Rapport TP

Prise en main de Redis

Addou Aicha Amira

Bombay Marc

3ème Année Ingénieur - INFO - Apprentissage

Création/suppression d'une paire clé-valeur

```
C:\Users\ing>docker exec -it redis redis-cli
127.0.0.1:6379> SET demo "Bonjour"
OK
127.0.0.1:6379> GET demo
"Bonjour"
127.0.0.1:6379> DEL demo
(integer) 1
127.0.0.1:6379> DEL demo
(integer) 0
```

Avec:

- `SET` pour la créer
- `GET` pour lire la valeur
- `DEL` pour sa suppression

Création d'un compteur

```
127.0.0.1:6379> set cpt 0
OK
127.0.0.1:6379> incr cpt
(integer) 1
127.0.0.1:6379> incr cpt
(integer) 2
127.0.0.1:6379> incr cpt
(integer) 3
127.0.0.1:6379> decr cpt
(integer) 2
```

- Si la valeur est un nombre, on peut l'incrémenter de 1 ou la décrémenter de 1 avec `INCR` et `DECR`, respectivement

TTL

```
127.0.0.1:6379> ttl cle
(integer) -1
127.0.0.1:6379> expire cle 80
(integer) 1
127.0.0.1:6379> ttl cle
(integer) 75
```

- `TTL` = Time To Live, ici en secondes, donne combien de temps la paire clé-valeur est sauvegardée.
- On change cette valeur via expire.

Pile/file

```
127.0.0.1:6379> RPUSH cours a
(integer) 1
127.0.0.1:6379> RPUSH cours b
(integer) 2
127.0.0.1:6379> RPUSH cours c
(integer) 3
127.0.0.1:6379> RPUSH cours d
(integer) 4
```

- RPUSH ajoute à la droite de la liste (= ajoute en fin de liste)

```
127.0.0.1:6379> RPOP cours
"d"
127.0.0.1:6379> LPOP cours
"a"
127.0.0.1:6379> lrange mesCours 0 -1
(empty array)
127.0.0.1:6379> lrange cours 0 -1
1) "b"
2) "c"
```

- RPOP enlève la valeur en fin de liste et nous la renvoie.
- LPOP fait de même pour la 1re valeur.

-
```
127.0.0.1:6379> lrange cours 0 -1
1) "b"
2) "c"
```
LRANGE permet de récupérer une liste du début à la fin

Si on souhaite récupérer l'ordre inverse ça ne fonctionne pas avec LRANGE mais avec LREVRANGE

Ensembles

```
127.0.0.1:6379> SADD setusers aa
(integer) 1
127.0.0.1:6379> SADD setusers bb
(integer) 1
127.0.0.1:6379> SADD setusers cc
(integer) 1
127.0.0.1:6379> SADD setusers aa
(integer) 0
127.0.0.1:6379> SADD setusers dd
(integer) 1
```

- Créer avec SADD
- On ne peut pas avoir de valeurs dupliquées à l'intérieur : le deuxième SADD setusers aa n'a pas fonctionné (renvoie 0)

```
127.0.0.1:6379> SMEMBERS setusers
1) "aa"
2) "bb"
3) "cc"
4) "dd"
127.0.0.1:6379> SREM aa
(error) ERR wrong number of arguments for 'srem' command
127.0.0.1:6379> SREM setusers aa
(integer) 1
127.0.0.1:6379> SREM setusers
(error) ERR wrong number of arguments for 'srem' command
127.0.0.1:6379> SUNION setusers cours
(error) WRONGTYPE Operation against a key holding the wrong k
```

- Pour récupérer les valeurs de l'ensemble, SMEMBERS
- SREM permet d'enlever une des valeurs
- On peut faire des opérations d'ensemble (union, intersection, …) ; ici l'union est faite par exemple avec SUNION

