

Fait par Kulieshova Daria INFOA3

Partie 1 : La différence entre les BDD relationnelles et les bases de données NoSQL

Les bases de données relationnelles (SQL) utilisent des tables organisées en lignes et colonnes, avec des relations définies entre les données grâce à des clés. Elles conviennent aux données structurées et aux applications nécessitant une cohérence stricte (par ex., banques).

Les bases de données NoSQL sont non relationnelles, plus flexibles et conçues pour gérer de grandes quantités de données non structurées ou semi-structurées. Elles offrent différents modèles (documents, colonnes, graphes, etc.) et sont idéales pour les applications nécessitant une grande scalabilité et rapidité (par ex., réseaux sociaux).

Partie 2 : Utilisation de Redis sur Docker

Les étapes d'installation :

1. Installer le Docker (<https://docs.docker.com/desktop/>)
2. Installer Redis sur Docker en lançant la commande `docker pull redis`

Résultat attendu :



```
Terminal
PS C:\Users\daria> docker pull redis
Using default tag: latest
latest: Pulling from library/redis
af302e5c37e9: Pull complete
01b95e092fd0: Pull complete
c111ca53a743: Pull complete
f7d6cf14046e: Pull complete
589f36d317d9: Pull complete
94041d0cae8f: Pull complete
4f4fb700ef54: Pull complete
4f5f785c9703: Pull complete
Digest: sha256:ca65ea36ae16e709b0f1c7534bc7e5b5ac2e5bb3c97236e4fec00e3625eb678d
Status: Downloaded newer image for redis:latest
docker.io/library/redis:latest
```

3. Pour lancer le conteneur :

```
docker run --name redis1 -d redis
```

Partie 3. Quelques fonctionnalités en plus de celles décrites dans les vidéos

1) LLEN (nom de la liste)

Renvoie la longueur de la liste stockée pour la clé donnée. Si la clé n'existe pas, elle est interprétée comme une liste vide et la valeur 0 est renvoyée. Une erreur est renvoyée lorsque la valeur stockée dans la clé n'est pas une liste.

Pour tester cette fonctionnalité, on va utiliser les commandes :

- LPUSH ou RPUSH
- LRANGE
- LLEN

```

docker exec -it 2217d8174304 x + v
127.0.0.1:6379> RPUSH cours "BDA"
(integer) 1
127.0.0.1:6379> RPUSH cours "NoSQL"
(integer) 2
127.0.0.1:6379> RPUSH cours "Anglais"
(integer) 3
127.0.0.1:6379> RPUSH cours "Java"
(integer) 4
127.0.0.1:6379> RPUSH cours "P00"
(integer) 5
127.0.0.1:6379> LLEN cours
(integer) 5
127.0.0.1:6379> LRange cours 0 -1
1) "BDA"
2) "NoSQL"
3) "Anglais"
4) "Java"
5) "P00"
127.0.0.1:6379> |

```

2) `LMOVE source destination <LEFT | RIGHT> <LEFT | RIGHT>`

Renvoie et supprime de manière atomique le premier/dernier élément (tête/queue selon l'argument *wherfrom*) de la liste stockée à la source, et pousse l'élément au premier/dernier élément (tête/queue selon l'argument *whereto*) de la liste stockée à destination.

Pour tester cette fonctionnalité, on va créer deux listes (*notes:user1* et *notes:user2*), les remplir avec quelques valeurs et transférer la dernière valeur de la liste *notes:user1* dans la liste *notes:user2*.

```

docker exec -it 2217d8174304 x + v
127.0.0.1:6379> LPUSH notes:user1 BDA:11
(integer) 1
127.0.0.1:6379> LPUSH notes:user1 NoSQL:15
(integer) 2
127.0.0.1:6379> LPUSH notes:user2 BDA:8
(integer) 1
127.0.0.1:6379> LMOVE notes:user1 notes:user2 LEFT LEFT
"NoSQL:15"
127.0.0.1:6379> LRange notes:user1 0 -1
1) "BDA:11"
127.0.0.1:6379> LRange notes:user2 0 -1
1) "NoSQL:15"
2) "BDA:8"
127.0.0.1:6379>

