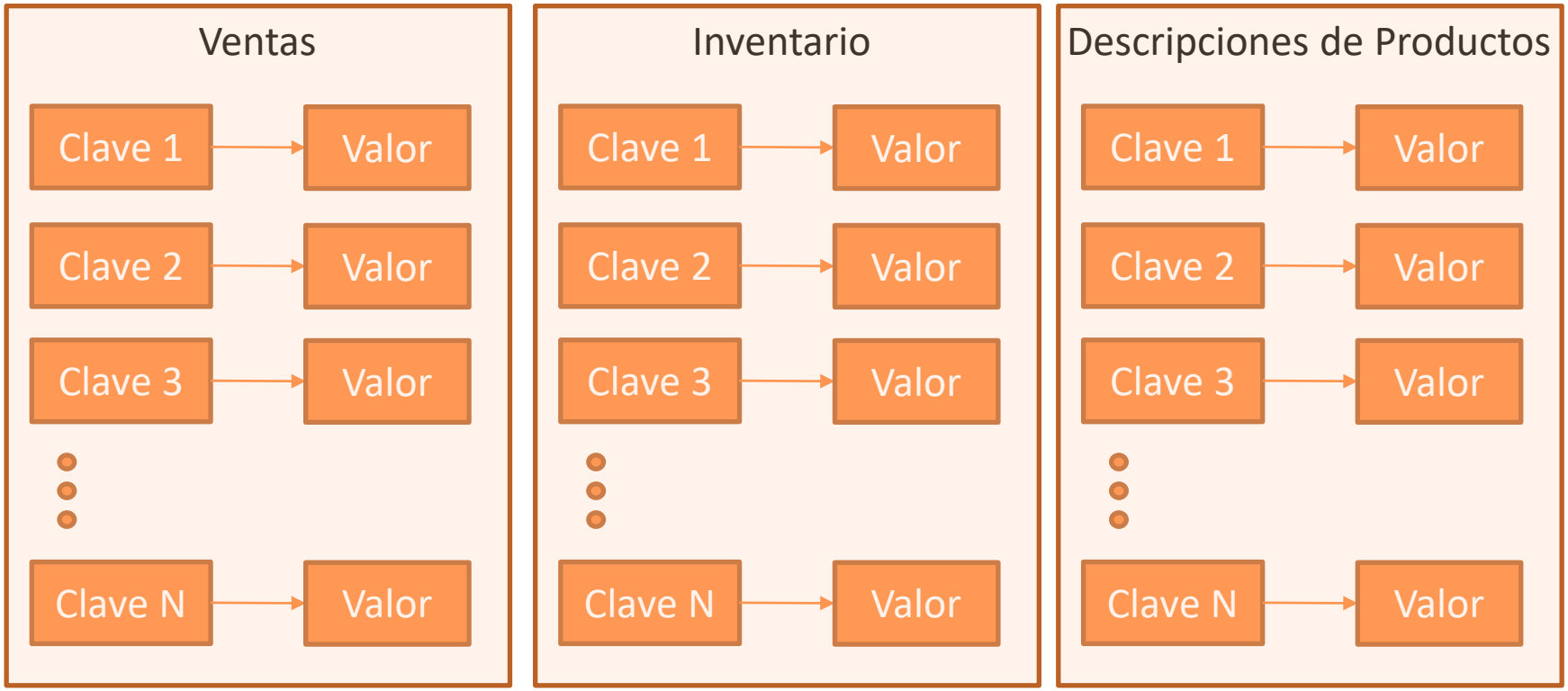
Bases de Datos Clave-Valor

- Claves
  - Las **claves deben ser únicas** dentro de un **mismo espacio de nombres**.
  - Un espacio de nombres se podría corresponder con una base de datos completa.
  - Algunas Bases de Datos Clave-Valor permiten la existencia de **varios espacios de nombres** en la **misma base de datos**.

  ⇨ Similar a los **esquemas en bases de datos** relacionales.