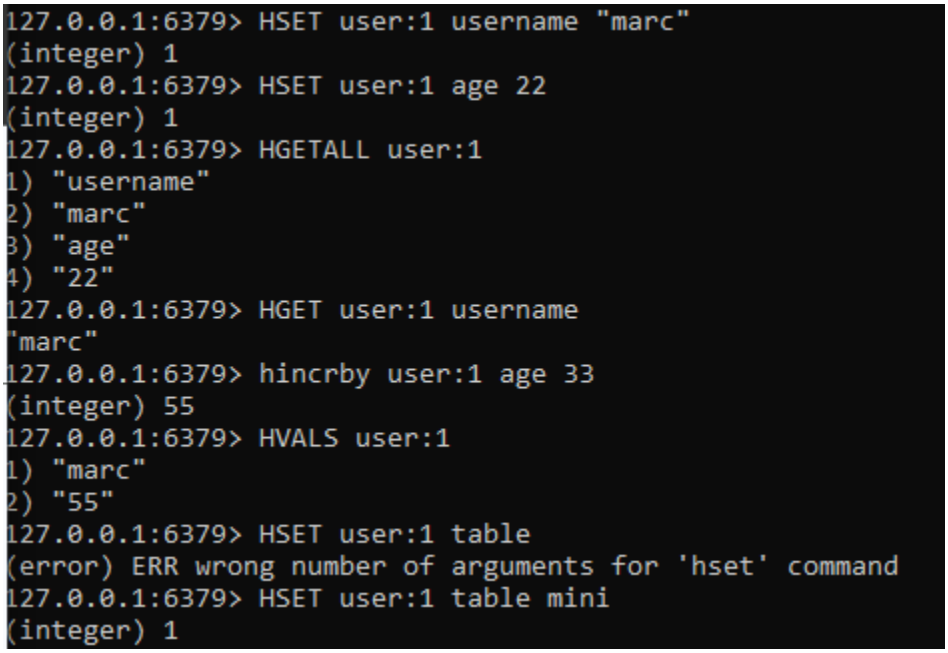
Ensembles ordonnés

```
127.0.0.1:6379> ZADD score 9 "marc"
(integer) 1
127.0.0.1:6379> ZADD score 2 "ahmed"
(integer) 1
127.0.0.1:6379> ZADD score 3 "gomez"
(integer) 1
127.0.0.1:6379> ZRANGE score 0 -1
1) "ahmed"
2) "gomez"
3) "marc"
127.0.0.1:6379> ZRANGE score -1 0
(empty array)
127.0.0.1:6379> ZREVRANGE score 0 -1
1) "marc"
2) "gomez"
3) "ahmed"
127.0.0.1:6379> ZRANK score "Marc"
(nil)
127.0.0.1:6379> ZRANK score "Marc"
(nil)
127.0.0.1:6379> ZRANK score "marc"
(integer) 2
```

- Comme un ensemble, mais les valeurs ont un rang.
- Commandes quasi identiques, il faut juste remplacer S par Z.
- On remarque que ZRANGE nous donne les valeurs dans l'ordre croissant

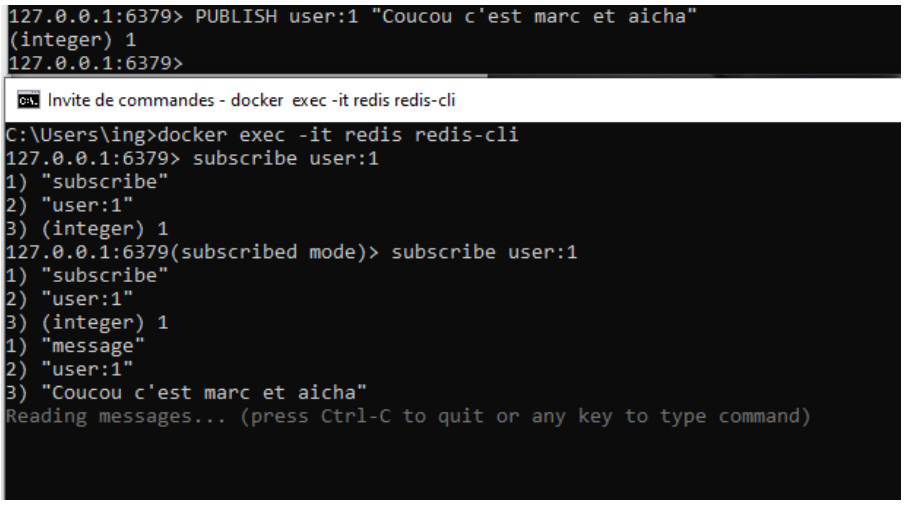
- `ZRANK` permet de retrouver le rang d'une valeur donnée.

Création de Hash (Dictionnaire)

