

Bases de Datos II

Bases de Datos NoSQL / Práctica con Redis

Grupo 1

Integrantes

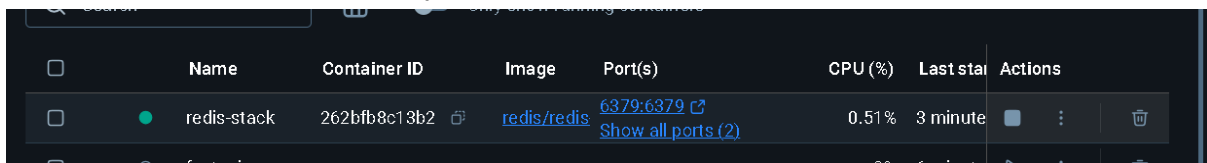
- Ladislao Bordon
- Lautaro Joaquin Gutierrez
- Facundo Feliú

Ayudante

- Natalia Ortu

Explicación de instalación: Elegimos Redis Stack, fuimos al link provisto por la cátedra y bajamos hasta abajo de todo donde decia otras descargas, ahí abrimos en redis Stack y apretamos en [release notes](#) de ahí elegimos la versión de Docker, ya que la otra versión nos mandaba directamente a la página de Redis devuelta, ahí seguimos los pasos de la documentación de [redis/redis-stack - Docker Image | Docker Hub](#):

1. primero ponemos este comando en la terminal de nuestra pc
"docker run -d --name redis-stack -p 6379:6379 -p 8001:8001 -e REDIS_ARGS="--requirepass mypassword" redis/redis-stack:latest"



<input type="checkbox"/>	Name	Container ID	Image	Port(s)	CPU (%)	Last state	Actions
<input type="checkbox"/>	redis-stack	262bfb8c13b2	redis/redis	6379:6379 Show all ports (2)	0.51%	3 minutes ago	

2. Con el siguiente comando accedemos a la consola de redis.

```
docker exec -it redis-stack redis-cli
```

Para autenticarnos usamos

"auth mypassword"

Para pararlo

docker stop redis-stack

Para iniciarlo devuelta si ya esta creado el container:

docker start redis-stack

Para borrarlo:

docker rm redis-stack

Para persistir nos dieron este comando:

```
docker run -v /local-data/:/data redis/redis-stack:latest
```

Cabe aclarar que se está ejecutando RedisInsight en <http://localhost:8001/>

Parte 1: Introducción

1. ¿Qué tipo de bd es Redis?

Redis es una base de datos NoSQL de tipo clave-valor distribuida en memoria. La durabilidad de los datos es opcional y soporta varias estructuras de datos y transacciones.

2. ¿Dónde almacena los datos Redis?

Redis almacena los datos principalmente en la memoria RAM del servidor, a diferencia de las bases de datos tradicionales que los guardan en discos. Esto permite que Redis ofrezca una alta velocidad y rendimiento, ya que el acceso a la memoria es mucho más rápido que el acceso a disco.

3. ¿Qué tipos de datos posee Redis?

Redis posee Strings, Hashes, Lists, Sets, Sorted Sets, Bitmaps, Bitfields, HyperLogLog, Geospatial indexes, y Streams como estructuras de datos nativas. Con cada estructura de datos, Redis mantiene comandos exclusivos para permitir la ejecución de operaciones de varios tipos de manera eficaz.

4. Enuncie las características de Redis

- Almacenamiento en memoria RAM, lo que lo hace extremadamente rápido.
- Persistencia opcional en disco, mediante dos configuraciones: RDB que toma snapshots cada cierto tiempo y AOF que guarda cada operación que modifica datos (más seguro, pero más lento).
- Estructuras de datos avanzada, no solo clave-valor, sino también Strings, Listas, conjuntos, etc.
- Alta disponibilidad con Redis Sentinel y alto particionamiento con Redis Cluster.
- Sistema de publicación/suscripción, útil para chats, notificaciones, etc.
- Soporte para TTL y expiración.
- Transacciones.
- Código abierto y muy liviano.
- Replicación: permite escalar las lecturas y tener redundancia.

5. Comparar Redis con los RDBMS

Características	Redis	RDBMS
Modelo de Datos	Clave-Valor	Tablas relacionales con filas y columnas
Velocidad	Muy alta (memoria RAM)	Alta, pero depende del disco
Persistencia	Opcional y configurable	Persistencia por defecto
Consultas	Limitadas, no usa SQL	Potente, usa SQL
Transacciones	Básicas	Completas (ACID)

Estructuras de Datos	Strings, List, Set, ect.	Tablas estructuradas
Integridad referencial	No	Sí (restricciones, claves primarias/foráneas)
Escalabilidad	Horizontal con Redis Cluster	Difícil horizontal, normalmente vertical
Lenguaje de consulta	Comandos simples	SQL
Almacenamiento	Memoria RAM y opcionalmente disco	Disco
Consistencia de datos	Eventual	Alta
Uso típico	Caché, colas, sesiones, contadores, pub/sub	Aplicaciones CRUD, sistemas de gestión

6. ¿Redis tiene transacciones?

