Nom : Loubna MANGOUCHI
N°Étudiant : 124099931
Classe : INFO3

Introduction pratique à Redis

1. Introduction

À travers trois vidéos successives, j'ai découvert et manipulé les principales structures et commandes de Redis, une base de données NoSQL orientée **clé–valeur**, très rapide car fonctionnant principalement **en mémoire RAM**.
L'objectif était de comprendre les types de données fondamentaux, les opérations de base, ainsi que des fonctionnalités avancées comme les sorted sets, les hashes et le système de publication/souscription.

2.Structures de base et commandes essentielles: Vid1

Dans la première partie, j'ai appris à utiliser les opérations CRUD sur les *strings* :

- **SET**, **GET**, **DEL** pour créer, lire et supprimer une clé.
- **INCR** et **DECR** pour gérer des compteurs.
- **TTL** et **EXPIRE** pour définir une durée de vie sur une clé.

J'ai ensuite manipulé deux structures simples :

Les Listes

- **LPUSH / RPUSH** pour ajouter des éléments
- **LRANGE** pour consulter une partie ou toute la liste
- **LPOP / RPOP** pour retirer un élément

Les Sets (ensembles)

- **SADD**, **SMEMBERS**, **SREM**

  valeurs uniques, sans ordre, idéal pour des collections où les doublons sont interdits.

3. Structures avancées : Sorted Sets et Hashes: Vid2

Ensembles ordonnés (ZSET)

Ce type permet d'associer un **score numérique** à chaque membre et de récupérer les éléments automatiquement triés :

- **ZADD** pour ajouter des scores
- **ZRANGE** et **ZREVRANGE** pour afficher du plus petit au plus grand (ou inversement)
- **ZRANK** pour obtenir la position d'un élément dans le classement

Ces opérations sont très utilisées pour les **classements**, les **recommandations** et les systèmes temps réel nécessitant un tri performant.

Hashes (Haches)

Ils permettent de regrouper plusieurs champs pour une même clé, comme un petit objet JSON :

- **HSET**, **HMSET** pour définir les champs
- **HGETALL** pour afficher toutes les informations
- **HINCRBY** pour incrémenter un champ numérique

Les hashes sont très utiles pour représenter un utilisateur ou un objet sans imposer de schéma strict, ce qui donne une grande flexibilité.

L'enseignant a aussi insisté sur la différence de performance entre un accès RAM (très rapide) et un accès disque (beaucoup plus lent), ce qui justifie l'usage de Redis comme **cache**.

4.Publication / Souscription: Vid3

Dans cette partie, j'ai découvert la fonctionnalité **Pub/Sub**, utilisée pour envoyer des messages en temps réel.

Subscribe (abonnement à un canal): SUBSCRIBE mescours

Le client reste en attente et reçoit instantanément tous les messages publiés sur ce canal.

Publish (envoyer un message): PUBLISH mescours "nouveau cours MongoDB"

Tous les utilisateurs abonnés au canal le reçoivent immédiatement.

Pattern subscribe (abonnement global): `PSUBSCRIBE me*`

reçoit tous les messages des canaux commençant par "me".

5.Gestion des bases internes

Redis comporte **16 bases internes** (0 à 15). Il est possible de changer de base avec : `SELECT 1`

et de revenir à la base 0 lorsque nécessaire.

On peut aussi utiliser :

- **KEYS *** pour afficher toutes les clés de la base sélectionnée.

6.Conclusion

Ces trois vidéos m'ont permis de maîtriser les concepts essentiels de Redis :

- Manipulation des clés simples
- Utilisation des listes, sets et sorted sets
- Stockage structuré avec hashes
- Communication temps réel via Pub/Sub
- Gestion des TTL et des bases internes

Redis se révèle être un outil **rapide, flexible et adapté aux applications modernes**, notamment pour le caching, les classements, les messages temps réel et la gestion de données légères à haute fréquence.