Bases de Datos Clave-Valor

Base de datos Clave – Valor con **tres espacios de nombres diferentes**:



Bases de Datos Clave-Valor

- **Valores**
  - Son **datos** que se almacenan con las claves. Pueden ser cadenas de caracteres, números, imágenes, objetos binarios…
  - Las Bases de Datos Clave-Valor proporcionan a los desarrolladores una **gran flexibilidad** para almacenar valores, con distintos tipos y tamaños. **No se comprueba el tipo de datos** de los valores.
  - Dado que las Bases de Datos Clave-Valor permiten **cualquier tipo de datos**, es importante que los desarrolladores implementen las **comprobaciones en sus aplicaciones**.

Bases de Datos Clave-Valor

- **Uso de BD Clave-Valor:**
  - Cuando la **facilidad de almacenamiento y recuperación** son **más importantes** que la ***organización de los datos*** en estructuras más complejas, como tablas o redes.
  - Permite implementar de forma sencilla estructuras de datos similares a tablas utilizando una **convención en el nombre de las claves**:

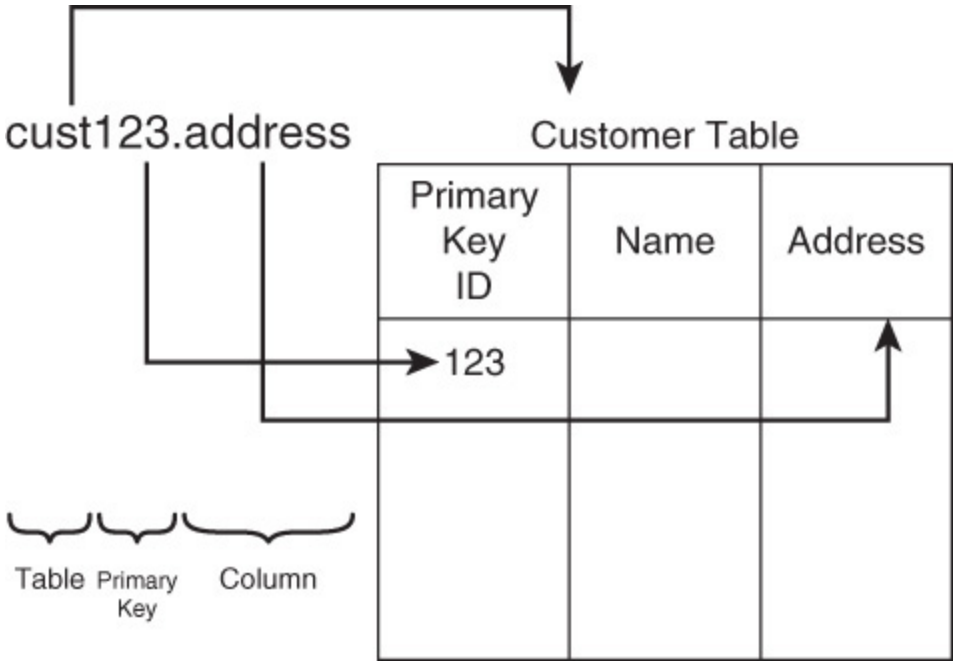
    nombre tabla: valor de clave primaria: nombre atributo

    Cliente:00001:nombre
    Cliente:00001:apellido1
    Cliente:00001:apellido2
    Cliente:00001:dirección
    Cliente:00001:ciudad
    Cliente:00001:teléfono