Sí, Redis tiene transacciones, pero son más simples y limitadas que las de una base de datos relacional.

En Redis, una transacción agrupa varios comandos para que se ejecuten de forma secuencial y atómica, sin que otros comandos se ejecuten entre medio.

Redis usa los comandos:

- MULTI – Inicia la transacción
- EXEC – Ejecuta todos los comandos en orden
- DISCARD – Cancela la transacción
- WATCH – Permite detectar cambios antes de ejecutar (control optimista de concurrencia)

7. ¿Redis tiene persistencia?

Sí, Redis si tiene persistencia aunque en lo que vamos a hacer en esta práctica me parece que no la usamos y no la tenemos configurada, ya que cada vez que reiniciamos el contenedor se pierden los datos.

Según el Documento de Redis tenemos las siguientes opciones sobre la persistencia en Redis:

RDB (Redis Database): RDB persistencia, hace snapshots de tu base de datos en intervalos específicos

AOF (Append Only File): AOF persistencia, hace un log por cada operación de escritura recibida en el servidor y las guarda, cuando se prende el server devuelta ejecuta cada una de las operaciones recuperando la base de datos original, guarda los comandos en el mismo formato que el protocolo de Redis

RDB + AOF: Se pueden combinar ambas técnicas

8. ¿Cuáles son los principales usos de Redis?

Según el propio Redis estos son los principales casos de uso de Redis:

- Base de datos vectorial
- Tiendas de funciones

- Caché semántica
- Caché
- Base de datos NoSQL
- Tablas de clasificación
- Desduplicación de datos
- Mensajería
- Almacenamiento de tokens de autenticación
- Ingesta rápida de datos
- Caché de consultas

Ref:

<https://redis.io/es/redis-enterprise/estructuras-de-datos/#:~:text=Puede%20almacenar%20cu%20tipo%20de,cosa%20que%20desee%20que%20tenga>

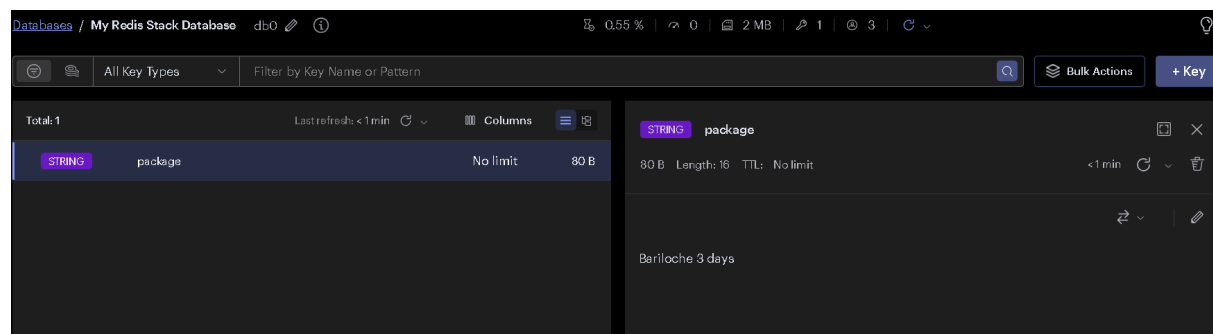
<https://aprenderbigdata.com/redis/#:~:text=Redis%20es%20una%20base%20de,Dlctionary%20Server%2C%20creado%20en%202009>

-[Redis persistence | Docs](#)

-[Una guía para principiantes sobre casos de uso populares de Redis | Redis](#)

Parte 2: Manejo Simple de Valores String

1. Agregue una clave package con el valor "Bariloche 3 days"
127.0.0.1:6379> SET package "Bariloche 3 days"
(error) NOAUTH Authentication required.
127.0.0.1:6379> AUTH mypassword
OK
127.0.0.1:6379> SET package "Bariloche 3 days"
OK



2. Agregue una clave user con el valor "Turismo BD2". Obtenga el valor de la clave user
127.0.0.1:6379> SET user "Turismo BD2"
OK
127.0.0.1:6379> get user
"Turismo BD2"
3. Obtenga todos los valores de claves almacenadas

```
127.0.0.1:6379> KEYS *
```

```
1) "user"
```

```
2) "package"
```

