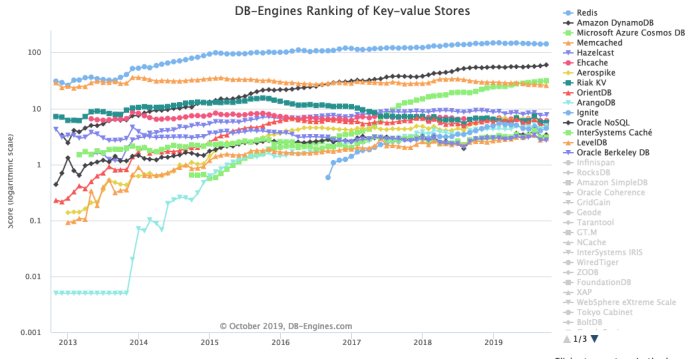

redis
LOCKE
&
KEY

# Plan

- Qu'est ce que REDIS ?
- Les fondamentaux
- Comment utiliser REDIS ?
- La haute disponibilité

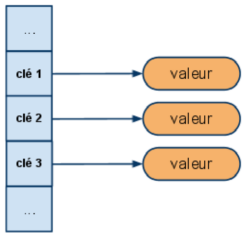
# Qu'est ce que Redis ?

- Créé en 2009
- Initialement conçu pour améliorer les performances de l'outil d'analyse des visites de sites Web.
- *Salvatore SANFILIPPO contributeur majeur de REDIS définit Redis de la manière suivante :*
  - « I see Redis definitely more as a flexible tool than as a solution specialized to solve a specific problem: his mixed soul of cache, store, and messaging server shows this very well »
- D'après DB-Engines, Redis est classée comme la 1ere base de données clé/valeur.



# Qu'est ce que Redis ?

- REDIS est l'acronyme de REmote Dictionary Server : serveur de dictionnaire distant.
- C'est un dictionnaire qui associe à une clé une et une seule valeur.
- Il n'y pas de table où de schéma prédéfini.
- Conserve les données en mémoire et peut être configuré pour conserver les données.
- Redis Enterprise offre les concepts de réplication pour la haute disponibilité et le sharding pour l'extensibilité horizontale.



# Le principe de Redis

- Conserve l'intégralité des données en RAM
  - Permet d'obtenir d'excellentes performances en évitant les accès aux disques qui sont généralement lents.
  - Redis peut répondre en lecture à 110 000 interrogations par seconde et en écriture à 81000 insertions par seconde.
- Ces performances en font une référence pour les systèmes de cache.
- Exemple : La page d'un site web est généralement générée dynamiquement à la demande. Le temps de traitement de la page peut coûter chère et stocker le résultat dans un cache comme redis améliorerait significativement les temps de réponse.
- Support de nombreux types de données.
  - SET(tableau), LIST (liste), HASH(dictionnaire),ZHASH (dictionnaire ordonné) …
- Chaque commande est une transaction atomique.
- Il dispose de certaines fonctionnalités avancées comme la notion d'abonnement, un système de notification, l'expiration d'une clé.
- Mise en œuvre dans de grandes entreprises :
  - The Guardian (quotidien d'information britannique), GitHub, Stack Overflow, YouPorn, Twitter

## Les Fondamentaux

- Dans ce slide, nous présentons succinctement la grammaire des commandes REDIS.
- Pour définir un couple de clé/valeur
  - SET user Bob -> OK
- Demander la valeur d'une clé
  - GET user -> Bob
- Supprimer un clé
  - DEL user
- Quelques spécificités sur clés :
  - Ne doit pas contenir d'espace et de retour à la ligne.
  - Le nom de la clé suit généralement le pattern suivant :
    - **« object-type:id » cela permet decontenir de l'information dans le nom de la clé** exemple **« user:1000 »**
    - **« object-type:id:field » pour simuler des champs** Exemple : « **user:1001:name** »

# Les Fondamentaux

- La valeur d'une clé peut être de différents types :
  - String (chaînes de caractères)
  - List (listes)
  - Set (tableaux)
  - Sorted set (tableaux ordonnés)
  - Hash (dictionnaires)

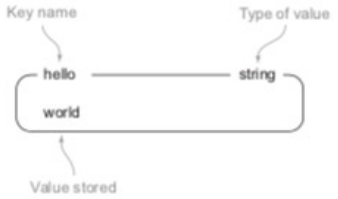



Figure 1.1 **An example of a `STRING`, `world`, stored under a key, `hello`**

- Atomic counter (compteurs numériques). Attention, les données numériques sont stockées dans la base sous la forme de chaîne de caractères

- La valeur d'une clé peut être de différents types :

String (chaînes de caractères)

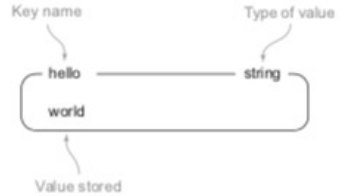



Figure 1.1 An example of a STRING, world, stored under a key, hello

List (listes)

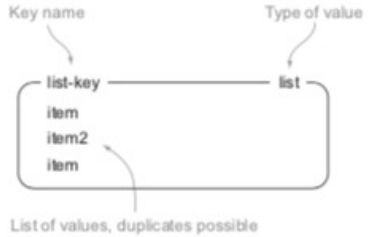



Figure 1.2 An example of a LIST with three items under the key, list-key. Note that item can be in the list more than once.