```

3) `PUBSUB NUMPAT [channel [channel ...]]`

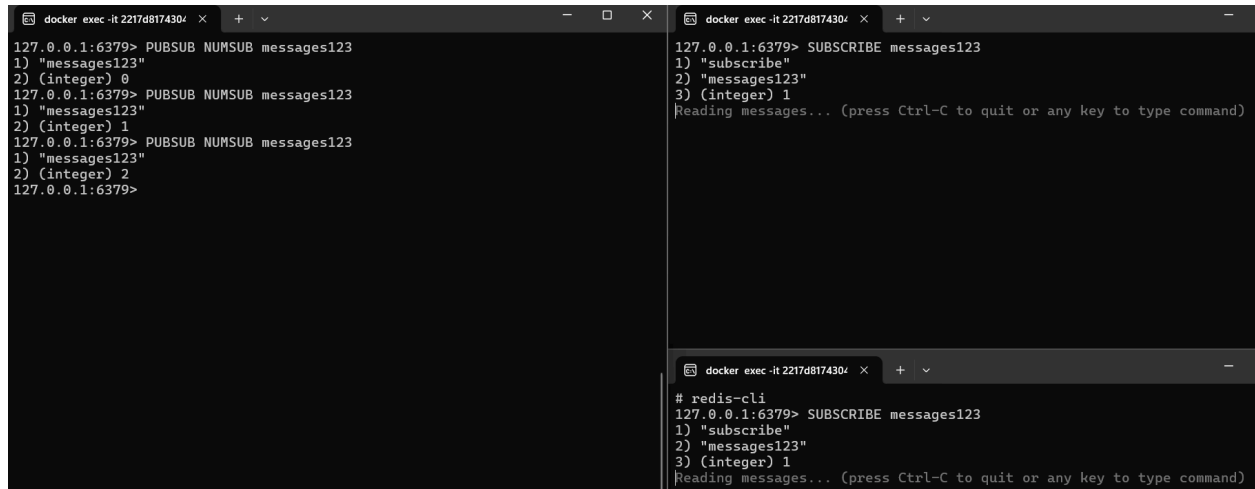
Renvoie le nombre d'abonnés (à l'exclusion des clients abonnés aux modèles) pour les canaux spécifiés.

```

docker exec -it 2217d8174304 x + v
127.0.0.1:6379> PUBSUB NUMSUB messages123
1) "messages123"
2) (integer) 0
127.0.0.1:6379> PUBSUB NUMSUB messages123
1) "messages123"
2) (integer) 1
127.0.0.1:6379>

# redis-cli
127.0.0.1:6379> SUBSCRIBE messages123
1) "subscribe"
2) "messages123"
3) (integer) 1
Reading messages... (press Ctrl-C to quit or any key to type command)

```



```
docker exec -it 2217d8174304 x + v
127.0.0.1:6379> PUBSUB NUMSUB messages123
1) "messages123"
2) (integer) 0
127.0.0.1:6379> PUBSUB NUMSUB messages123
1) "messages123"
2) (integer) 1
127.0.0.1:6379> PUBSUB NUMSUB messages123
1) "messages123"
2) (integer) 2
127.0.0.1:6379>

docker exec -it 2217d8174304 x + v
127.0.0.1:6379> SUBSCRIBE messages123
1) "subscribe"
2) "messages123"
3) (integer) 1
Reading messages... (press Ctrl-C to quit or any key to type command)

# redis-cli
127.0.0.1:6379> SUBSCRIBE messages123
1) "subscribe"
2) "messages123"
3) (integer) 1
Reading messages... (press Ctrl-C to quit or any key to type command)
```

4) PING

Renvoie PONG si aucun argument n'est fourni, sinon renvoie une copie de l'argument en bloc. Cette commande est utile pour :

- Tester si une connexion est toujours active.
- Vérifier la capacité du serveur à servir des données : une erreur est renvoyée lorsque ce n'est pas le cas (par exemple, lors du chargement à partir de la persistance ou de l'accès à une réplique obsolète).
- Mesurer la latence.



```
docker exec -it 2217d8174304 x + v
127.0.0.1:6379> PING "Hello"
"Hello"
127.0.0.1:6379> PING
PONG
127.0.0.1:6379>
```