4. Agregue una clave user con el valor "Cronos Turismo" ¿Cuál es el valor actual de la clave user?

```
127.0.0.1:6379> SET user "Cronos Turismo"
```

```
OK
```

```
127.0.0.1:6379> get user
```

```
"Cronos Turismo"
```

5. Concatene " S.A." a la clave user. ¿Cuál es el valor actual de la clave user?

```
127.0.0.1:6379> APPEND user " S.A."
```

```
(integer) 19
```

```
127.0.0.1:6379> get user
```

```
"Cronos Turismo S.A."
```

6. Elimine la clave user.

```
127.0.0.1:6379> DEL user
```

```
(integer) 1
```

7. ¿Qué valor retorna si queremos obtener la clave user?

```
127.0.0.1:6379> get user
```

```
(nil)
```

Parte 2: Manejo Simple de Valores Numéricos (tendría que ser parte 3)

1. Verificar si existe la clave visits

```
> EXISTS visits  
(integer) 0
```

Al devolver 0 indica que no existe.

2. Agregue una clave visits con el valor 0

```
> SET visits 0  
"OK"
```

3. Incremente en 1 visits ¿Cuál es el valor actual de la clave visits?

```
> INCR visits  
(integer) 1
```

Incremento en 1 visits y me devuelve el valor actual que es 1.

4. Incremente en 5 visits. ¿Cuál es el valor actual de la clave visits?

```
> INCRBY visits 5
(integer) 6
```

Incremento en 5 visits y me devuelve el valor actual que es 6.

5. Decremento en 1 visits ¿Cuál es el valor actual de la clave visits?

```
> DECR visits  
(integer) 5
```

Decremento en 1 visits y me devuelve el valor actual que es 5..

6. Incremente en 2 visits. ¿Cuál es el valor actual de la clave visits?

```
> INCRBY visits 2
(integer) 7
```

Incremento en 2 visits y me devuelve el valor actual que es 7.

7. Agregue una clave “value package” con el valor 539789.32

```
> SET "value package" 539789.32
"OK"
```

8. Incremente en 20000 la clave “value package”. ¿Cuál es el valor actual de “value package”?

```
> INCRBYFLOAT "value package" 20000
"559789.3199999999999999318"
```

9. ¿Cuál es el tipo de datos de “value package”, visits y user?

```
> TYPE "value package"
"string"
```

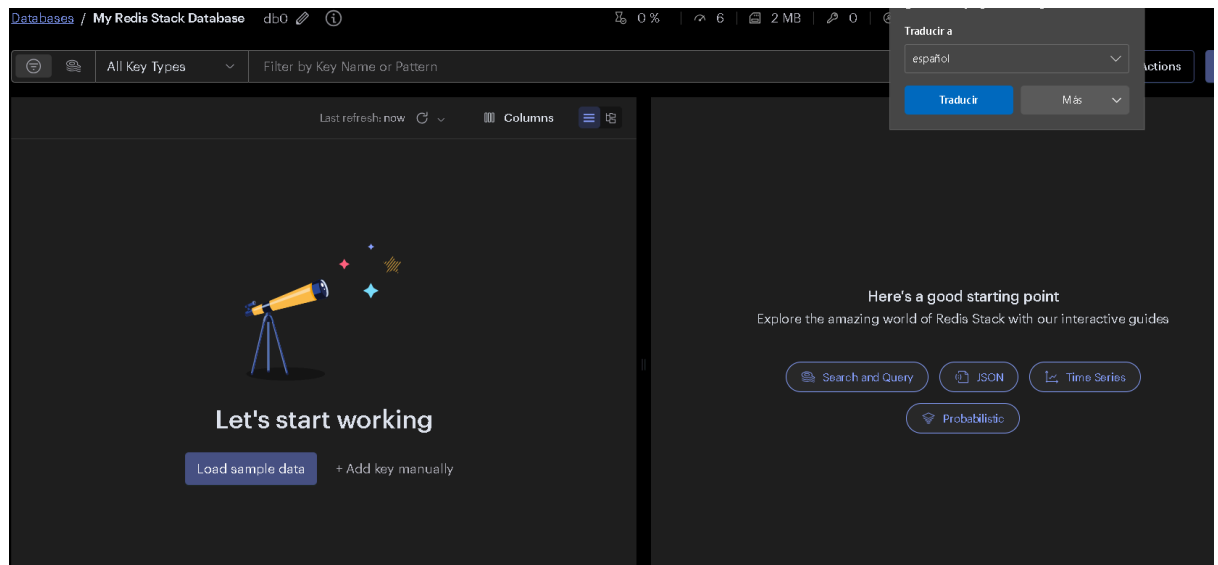
```
> TYPE visits
"string"
```

```
> TYPE user
"string"
```

