TP 1 - Redis : Introduction et Commandes de Base

Table des matières

Préambule

Redis est une base de données NoSQL de type clé-valeur. Elle est généralement utilisée pour gérer des données en RAM, ce qui offre des temps d'accès très rapides. Cependant, cette caractéristique signifie également que les données ne sont pas persistées sur le disque par défaut et peuvent être perdues en cas d'incident. Redis est souvent combinée avec une base relationnelle pour assurer la persistance.

1 Partie 1 : Initiation et Commandes de Base

1.1 Installation de Redis

Prérequis : Docker

1. Télécharger l'image Redis depuis Docker Hub :

```
docker pull redis
```

Rôle : Cette commande télécharge l'image officielle de Redis depuis Docker Hub, prête à être utilisée pour créer des conteneurs.

2. Lancer un conteneur Redis :

```
docker run --name mon-redis -d redis
```

Rôle : Crée et exécute un conteneur Redis en arrière-plan. Le conteneur est nommé `mon-redis`.

3. Accéder au serveur Redis :

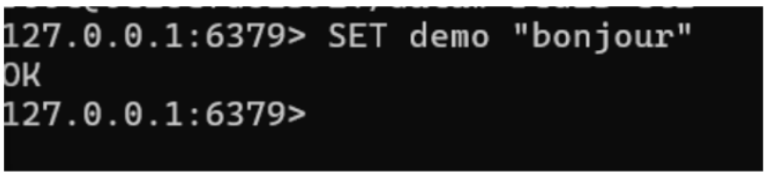
```
docker exec -it mon-redis bash
redis-server --port 6380
```

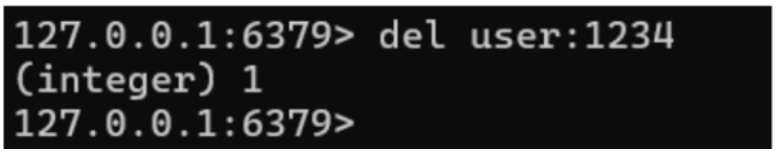
Rôle : L'accès au conteneur permet d'interagir directement avec le serveur Redis. La commande `redis-server --port 6380` démarre Redis sur un port spécifique.

4. Accéder au client Redis :

```
docker exec -it mon-redis bash
redis-cli
```

Rôle : Lance le client Redis CLI pour permettre l'exécution de commandes.





1.2 Définir une clé-valeur

— Commande pour définir une clé :

```
SET demo "bonjour"
```

Rôle : Définit une clé `demo` avec la valeur `bonjour`. Redis renvoie une confirmation `OK` si l'opération est réussie.

1.3 Opérations CRUD avec Redis

Redis permet de : créer, lire, modifier et supprimer des données. Voici quelques exemples :

— **Créer une clé-valeur** :

```
127.0.0.1:6379> SET user:1234 "ghassen"
OK
```

Rôle : Ajoute une clé `user:1234` avec la valeur `ghassen`.

— **Lire une valeur à partir d'une clé** :

```
127.0.0.1:6379> GET user:1234
"ghassen"
```

Rôle : Récupère la valeur associée à la clé `user:1234`.

— **Supprimer une clé** :

```
127.0.0.1:6379> DEL user:1234
(integer) 1
```

Rôle : Supprime la clé `user:1234`. Si la clé n'existe pas :

```
127.0.0.1:6379> DEL user:123456
(integer) 0
```

Rôle : Tente de supprimer une clé inexistante. Redis renvoie `0` pour indiquer qu'aucune suppression n'a été effectuée.

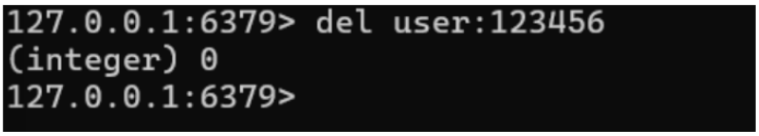


FIGURE 1 – Enter Caption

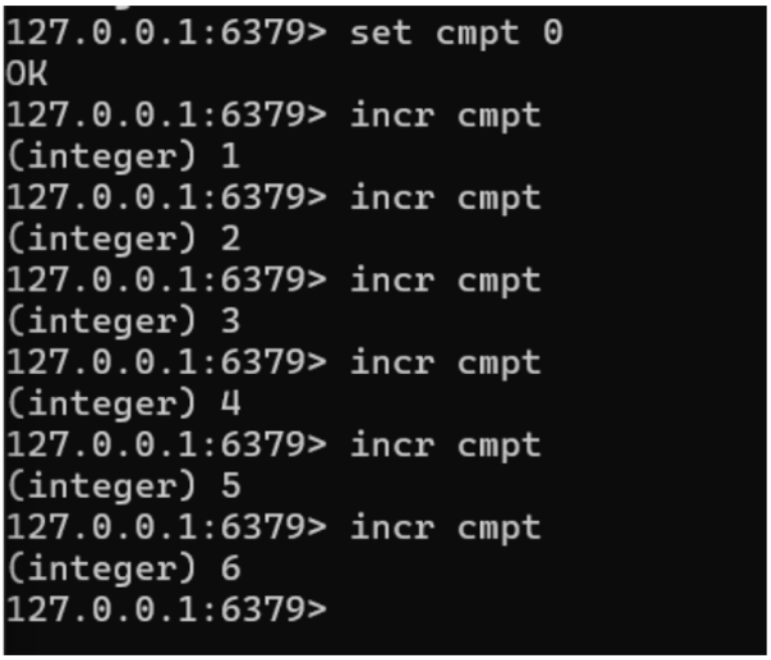


FIGURE 2 – Enter Caption

Compteur de visiteurs Exemple pratique : gestion d'un compteur de visiteurs.