Set (tableau)

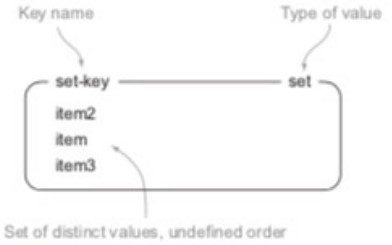
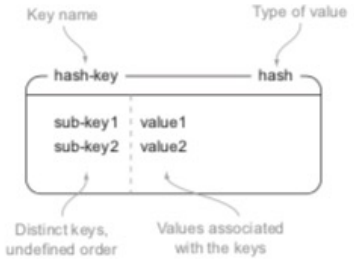
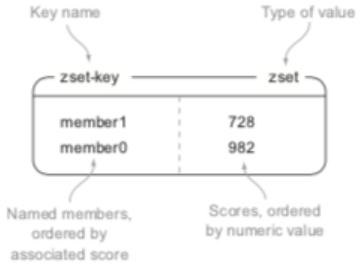



Figure 1.3 An example of a SET with three items under the key, set-key

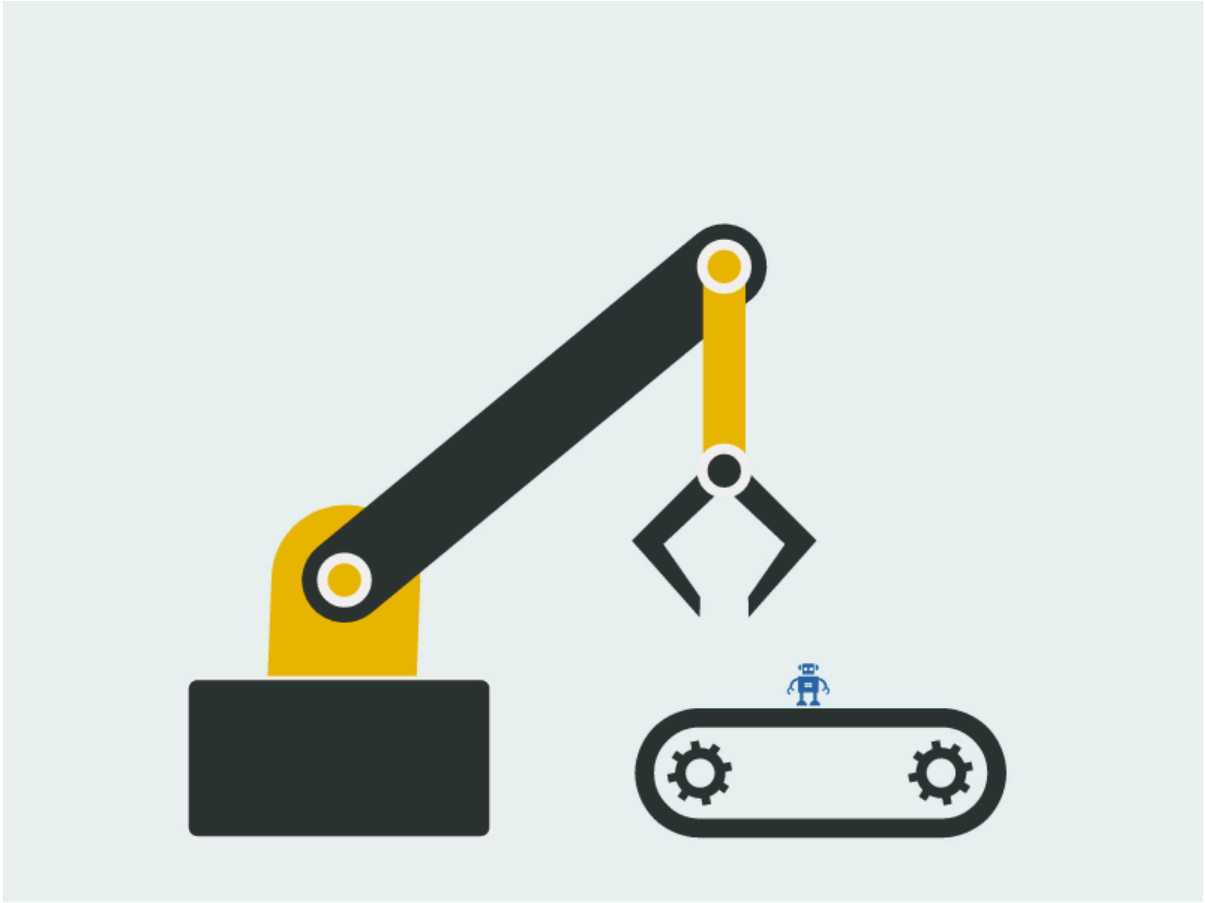
Hash (dictionnaire)