Parte 4: Manejo de Claves

1. Obtenga todas las claves que empiecen con v
127.0.0.1:6379> KEYS v*
1) "visits"
2) "value package"
2. Obtenga todas las claves que contengan la "t"
127.0.0.1:6379> KEYS *t*
1) "visits"
3. Obtenga todas las claves que terminen con "age"
127.0.0.1:6379> KEYS *age
1) "value package"
2) "package"
4. Renombre la clave "package" por "bariloche package"
127.0.0.1:6379> RENAME package 'bariloche package'
OK
127.0.0.1:6379> get package
(nil)
127.0.0.1:6379> get 'bariloche package'
"Bariloche 3 days"
5. Si quisiera evitar renombrar una clave por otra existente, ¿qué comando utilizaría?
Para evitar esto usaria REMANEX el comando verifica si existe y si es asi devuelve 0 y no hace nada, pero si no existe devuelve 1 y si renombra.
127.0.0.1:6379> keys *
1) "visits"
2) "bariloche package"
3) "value package"
127.0.0.1:6379> renamenx 'value package' 'visits'
(integer) 0
127.0.0.1:6379> keys *
1) "visits"
2) "bariloche package"
3) "value package"
127.0.0.1:6379> renamenx 'value package' 'visits new'
(integer) 1
127.0.0.1:6379> keys *
1) "visits"
2) "bariloche package"
3) "visits new"
6. Elimine todas las claves
127.0.0.1:6379> flushdb
OK
127.0.0.1:6379> keys *

(empty array)



Parte 5: Expiración de Claves

1. Agregue una clave agency con el valor “Cronos Tours”

```
127.0.0.1:6379> set agency "Cronos Tours"
```

OK

```
127.0.0.1:6379> get agency
```

"Cronos Tours"

2. ¿Cuál es el tiempo de vida de la clave agency?

infinito:

```
127.0.0.1:6379> ttl agency
```

(integer) -1

3. Agregue una expiración de 30 segundos a la clave agency

```
127.0.0.1:6379> EXPIRE agency 30
```

(integer) 1

4. ¿Cuál es el tiempo de vida de la clave agency?

Depende de la velocidad que pongas el comando, pero automáticamente tiene 30 segundos de tiempo de vida, pero cuando puse el comando le quedaban 28s

```
127.0.0.1:6379> ttl agency
```

(integer) 28

5. Pasados los 30 segundos, ¿Cuál es el tiempo de vida de la clave agency? ¿Qué retorna si pido el valor de agency?

```
127.0.0.1:6379> ttl agency
```

(integer) -2

```
127.0.0.1:6379> get agency
```

(nil)

6. Agreguemos una clave agency con el valor “Cronos Tours” que expire en 20 segundos


```
127.0.0.1:6379> SET agency "Cronos Tours" EX 20
```

```
OK
```

```
127.0.0.1:6379> ttl agency
```

```
(integer) 15
```

Parte 6: Listas

1. Inserte una lista llamada pets con el valor dog.

```
> LPUSH pets dog  
(integer) 1
```

2. ¿Qué sucede si ejecuto el comando get pets? ¿Cómo obtengo los valores de la lista?

```
> GET pets  
"WRONGTYPE Operation against a key holding the wrong kind of value"
```

Da error porque GET solo funciona con claves tipo string.

```
> LRANGE pets 0 -1  
1) "dog"
```

El parámetro 0 indica la posición inicial (primer elemento de la lista) y el parámetro -1 la posición final (último elemento de la lista).

3. Agregue a la lista pets el valor cat a la izquierda.

```
> LPUSH pets cat  
(integer) 2
```

LPUSH agrega al inicio de la lista.

```
> LRANGE pets 0 -1  
1) "cat"  
2) "dog"
```

4. Agregue a la lista pets el valor fish a la derecha.

```
> RPUSH pets fish  
(integer) 3
```

```
> LRANGE pets 0 -1  
1) "cat"  
2) "dog"  
3) "fish"
```

5. ¿Qué tipo de datos es el valor de pets?

```
> TYPE pets  
"list"
```

6. Elimine el valor a la izquierda de la lista.

```
> LPOP pets  
"cat"
```

```
> LRANGE pets 0 -1  
1) "dog"  
2) "fish"
```

7. Elimine el valor a la derecha de la lista.

Aclaración: Asumimos que quisieron referirse a la derecha de la lista.

```
> RPOP pets  
"fish"
```

```
> LRANGE pets 0 -1  
1) "dog"
```

