

TP1 : Redis, une base de données NoSQL rapide et performante

Réalisée par : Amal TAOUS 12204425

Introduction à Redis

Redis (Remote Dictionary Server) est une base de données NoSQL open source, principalement utilisée pour le stockage en mémoire vive. Grâce à cette architecture, Redis offre des performances exceptionnelles avec des temps de réponse extrêmement rapides, même pour des millions de requêtes par seconde. Il est utilisé dans divers secteurs comme les jeux vidéo, la publicité, les services financiers et la santé.

Les données dans Redis sont organisées sous forme de paires clé-valeur. La clé est une chaîne unique, et la valeur peut être de différents types de structures de données :

- Chaînes (Strings): Valeurs textuelles simples.
- Listes (Lists): Collections ordonnées d'éléments.
- Ensembles (Sets): Collections non ordonnées d'éléments uniques.
- Ensembles triés (Sorted Sets): Ensembles où chaque élément est associé à un score permettant le tri.
- Hashs (Hashes): Tableaux associatifs avec des paires champ-valeur.

Redis peut également persister les données sur disque pour garantir leur disponibilité en cas de panne

Connexion à Redis

1. Lancer le serveur Redis:

redis-server

2. Se connecter au client Redis :

Ouvrez un autre terminal et utilisez la commande suivante pour interagir avec Redis :

redis-cli

Commandes de base et opérations CRUD

1. Créer (Create) :

SET KEY VALUE -> Définit une clé et une valeur

- Chaînes :

SET nom "Jean Dupont"

SET age 30

- Hashs :

HSET utilisateur:1 nom "Jean Dupont"

HSET utilisateur:1 age 30

HMSET utilisateur :1 nom"Jean Dupont" age 30

- Listes:

R PUSH fruits "pomme" : pour ajouter l'élément pomme à droite de la liste fruits

L PUSH legumes "carotte": pour ajouter l'élément carotte à gauche de la liste fruits

- Ensembles :

SADD couleurs "rouge" "vert" "bleu"

PS : les valeur doivent etre distincts les uns des autres

- Ensembles triés :

ZADD scores 10 "Alice" 20 "Bob"

2. Lire (Read)

- Récupérer une chaîne :

GET nom # Renvoie "Jean Dupont"

GET age # Renvoie 30

- Lire un hash :

HGET utilisateur:1 nom # Renvoie "Jean Dupont"

HGET utilisateur:1 age # Renvoie 30

HGETALL utilisateur:1 # Renvoie tous les champs et valeurs du hash

HVALS utilisateur :1 # Renvoie que les valeurs de cet utilisateur

- Lister les éléments d'une liste :

LRANGE list_key START STOP

LRANGE fruits 0 -1 # Affiche tous les éléments de la liste fruits

- Afficher un ensemble:

SMEMBERS couleurs # Affiche tous les éléments uniques du set

- Afficher un ensemble trié:

ZRANGE scores 0 -1 # Tri croissant selon le score

ZREVRANGE scores 0 -1 # Tri décroissant selon le score

ZRANK scores "valeurs" # Renvoie le rang de cette valeur

2. Mettre à jour (Update)

SET KEY new_value : met à jour la Valeur de la clé existante

- Mettre à jour une chaîne existante :

SET nom "Pierre Martin"

- Modifier un champ dans un hash :

HSET utilisateur:1 age 31

- Incrémenter une valeur entière :

INCR age # Incrémente l'âge de +1

4. Supprimer (Delete)

- Supprimer une clé :

DEL nom

- Supprimer un champ dans un hash :

HDEL utilisateur:1 age

- Supprimer un élément dans une liste:

LPOP nomdelaliste : supprime l'élément de la gauche

RPOP nomdelaliste : supprime l'élément de la droite

- Supprimer un élément d'un set:

SREM nomdelaliste "nom de la valeurs à supprimer"

- Supprimer tous les éléments d'une base de données courante (opération irréversible) :

FLUSHDB

Gestion des données sur disque dans Redis

Bien que Redis stocke principalement ses données en mémoire vive pour maximiser la rapidité, il peut également persister ces données sur disque via deux mécanismes principaux:

1. Snapshots (RDB): Sauvegardes périodiques des données sous forme binaire.
2. Journalisation (AOF): Enregistrement séquentiel des commandes pour reconstruire l'état exact.