Bases de Datos Clave-Valor

- **Diferencias entre Bases de Datos Clave-Valor y Relacionales**
  - Las Bases de Datos Clave-Valor se modelan sobre **principios <u>mínimos</u>** para **almacenar y recuperar datos**.
  - A diferencia de las Bases de Datos Relacionales:
    - **No hay tablas**, por tanto no existen propiedades asociadas a las tablas, como **columnas y restricciones** sobre ellas.
    - **No** son necesarios los **joins**, por tanto no hay claves ajenas.
    - **No soportan un lenguaje de consulta rico**, como SQL.
  - Algunas BD Clave-Valor soportan **espacios de nombres separados** para implementar algo **análogo a los esquemas** de las Bases de Datos relacionales.
    - Si lo combinamos con una convención de nombres de clave (p.e. anteponiendo el nombre de la "entidad")
  - Existen paralelismos entre la **convención de nombres de claves** y las tablas, claves primarias y columnas:
    - La clave 'cli123.dirección' equivaldría a una tabla CLI, con una columna llamada Dirección y un identificador (clave primaria) de 123.

Bases de Datos Clave-Valor



Bases de Datos Clave-Valor

- Al diseñar una aplicación que utiliza una BD Clave-Valor, hay que tener en cuenta:
  - Cómo estructurar las claves
    - **Claves bien diseñadas** pueden hacer que el código de las aplicaciones sea **más legible y mantenible**.
  - Qué información capturar en los valores
    - Capturar los datos correctos es importante para alcanzar los **requisitos funcionales** y para asegurar un **rendimiento** adecuado de la aplicación.
  - Cómo introducir abstracciones (patrones de diseño para usar en las aplicaciones) que ayuden a crear estructuras de más alto nivel que las simples pares Clave-Valor

Bases de Datos Clave-Valor. Redis

- **Redis** es una Base de Datos NoSQL del tipo clave-valor.
- https://redis.io/

- Redis almacena **todos los datos en memoria**: operaciones lectura y escritura muy rápidas.
- Los datos también se pueden hacer persistentes en disco *(snapshot)* o ir añadiendo a un fichero de log (*command log*).
- REDIS es "single threaded", es decir, siempre ejecuta únicamente un comando.
- Los comandos son atómicos.
- Permite trabajar con **múltiples bases de datos identificadas por un número secuencial** (por defecto, conexión a la BD 0). Se puede cambiar a otra BD utilizando el comando SELECT 1.
- Actualmente Redis es usado por: Twitter, Instagram, GitHub, Flickr, ….

Bases de Datos Clave-Valor. Redis

ALGUNAS SENTENCIAS:

- redis-server es el actual almacenamiento de datos de REDIS.
- redis-cli es el cliente de REDIS (interfaz de línea de comandos).
- Por defecto: puerto 6379

http://try.redis.io/

- Comandos **SET Y GET**:

```
$ redis-cli
127.0.0.1:6379> SET filosofo “Socrates"
OK
127.0.0.1:6379> GET filosofo
“Socrates"
127.0.0.1:6379> GET Filosofo
(nil)
127.0.0.1:6379>
127.0.0.1:6379 > SET filosofo "Socratin"
OK
127.0.0.1:6379> GET filosofo
"Socratin"
```

Claves son Case-sensitive

Comando **SETNX** -> SET if Not eXists

Si existe, se sobreescribe

```
127.0.0.1:6379 >  SETNX filosofo "Socratin"
(integer) 0
127.0.0.1:6379 > GET filosofo
"Socrates"
```

Bases de Datos Clave-Valor. Redis

Si ya existe la clave, se sobreescribe su valor, si no se usa MSETNX.

- Comandos **MSET Y MGET**:

  $ redis-cli

  127.0.0.1:6379> **MSET** *filosofo1* "Platon" *filosofo2* "Aristoteles" filosofa "Hipatia"

  OK

  127.0.0.1:6379> **MGET** *filosofo1 filosofo2*

  1) "Platon"

  2) "Aristoteles"

  127.0.0.1:6379>

- Comando **DEL**:

  $ redis-cli

  127.0.0.1:6379> **DEL** *filosofo1 filosofo2*

  (integer) 2

Bases de Datos Clave-Valor. Redis

- Comando **KEYS**: para obtener todas las claves almacenadas que coincidan con el *patrón*

```
$ redis-cli
127.0.0.1:6379> KEYS f*
1) "filosofo2"
2) "filosofo1"
3) "filosofo"
```