8. Agregue una clave “vuelo:ar389” los valores: aep, mdz, brc, nqn y mdq.

```
> RPUSH vuelo:ar389 aep mdz brc nqn mdq  
(integer) 5
```

```
> LRANGE vuelo:ar389 0 -1  
1) "aep"  
2) "mdz"  
3) "brc"  
4) "nqn"  
5) "mdq"
```

9. Ordene los valores de lista “vuelo:ar389”. ¿Qué sucede si solicito todos los valores de la lista?

```
> SORT vuelo:ar389 ALPHA
1) "aep"
2) "brc"
3) "mdq"
4) "mdz"
5) "nqn"
```

Devuelve la lista ordenada alfabéticamente, pero no la altera en la lista original, solo la devuelve así. Por lo que si solicitamos todos los valores se va a mostrar la lista con su orden original.

```
> LRANGE vuelo:ar389 0 -1
1) "aep"
2) "mdz"
3) "brc"
4) "nqn"
5) "mdq"
```

10. Inserte el valor "fte" luego de "brc"

```
> LINSERT vuelo:ar389 AFTER brc fte
(integer) 6
```

```
> LRANGE vuelo:ar389 0 -1
1) "aep"
2) "mdz"
3) "brc"
4) "fte"
5) "nqn"
6) "mdq"
```

11. Inserte el valor "ush" antes de "fte"

```
> LINSERT vuelo:ar389 BEFORE fte ush
(integer) 7
```

```
> LRANGE vuelo:ar389 0 -1
1) "aep"
2) "mdz"
3) "brc"
4) "ush"
5) "fte"
6) "nqn"
7) "mdq"
```

12. Modifique el último elemento por “sla”

Al haber 7 elementos la última posición es la 6.

```
> LSET vuelo:ar389 6 sla
"OK"
```

```
> LRANGE vuelo:ar389 0 -1
1) "aep"
2) "mdz"
3) "brc"
4) "ush"
5) "fte"
6) "nqn"
7) "sla"
```

13. Obtenga la cantidad de elementos de “vuelo:ar389”

```
> LLEN vuelo:ar389
(integer) 7
```

14. Obtenga el 3 valor de “vuelo:ar389”

```
> LINDEX vuelo:ar389 2
"brc"
```

15. Elimine el valor “aep” de “vuelo:ar389”

```
> LREM vuelo:ar389 1 aep
(integer) 1
```

El parámetro 1 significa que va a eliminar una sola ocurrencia del valor de izquierda a derecha.

16. Quédese con los valores de las posiciones 3 a 5 de “vuelo:ar389”

```
> LRANGE vuelo:ar389 3 5
1) "fte"
2) "nqn"
3) "sla"
```

17. Agregue en "vuelo:ar389" el valor "fte". ¿Cuántas veces aparece?

```
> RPUSH vuelo:ar389 fte
(integer) 7
```

```
> LRANGE vuelo:ar389 0 -1
1) "mdz"
2) "brc"
3) "ush"
4) "fte"
5) "nqn"
6) "sla"
7) "fte"
```

Aparece 2 veces.

Parte 7: Conjuntos

1. Agregue un conjunto llamado airports los valores: eze aep nqn mdz mdq ush fte sla aep nqn brc cpc juj aep tuc eqs

```
127.0.0.1:6379> SADD airports eze aep nqn mdz mdq ush fte sla aep nqn brc cpc juj aep tuc eqs
(integer) 13
```

2. ¿Cuántos valores tiene el conjunto?

Tiene 13 valores, ya que los conjuntos no aceptan valores duplicados.

3. Liste los valores del conjunto airports.

```
127.0.0.1:6379> SMEMBERS airports
1) "eze"
2) "aep"
3) "nqn"
4) "mdz"
5) "mdq"
6) "ush"
7) "fte"
8) "sla"
9) "brc"
10) "cpc"
11) "juj"
12) "tuc"
13) "eqs"
```

4. Quite el valor cpc del conjunto airports.

```
127.0.0.1:6379> SREM airports cpc
(integer) 1
```

5. Quite un valor aleatorio del conjunto airports.

```
127.0.0.1:6379> SPOP airports
"brc"
```

6. ¿Qué cantidad de valores tiene ahora airports?

```
127.0.0.1:6379> SCARD airports
(integer) 11
```

Tiene 11 elementos

7. Compruebe si cpc es miembro del conjunto airports.

```
127.0.0.1:6379> SISMEMBER airports cpc  
(integer) 0
```

No es miembro

8. Mueva los valores sla y juj a un conjunto denominado noa_airports.

```
127.0.0.1:6379> SMOVE airports noa_airports sla  
(integer) 1  
127.0.0.1:6379> SMOVE airports noa_airports juj  
(integer) 1
```

9. Retorne la unión de los conjuntos airports y noa_airports. ¿Modifica los conjuntos base?

```
127.0.0.1:6379> SUNION airports noa_airports  
1) "eze"  
2) "aep"  
3) "nqn"  
4) "mdz"  
5) "mdq"  
6) "ush"  
7) "fte"  
8) "tuc"  
9) "eqs"  
10) "sla"  
11) "juj"
```

No, SUNION no modifica los conjuntos base. Solo devuelve la unión de sus elementos.