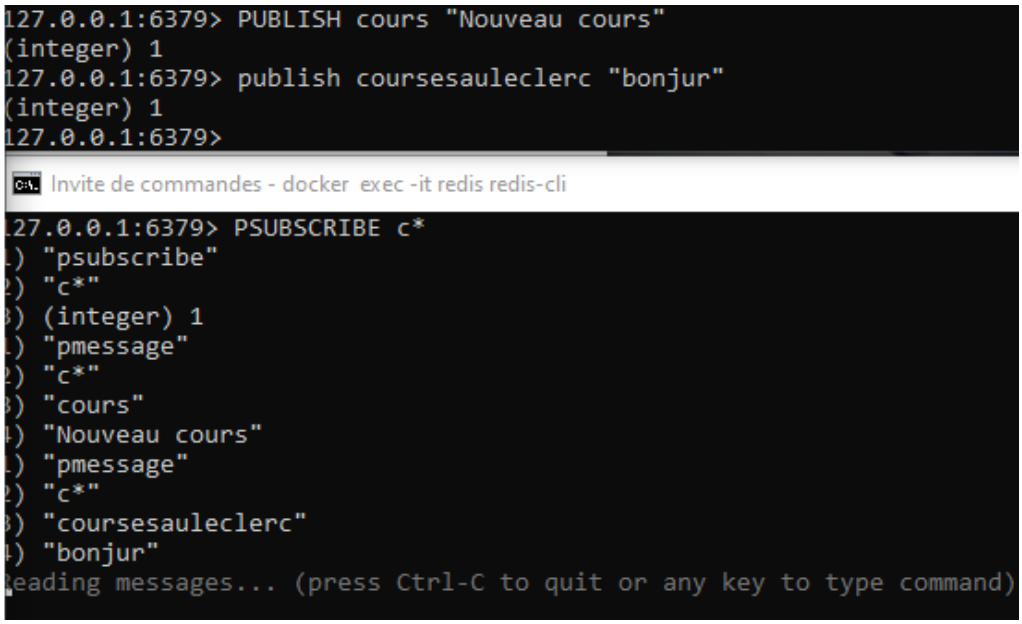


- `HSET` : Permet d'ajouter une clé/valeur à un Hash, il faut naturellement spécifier la valeur de la clé pour que cette dernière soit ajouté au Hash
- `HGETALL` : permet de récuperer toutes les valeurs du Hash (noms des clés suivis par leurs valeurs)
- `HINCRBY` : pour incrémenter un argument on lui additionnant une valeur qu'on spécifie (ici on additionne 33 à l'âge 22)

Souscription



- On ouvre un nouveau terminal et on se souscrit “SUBSCRIBE” à user:1 un hash qu’on a créé précédemment
- Dans le terminal initial on PUBLISH une nouvelle valeur dans le Hash, cette valeur va apparaître instantanément sur le terminal de souscription



- On fait de même mais cette fois on utilise * pour suscribe à tous les cours qui commence par la lettre “c”
- Pour la souscription dans ce cas on utilise PSUBSCRIBE
- Dès lors qu’on publie une valeur sur une liste commençant par “c”, celle ci va s’afficher au niveau de la souscription

```
127.0.0.1:6379> SELECT 1
OK
127.0.0.1:6379[1]> HGET user:1
(error) ERR wrong number of arguments for 'hget' command
127.0.0.1:6379[1]> HGET user:1 username
(nil)
127.0.0.1:6379[1]> KEYS *
(empty array)
127.0.0.1:6379[1]> SELECT 0
OK
127.0.0.1:6379> KEYS *
1) "user:1"
2) "cours"
3) "setusers"
4) "cpt"
5) "score"
127.0.0.1:6379> SELECT 1
OK
127.0.0.1:6379[1]>
```

- Sur Redis il existe 16 BDD par défaut, numérotées de 0 à 15
- On peut se connecter à une des BDD en utilisant `SELECT` suivi du numéro de la BDD
- Comme on a créé user:1 dans la BDD numéro 0 alors lorsqu'on souhaite afficher user:1, cette clé n'existe pas
- pour afficher toutes les clés de la BDD on utilise `KEYS *`

Commandes Bonus

```
127.0.0.1:6379> json.set jsonne . '{}'
OK
127.0.0.1:6379> json.get jsonne .
"{}"
127.0.0.1:6379> json.set jsonne ./test 3
OK
127.0.0.1:6379> json.get jsonne .
"{\"/test\":3}"
127.0.0.1:6379> json.set jsonne .liste '[1,2]'
OK
127.0.0.1:6379> json.get jsonne .
"{\"/test\":3,\"liste\":[1,2]}"
127.0.0.1:6379> json.get jsonne .liste
"[1,2]"
127.0.0.1:6379> json.del jsonne ./test
(integer) 1
127.0.0.1:6379> json.get jsonne .
"{\"liste\":[1,2]}"
127.0.0.1:6379> json.clear jsonne .liste
(integer) 1
127.0.0.1:6379> json.get jsonne .
"{\"liste\":[]}"
```

Redis permet aussi de travailler avec le format JSON.
Cela se fait avec les commandes préfixées par “`JSON.`”

Ici, 4 commandes sont données :

- `JSON.SET` affecte une valeur dans le JSON spécifié.
  - `json.set jsonne . '{}'` va créer un JSON vide nommé `jsonne`.
  - `json.set jsonne ./test 3` ajoute la clé `/test` de valeur 3 à la racine du JSON `jsonne`.
- `JSON.GET` retrouve une valeur du JSON.
  - On spécifie un chemin après le nom du JSON, commençant par ‘.’ pour symboliser la racine
- `JSON.DEL` supprime une clé du JSON.
- `JSON.CLEAR` vide un objet ou un tableau spécifié.