Posibles patrones:

h?llo → hello, hallo, hxllo *(exactamente un carácter)*
h*llo → hllo, heeeello *(0 .. n caracteres)*
h[ae]llo → hello hallo, pero NO hullo *(cualquiera de los caracteres entre corchetes)*
h[^e]llo → hallo, hbllo, ... pero NO hello *(un carácter que no sea "e")*
h[a-b]llo → hallo and hbllo *(cualquiera de los caracteres comprendido en el rango entre corchetes)*

Bases de Datos Clave-Valor. Redis

- **<u>Tipos de Datos Básicos de los Valores:</u>**
- **String:**
  - Puede almacenar cualquier tipo de dato: texto (XML, JSON, HTML o texto plano), integers, floats, o datos binarios (vídeos, imágenes o ficheros de audio).
  - Un valor String **no puede superar 512 MB** de texto o datos binarios.

Bases de Datos Clave-Valor. Redis

- **String:** Caché con expiración automática de una clave (**comandos EXPIRE, TTL**)

  $ redis-cli

  127.0.0.1:6379> SET capitulo_actual “Capítulo 1: Introducción"

  OK

  127.0.0.1:6379> **TTL** capitulo_actual

  (integer) **-1** ⇨Clave existe pero sin especificar el tiempo de expiración

  127.0.0.1:6379> EXPIRE capitulo_actual 10

  (integer) 1 ⇨ Especificar el tiempo de vida (en segundos) de una clave dada

  127.0.0.1:6379> GET capitulo_actual

  " Capítulo 1: Introducción "

  127.0.0.1:6379> TTL capitulo_actual

  (integer) 3 ⇨ Número de segundos de vida que le quedan

  127.0.0.1:6379> TTL capitulo_actual

  (integer) **-2** ⇨ Clave ha expirado

  127.0.0.1:6379> GET capitulo_actual

  (nil)

Bases de Datos Clave-Valor. Redis

- Comandos **INCR, INCRBY, INCRBYFLOAT, DECR, DECRBY, EXISTS y GETSET**

El valor debe ser numérico o se tiene que poder representar como tal (p.e. “22”)

```
$ redis-cli
127.0.0.1:6379> SET contador 100
OK
127.0.0.1:6379> INCR contador
(integer) 101
127.0.0.1:6379> INCRBY contador 5
(integer) 106
127.0.0.1:6379> DECR contador
(integer) 105
127.0.0.1:6379> DECRBY contador 100
(integer) 5
127.0.0.1:6379> GET contador
"5"
127.0.0.1:6379> INCRBYFLOAT contador 2.4
"7.4“
```

```
127.0.0.1:6379 > EXISTS Variable
(integer) 0
127.0.0.1:6379 > INCR Variable
(integer) 1
```

⇨ Si no existe la clave Variable, primero se crea, inicializándola a 0.

```
127.0.0.1:6379 > EXISTS Variable
(integer) 1
```

```
127.0.0.1:6379 > INCR Clave
(integer) 102
127.0.0.1:6379 > GETSET Clave 54
"102“
```

⇨ Inicializa el valor de la Clave a 54 y muestra el valor antiguo.

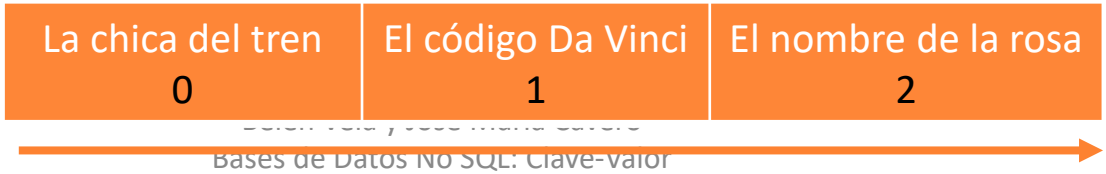
Bases de Datos Clave-Valor. Redis

- **<u>Tipos de Datos Básicos de los Valores:</u>**
- **Listas:**
  - Tipo de datos muy flexible.
  - Puede funcionar como una simple colección o como una pila o cola.
  - Los **comandos de listas son atómicos** (garantiza que sistemas concurrentes no solapen ítems en una cola)
  - La **longitud máxima** de una lista es de $2^{32} - 1$ (más de 4 000 millones de elementos).