10. Realice la unión de los conjuntos airports y noa_airports en un conjunto llamado total_airports.

```
127.0.0.1:6379> SUNIONSTORE total_airports airports noa_airports  
(integer) 11
```

11. Realice la intersección entre los conjuntos total_airports y noa_airports.

```
127.0.0.1:6379> SINTER total_airports noa_airports  
1) "sla"  
2) "juj"
```

12. Realice la diferencia entre los conjuntos total_airports y noa_airports.

```
127.0.0.1:6379> SDIFF total_airports noa_airports  
1) "eze"  
2) "aep"  
3) "nqn"  
4) "mdz"  
5) "mdq"  
6) "ush"  
7) "fte"  
8) "tuc"  
9) "eqs"
```

Parte 8: Conjuntos Ordenados

1. En un conjunto ordenado llamado passengers agregue los siguientes datos:

2.5 federico 4 alejandra 3 julian 1 ivan 2 andrea 2 luciana 2.4 natalia

```
127.0.0.1:6379> ZADD passengers 2.5 federico 4 alejandra 3 julian 1 ivan 2 andrea 2 luciana 2.4 natalia  
(integer) 7
```

2. Obtenga los valores del conjunto passengers

```
127.0.0.1:6379> ZRANGE passengers 0 -1
1) "ivan"
2) "andrea"
3) "luciana"
4) "natalia"
5) "federico"
6) "julian"
7) "alejandra"
```

3. Actualice el score de luciana a 2.7

```
127.0.0.1:6379> ZADD passengers 2.7 luciana
(integer) 0
```

4. Agregue al conjunto passengers a silvia con score 5.1

```
127.0.0.1:6379> ZADD passengers 5.1 silvia
(integer) 1
```

5. Incremente en 2 el score de alejandra en el conjunto passengers.

```
127.0.0.1:6379> ZINCRBY passengers 2 alejandra
"6"
```

6. Obtenga los valores del conjunto passengers con sus scores.

```
127.0.0.1:6379> ZRANGE passengers 0 -1 WITHSCORES
1) "ivan"
2) "1"
3) "andrea"
4) "2"
5) "luciana"
6) "2"
7) "natalia"
8) "2.4"
9) "federico"
10) "2.5"
11) "julian"
12) "3"
13) "alejandra"
14) "4"
```

7. Obtenga los valores del conjunto passengers con sus scores en orden inverso.

```
127.0.0.1:6379> ZREVRANGE passengers 0 -1 WITHSCORES
1) "alejandra"
2) "6"
3) "silvia"
4) "5.1"
5) "julian"
6) "3"
7) "luciana"
8) "2.7"
9) "federico"
10) "2.5"
11) "natalia"
12) "2.4"
13) "andrea"
14) "2"
15) "ivan"
16) "1"
```

8. Obtenga la cantidad de elementos del conjunto passengers.

```
127.0.0.1:6379> ZCARD passengers
(integer) 8
```

9. Obtenga la cantidad de elementos que tienen scores entre 2 y 3.

```
127.0.0.1:6379> ZCOUNT passengers 2 3
(integer) 5
```

10. Obtenga el ranking de julian en el conjunto passengers.

```
127.0.0.1:6379> ZRANK passengers julian  
(integer) 5
```

11. Obtenga el score de andrea en el conjunto passengers.

```
127.0.0.1:6379> ZRANK passengers andrea  
(integer) 1
```

12. Extraiga el valor de menor score del conjunto passengers.

```
127.0.0.1:6379> ZRANGE passengers 0 0  
1) "ivan"
```

13. Extraiga el valor de mayor score del conjunto passengers.

```
127.0.0.1:6379> ZREVRANGE passengers 0 0  
1) "alejandra"
```

14. Elimine del conjunto passengers al valor silvia.

```
127.0.0.1:6379> ZREM passengers silvia  
(integer) 1
```

Parte 9: Hashes

1. Agregue a un hash llamado user:cronos los valores:

“razon social” “cronos s.a”, domicilio “47 236 La Plata”, “teléfono” 2215556677

```
> HSET user:cronos "razon social" "cronos s.a" domicilio "47 236 La Plata" teléfono 2215556677  
(integer) 3
```