1. Initialiser un compteur :

```
SET cmpt 0
```

Rôle : Crée un compteur initialisé à 0.

2. Incrémenter le compteur :

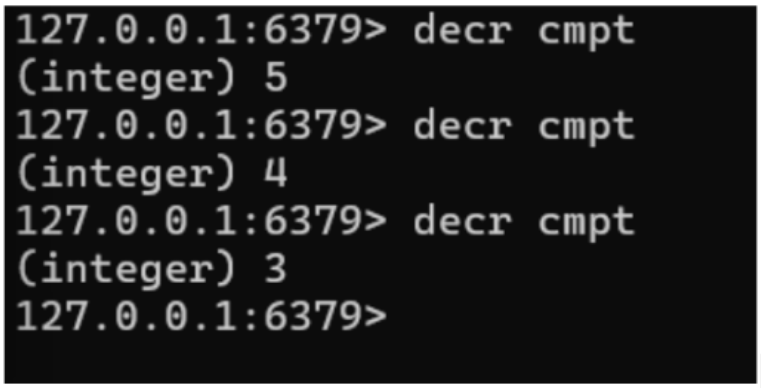
```
INCR cmpt
```

Rôle : Augmente la valeur du compteur de 1.

3. Décrémenter le compteur :

```
DECR cmpt
```

Rôle : Réduit la valeur du compteur de 1.



Durée de vie d'une clé

— Définir une clé avec une durée de vie :

```
SET uneCle uneValeur
EXPIRE uneCle 30
```

Rôle : Associe une durée de vie (en secondes) à une clé. Ici, la clé `uneCle` expirera après 30 secondes.

— Consulter la durée restante :

```
TTL uneCle
```

Rôle : Renvoie le temps restant avant l'expiration de la clé.

1.4 Structures Avancées

Listes

— Ajouter des éléments dans une liste :

```
RPUSH mesCours "BDA"
RPUSH mesCours "WEB"
```

Rôle : Ajoute des éléments à la fin de la liste `mesCours`.

— Afficher la liste :

```
LRANGE mesCours 0 -1
```

Rôle : Affiche tous les éléments de la liste.

— Supprimer un élément :

```
RPOP mesCours
LPOP mesCours
```

Rôle : Supprime respectivement un élément depuis la droite (`RPOP`) ou la gauche (`LPOP`) de la liste.

Ensembles

— Créer un ensemble :

```
SADD myUsers ghassen
```

Rôle : Ajoute un élément à l'ensemble `myUsers`.

— Visualiser les éléments :

```
SMEMBERS myUsers
```

Rôle : Renvoie tous les éléments de l'ensemble `myUsers`.

— Supprimer un élément :

```
SREM myUsers ghassen
```

Rôle : Supprime un élément spécifique de l'ensemble.

1.5 Pub/Sub : Messagerie en Temps Réel

— **Client 1 : écouter un canal**

```
SUBSCRIBE mesCours
```

Rôle : Écoute les messages publiés sur le canal `mesCours`.

— **Client 2 : envoyer un message**

```
PUBLISH mesCours "Message␣de␣test␣:)"
```

Rôle : Publie un message sur le canal `mesCours`.

1.6 Sélection de Base de Données

Redis offre la possibilité d'utiliser plusieurs bases (par défaut 16).

```
127.0.0.1:6379> SELECT 1
127.0.0.1:6379[1]> KEYS *
```

Rôle : Change la base de données active (ici la base 1). La commande `KEYS *` affiche les clés disponibles dans cette base.

2 Partie 2 : Systèmes de Recommandation avec Redis

2.1 Sorted Sets (ZSET)

— Ajouter des éléments :

```
ZADD score1 10 "Karim" 20 "Ahmed"
```

Rôle : Ajoute des éléments avec des scores au sorted set `score1`.

— Afficher les éléments triés :

```
ZRANGE score1 0 -1
ZREVRANGE score1 0 -1
```

Rôle : Affiche les éléments dans l'ordre croissant (`ZRANGE`) ou décroissant (`ZREVRANGE`).

— Consulter le rang d'un utilisateur :

```
ZRANK score1 "Karim"
```

Rôle : Renvoie le rang de l'utilisateur `Karim`.

3 Partie 3 : Gestion de Données avec Hashes

3.1 Utilisation de Hashes

— Définir un hash :

```
HSET user:11 username "aa" age 23 email "aa@aa.fr"
```

Rôle : Crée un hash `user:11` contenant plusieurs champs et valeurs.

— Consulter toutes les données :

```
HGETALL user:11
```

Rôle : Renvoie tous les champs et valeurs associés au hash `user:11`.

— Incrémenter une valeur :

```
HINCRBY user:11 age 5
```

Rôle : Augmente la valeur du champ `age` de 5.

— Afficher uniquement les valeurs :

```
HVALS user:11
```

Rôle : Renvoie toutes les valeurs associées au hash `user:11`.