Bases de Datos Clave-Valor. Redis

- Existen comandos para insertar datos al principio y al final de la lista:
- Comando **LPUSH:** inserta los datos al principio de la lista
- Comando **RPUSH:** inserta los datos al final de la lista

```
$ redis-cli
127.0.0.1:6379> LPUSH libros “El código Da Vinci"
(integer) 1
127.0.0.1:6379> RPUSH libros “El nombre de la rosa"
(integer) 2
127.0.0.1:6379> LPUSH libros “La chica del tren"
(integer) 3
```

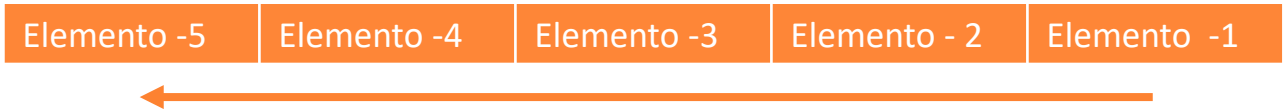


Bases de Datos Clave-Valor. Redis

- Comando **LLEN:** devuelve la longitud de una lista.
- Comando **LINDEX:** devuelve el elemento de la posición indicada por el índice.

Los elementos siempre se acceden de izquierda a derecha, siendo el primer elemento el del índice 0.

Para acceder la lista en orden inverso (empezando desde el final, se puede empezar con el índice -1 para el ultimo elemento, el -2 para el penúltimo, y así sucesivamente.




```
$ redis-cli
127.0.0.1:6379> LLEN libros
(integer) 3
127.0.0.1:6379> LINDEX libros 1
“El código Da Vinci“
```

Bases de Datos Clave-Valor. Redis

- Comando **LRANGE:** devuelve los elementos de la lista con todos los elementos comprendidos en un rango (0 primer elemento y -1 ultimo elemento):

```
$ redis-cli
127.0.0.1:6379> LRANGE libros 0 1
1) "La chica del tren"
2) "El código Da Vinci“
127.0.0.1:6379> LRANGE libros 0 -1  → Todos los elementos de la lista
1) "La chica del tren"
2) "El código Da Vinci"
3) "El nombre de la rosa"
```

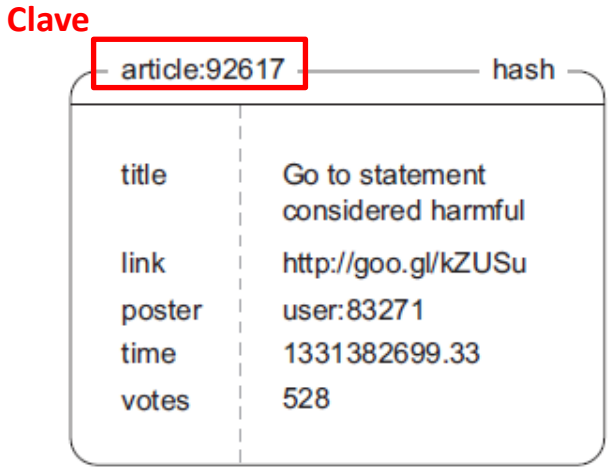
Bases de Datos Clave-Valor. Redis

- Comando **LPOP**: borra y devuelve el primer elemento de la lista
- Comando **RPOP:** borra y devuelve el último elemento de la lista

```
$ redis-cli
127.0.0.1:6379> LPOP libros → Borra el primer elemento (0)
"La chica del tren“
127.0.0.1:6379> RPOP libros → Borra el último elemento (2)
"El nombre de la rosa“
127.0.0.1:6379> LRANGE libros 0 -1 → Todos los elementos de la lista (primero 0 al ultimo -1)
1) "El código Da Vinci"
```