2. Agregue el mail info@cronos.com.ar a user:cronos

```
> HSET user:cronos mail info@cronos.com.ar  
(integer) 1
```

3. Obtenga todos los valores de user:cronos

```
> HVALS user:cronos  
1) "cronos s.a"  
2) "47 236 La Plata"  
3) "2215556677"  
4) "info@cronos.com.ar"
```

4. Obtenga el mail de user:cronos

```
> HGET user:cronos mail  
"info@cronos.com.ar"
```

5. Elimine el teléfono de user:cronos


```
> HDEL user:cronos teléfono
(integer) 1
```

6. Obtenga la cantidad de campos de user:cronos

```
> HLEN user:cronos
(integer) 3
```

7. Obtenga las claves de los campos de user:cronos

```
> HKEYS user:cronos
1) "razon social"
2) "domicilio"
3) "mail"
```

8. Determine si existe el campo cuil en user:cronos

```
> HEXISTS user:cronos cuil
(integer) 0
```

El 0 determina que no existe, si hubiese devuelto 1 significa que existe.

9. Obtenga todos los valores de los campos de user:cronos

```
> HVALS user:cronos
1) "cronos s.a"
2) "47 236 La Plata"
3) "info@cronos.com.ar"
```

10. Obtenga la longitud del campo mail de user:cronos

```
> HSTRLEN user:cronos mail
(integer) 18
```

Parte 10: Geospatial

1. Agregue en un conjunto denominado cities las siguientes localidades:

Buenos Aires	-34.61315, -58.37723
Córdoba	-31.4135, -64.18105
Rosario	-32.94682, -60.63932
Mendoza	-32.89084, -68.82717
San Miguel de Tucumán	-26.82414, -65.2226

La Plata	-34.92145, -57.95453
Mar del Plata	-38.00042, -57.5562
Salta	-24.7859, -65.41166
Santa Fe	-31.64881, -60.70868
San Juan	-31.5375, -68.53639
Resistencia	-27.46056, -58.98389
Santiago del Estero	-27.79511, -64.26149
Posadas	-27.36708, -55.89608
San Salvador de Jujuy	-24.19457, -65.29712
Bahía Blanca	-38.71959, -62.27243
Paraná	-31.73271, -60.52897

127.0.0.1:6379> GEOADD cities -58.37723 -34.61315 "Buenos Aires"
(integer) 1

127.0.0.1:6379> GEOADD cities -64.18105 -31.4135 "Córdoba"
(integer) 1

127.0.0.1:6379> GEOADD cities -60.63932 -32.94682 "Rosario"
(integer) 1

127.0.0.1:6379> GEOADD cities -68.82717 -32.89084 "Mendoza"
(integer) 1

127.0.0.1:6379> GEOADD cities -65.2226 -26.82414 "San Miguel de Tucumán"
(integer) 1

127.0.0.1:6379> GEOADD cities -57.95453 -34.92145 "La Plata"
(integer) 1

127.0.0.1:6379> GEOADD cities -57.5562 -38.00042 "Mar del Plata"
(integer) 1

127.0.0.1:6379> GEOADD cities -65.41166 -24.7859 "Salta"
(integer) 1

127.0.0.1:6379> GEOADD cities -60.70868 -31.64881 "Santa Fe"
(integer) 1

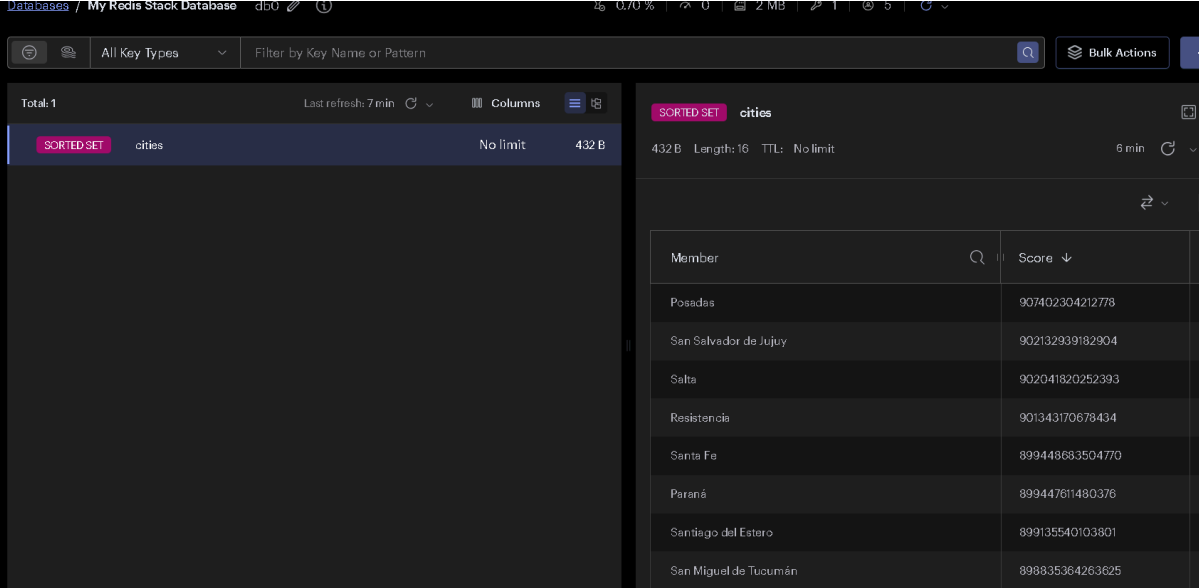
127.0.0.1:6379> GEOADD cities -68.53639 -31.5375 "San Juan"
(integer) 1