Ces mécanismes garantissent que les données peuvent être restaurées après une panne.

Commandes de base du Pub/Sub dans Redis

Souscription à un canal :

- La commande SUBSCRIBE permet à un client de s'abonner à un ou plusieurs canaux pour recevoir les messages publiés sur ces derniers.

```
SUBSCRIBE channel_name
```

Publication d'un message :

- La commande PUBLISH envoie un message sur un canal spécifique. Tous les clients abonnés à ce canal recevront le message.

```
PUBLISH channel_name "message"
```

Exemple pratique

1. Lancer le serveur Redis:

ouvrez un terminal et tapez la commande suivante

```
redis-server
```

2. Se connecter au client 1 Redis :

Ouvrez un autre terminal et utilisez la commande suivante pour interagir avec Redis :

```
redis-cli
```

3. Se connecter au client 2 Redis :

Ouvrez un autre terminal et utilisez la commande suivante pour interagir avec Redis :

```
redis-cli
```

Client 1 : S'abonner au canal "redisChat" :

```
SUBSCRIBE redisChat
```

Client 2 : Publier des messages sur "redisChat" :

```
PUBLISH redisChat "Message 1"
```

```
PUBLISH redisChat "Message 2"
```

Résultat (Client 1) :

Le client 1 recevra les messages suivants :

1) "message"

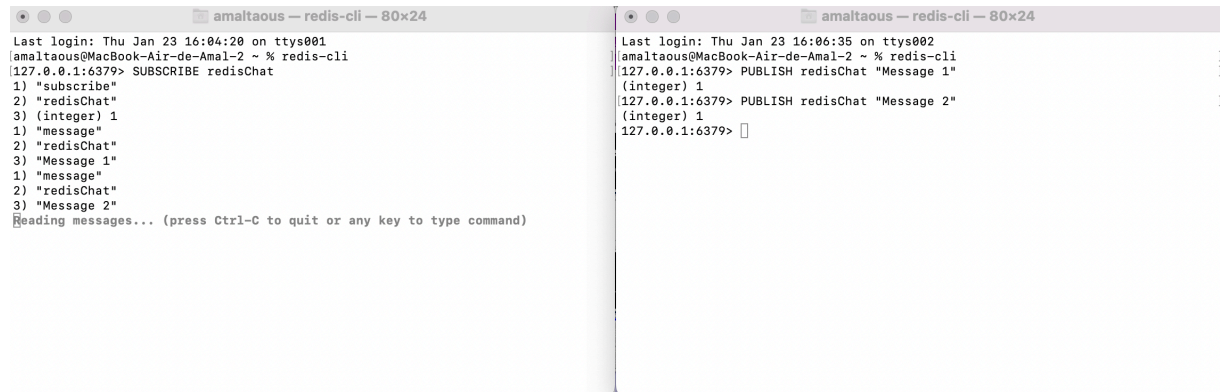
2) "redisChat"

3) "Message 1"

1) "message"

2) "redisChat"

3) "Message 2"



```
amaltaous — redis-cli — 80x24
Last login: Thu Jan 23 16:04:20 on ttys001
amaltaous@MacBook-Air-de-Amal-2 ~ % redis-cli
127.0.0.1:6379> SUBSCRIBE redisChat
1) "subscribe"
2) "redisChat"
3) (integer) 1
1) "message"
2) "redisChat"
3) "Message 1"
1) "message"
2) "redisChat"
3) "Message 2"
Reading messages... (press Ctrl-C to quit or any key to type command)

amaltaous — redis-cli — 80x24
Last login: Thu Jan 23 16:06:35 on ttys002
amaltaous@MacBook-Air-de-Amal-2 ~ % redis-cli
127.0.0.1:6379> PUBLISH redisChat "Message 1"
(integer) 1
127.0.0.1:6379> PUBLISH redisChat "Message 2"
(integer) 1
127.0.0.1:6379> 
```