Bases de Datos Clave-Valor. Redis

- **<u>Tipos de Datos Básicos de los Valores:</u>**
- **Hashes:**
  - Son estructuras de datos para almacenar **objetos**. Permiten mapear campos a valores, ambos strings.
  - Están optimizados para hacer un uso de memoria eficiente y su acceso es muy rápido.



Bases de Datos Clave-Valor. Redis

- Comando **HSET:** especifica un valor para un campo de una **clave dada**.
  - Sintaxis: HSET clave campo valor.
- Comando **HMSET:** especifica múltiples campos para **una clave**, separados por espacios.
- En ambos casos, si el campo existe se sobreescribe su valor, si no se crea.
- Comando **HINCRBY:** incrementa un campo de una clave con el valor indicado.

clave

```
$ redis-cli
127.0.0.1:6379> HSET pelicula1 "titulo" “Un Monstruo viene a verme"
(integer) 1
```

clave

```
127.0.0.1:6379> HMSET pelicula1 “a_estreno" 2016 “calificacion" 6.8
“espectadores" 4000000
OK
127.0.0.1:6379> HINCRBY pelicula1 “espectadores" 100000
(integer) 4100000
```

Bases de Datos Clave-Valor. Redis

- Comando **HGET:** recupera un campo de un Hash
- Comando **HMGET:** recupera varios campos de un Hash a la vez

```
127.0.0.1:6379> HGET pelicula1 "titulo“
"Un Monstruo viene a verme«
127.0.0.1:6379> HMGET pelicula1 "titulo“ “espectadores”
1) "Un Monstruo viene a verme"
2) "4100000"
```

- Comando **HGETALL:** devuelve un array con todos los pares clave-valor de un objeto Hash:

```
127.0.0.1:6379> HGETALL pelicula1
1) "titulo"
2) "Un Monstruo viene a verme"
3) "fecha_estreno"
4) "2016"
5) "calificacion"
6) "6.8"
7) "espectadores"
8) "4100000"
```

Bases de Datos Clave-Valor. Redis

- Comando **HKEYS:** devuelve los campos de una clave Hash

```
127.0.0.1:6379> HKEYS pelicula1
1) "titulo"
2) "fecha_estreno"
3) "calificacion"
4) "espectadores“
```

- Comando **HVALS:** devuelve los **valores** de un Hash

```
127.0.0.1:6379> HVALS pelicula1
1) "Un Monstruo viene a verme"
2) "2016"
3) "6.8"
4) "4100000"
```

Bases de Datos Clave-Valor. Redis

- **Otros Tipos de Datos:**
- **Sets:** conjunto **no ordenado** de cadenas diferentes (no es posible añadir elementos repetidos)
  - SADD, SINTER, SDIFF, SUNION, SRANDMEMBER, SISMEMBER, SREM, SMEMBERS

```
> SADD BD_NoSQL “REDIS" “MongoDB" “Neo4J" “Cassandra“
(integer) 4
> SMEMBERS BD_NoSQL
1) "Neo4J"
2) "REDIS"
3) "Cassandra"
4) "MongoDB"
>SRANDMEMBER BD_NoSQL -2 ⇨ Dos elementos con repetidos (-)
1) "Neo4J"
2) "Cassandra"
> SISMEMBER BD_NoSQL "Oracle"
(integer) 0 ⇨ “Oracle” no forma parte del conjunto
```

Bases de Datos Clave-Valor. Redis

- **Sorted Sets:** conjunto **ordenado** de cadenas de caracteres ponderadas
  - ZADD, ZCOUNT, ZRANGE, ZREVRANGE, ZRANGEBYSCORE, ZREVRANGEBYSCORE, ZRANGEBYLEX, ZREVRANGEBYLEX, ZREM