127.0.0.1:6379> GEOADD cities -58.98389 -27.46056 "Resistencia"
(integer) 1

127.0.0.1:6379> GEOADD cities -64.26149 -27.79511 "Santiago del Estero"
(integer) 1

127.0.0.1:6379> GEOADD cities -55.89608 -27.36708 "Posadas"
(integer) 1

```
127.0.0.1:6379> GEOADD cities -65.29712 -24.19457 "San Salvador de Jujuy"
(integer) 1
127.0.0.1:6379> GEOADD cities -62.27243 -38.71959 "Bahía Blanca"
(integer) 1
127.0.0.1:6379> GEOADD cities -60.52897 -31.73271 "Paraná"
(integer) 1
```



The screenshot shows the Redis Stack Database interface. The main panel displays the 'cities' sorted set with 16 members. The table lists the member names and their corresponding scores, sorted in descending order.

Member	Score
Posadas	907402304212778
San Salvador de Jujuy	902132939182904
Salta	902041820252393
Resistencia	901343170678434
Santa Fe	899448683504770
Paraná	899447611480376
Santiago del Estero	899135540103801
San Miguel de Tucumán	898835364263625

2. Obtenga los valores del conjunto cities

```
127.0.0.1:6379> ZRANGE cities 0 -1
```

- 1) "Mendoza"
- 2) "San Juan"
- 3) "Bah\xca3\xda Blanca"
- 4) "Mar del Plata"
- 5) "Buenos Aires"
- 6) "La Plata"
- 7) "Rosario"
- 8) "C\xca3\xbdoba"
- 9) "San Miguel de Tucum\xca1n"
- 10) "Santiago del Estero"
- 11) "Paran\xca3\xal"
- 12) "Santa Fe"
- 13) "Resistencia"
- 14) "Salta"
- 15) "San Salvador de Jujuy"
- 16) "Posadas"

Nota: esos valores raros son por las tildes

3. Obtenga las coordenadas de Santa Fe

```
127.0.0.1:6379> GEOPos cities "Santa Fe"
```

- 1) 1) "-60.70868164300918579"

2) "-31.64881101691540977"

4. Obtenga la distancia en km entre Buenos Aires y Córdoba

GEODIST cities "Buenos Aires" "Córdoba" km

"647.6303"

5. Obtenga las ciudades que están en un radio de 100 km de la coordenada -27.37 -55.9 con su distancia.

127.0.0.1:6379> GEORADIUS cities -55.9 -27.37 100 km WITHDIST

1) 1) "Posadas"

2) "0.5055"

6. Obtenga las ciudades que están a menos de 700 km de Córdoba.

127.0.0.1:6379> GEORADIUS cities Córdoba

1) 1) "-64.18104797601699829"

2) "-31.41350017808817796"

127.0.0.1:6379> GEORADIUS cities -64.18105 -31.4135 700 km WITHDIST

1) 1) "C\x3\xb3rdoba"

2) "0.0002"

2) 1) "San Miguel de Tucum\x3\xa1n"

2) "520.3835"

3) 1) "Santiago del Estero"

2) "402.5353"

4) 1) "Paran\x3\xa1"

2) "347.8773"

5) 1) "Santa Fe"

2) "330.2204"

6) 1) "Resistencia"

2) "668.2174"

7) 1) "Buenos Aires"

2) "647.6305"

8) 1) "La Plata"

2) "698.5503"

9) 1) "Rosario"

2) "374.4700"

10) 1) "San Juan"

2) "413.3543"

11) 1) "Mendoza"

2) "467.3003"