Sorted set (tableaux ordonnés)



# TP : chap5_REDIS_TP01 / TP : chap5_REDIS_TP02 / TP : chap5_REDIS_TP03



## Le modèle de données

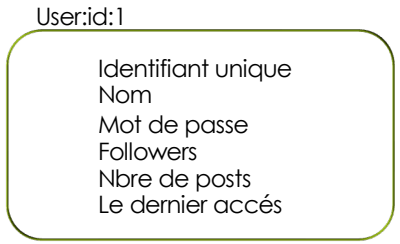
- Fondamentalement, Redis est un serveur de structure de données.
- Redis n'intègre pas de schéma, pour l'intégrité cela se passera du coté de l'application.
- Même si Redis assure la persistance des données, les applications ne l'utilisent pas comme solution unique de stockage mais comme une couche en amont de la base de données relationnelles.
  - cache de données.
- Redis est une manière d'améliorer les performances de vos applications. Il n'a pas pour vocation d'implémenter un modèle de données exhaustif comme les BDD relationnelles avec la normalisation. Redis fait simple pour des choses simples.
- Structurer vos données de manière à minimiser le nombre d'échanges avec le serveur.

# Le modèle de données

- La façon dont vous structurez vos données dépend entièrement de ce qu'elles sont et de la manière dont vous devez y accéder.
- La manière de penser avec Redis est très similaire à l'algorithmique de structure de données.
- Pour modéliser vos données vous devrez :
  - Déterminer les entités que vous allez manipuler ? Un compte utilisateur, des messages, des personnes, des emails.
  - Déterminer les fonctionnalités que vous devrez implémenter ?
    - Créer / supprimer un compte utilisateur, permettre à un utilisateur de se connecter …
    - Créer / supprimer un message, noter un message…
  - Vous ne structurez que les données dont vous avez besoins pour implémenter les fonctionnalités.

# Le modèle de données

- Exemple de modélisation
  - Essayons de modéliser Twitter de manière simplifiée.
  - Qu'est ce que fera notre twitter ?
    - Il permettra à un utilisateur d'envoyer de brefs messages, appelés *tweets*, entre utilisateurs.
    - L'utilisateur devra créer un compte.
    - Un compte peut avoir des followers.
    - L'utilisateur peut consulter ses tweets.
  - Qu'est ce qu'un utilisateur ?
    - L'utilisateur est caractérisé par son nom, mot de passe, le nombre de followers, le nombre de tweets et son dernier accès. Toutes ses informations sont disponibles dans le profile de l'utilisateur. On privilégiera donc un structure hash.



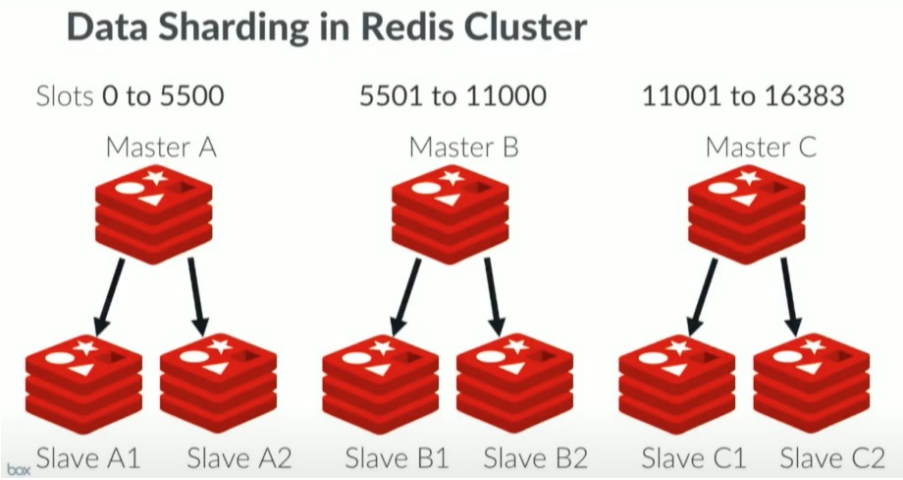
# La haute disponibilité

https://severalnines.com/blog/hash-slot-vs-consistent-hashing-redis/

There are 16384 hash slots in Redis Cluster (Redis clustering follows a distributed slot map approach which distribute 16384 slots among Cluster nodes), and to compute what is the hash slot of a given key, we simply take the CRC16 of the key modulo 16384.

Every node in a Redis Cluster is responsible for a subset of the hash slots, so for example you may have a cluster with 3 nodes, where:

- Node A contains hash slots from 0 to 5500.
- Node B contains hash slots from 5501 to 11000.
- Node C contains hash slots from 11001 to 16383.



# TP : chap5_REDIS_TP04