Ordenados de menor a mayor
1) "natalia"
2) "maria"
3) "alicia"
4) "carla"
5) "pablo"

- ZADD **amigos_juan** 70 maria 95 pablo 95 carla 75 alicia 1 natalia
- ZCOUNT amigos_juan 90 100

  2 -> 2 miembros de amigos_juan de 90 a 100 puntos (incluidos extremos)
- ZRANGE amigos_juan 0 2

  1) "natalia"
  2) "maria"
  3) "alicia"
- ZREVRANGE amigos_juan 0 -1 -> TODOS (más a menos)

  1) "pablo"
  2) "carla"
  3) "alicia"
  4) "maria"
  5) "natalia"

Bases de Datos Clave-Valor. Redis

- **Sorted Sets:** conjunto **ordenado** de cadenas de caracteres ponderadas
  - ZADD, ZCOUNT, ZRANGE, ZREVRANGE, ZRANGEBYSCORE, ZREVRANGEBYSCORE, *ZRANGEBYLEX, ZREVRANGEBYLEX*, ZREM
    - ZRANGEBYSCORE amigos_juan -inf +inf

      1) "natalia"
      2) "maria"
      3) "alicia"
      4) "carla"
      5) "pablo"
    - ZRANGEBYSCORE amigos_juan 1 75 (inclyendo extremos)
    - ZRANGEBYSCORE amigos_juan 1 (95 (excluyendo el extremo 95)
    - ZREVRANGEBYSCORE amigos_juan +inf –inf WITHSCORES
    - ZREM amigos_juan “carla”

Ordenados de menor a mayor
1) "natalia"
2) "maria"
3) "alicia"
4) "carla"
5) "pablo"

- **Otros:**
  - **Bitmaps, HyperLogLogs:** No son realmente tipos de datos de REDIS.

Bases de Datos Clave-Valor. Redis

Ejercicio Repaso: http://try.redis.io

- Insertar en la BD REDIS 5 títulos de **películas**:
  - Utilice los comandos SET, MSET, SETNX, MSETNX
  - Compruebe el funcionamiento del comando EXPIRE y TLL
  - Use GET y MGET para recuperar los valores de las películas insertadas
  - Busque las claves insertadas con el comando KEYS (use diferentes patrones)
- Cree una lista de cantantes **L_Cantantes** e inserte 5 elementos.
  - Use los comandos LPUSH y RPUSH
  - Verifique que ha insertado de forma correcta los cantantes usando el comando LRANGE y LLEN
  - Recorra la lista en orden inverso (elemento a elemento)
  - Borre el último elemento de la cola y recupere la lista
- Cree un hash para 5 **Libros**, incluyendo su título, su ISBN, la editorial y el año.
  - Utilice los comandos HSET, HMSET
  - Recupere el contenido usando los comando HGET, HMGET, HGTALL, HKEYS y HVALS
- Cree un conjunto de productos de **BD NoSQL** y otro de **BD Relacionales**.
  - Utilice el comando SADD, incluyendo al menos 3 productos en cada uno de los conjuntos.
  - Visualice su contenido utilizando el comando SMEMBERS
  - Recupere la unión, intersección y diferencia de ambos conjuntos usando los comandos SUNION, SINTER y SDIFF.
  - Recupere 2 elementos aleatorios del conjunto de BD NoSQL.
  - Recupere 3 elementos aleatorios (con posibles repetidos) del conjunto de BD Relacionales.
- Cree un conjunto ordenado de productos de **BD NoSQL** incluyendo una puntuación de popularidad.
  - Utilice el comando ZADD, incluyendo 4 productos comerciales.
  - Visualice su contenido completo utilizando los comandos ZRANGE y ZREVRANGE
  - Borre uno de los productos del conjunto
  - Recupere los elementos del conjunto cuya puntuación esté comprendida en un rango especificado.

Índice