Master Mathématiques, Informatique, Décision, Organisation (MIDO)
2ère année - MIAGE-IF App.

ANNEE 2024/ 2025

Désignation de l'enseignement : NoSQL

Nom du document : TP Clé-Valeur sous Redis et PostgreSQL

Rédacteur : Maude Manouvrier

La reproduction de ce document par tout moyen que ce soit est interdite conformément aux articles L111-1 et L122-4 du code de la propriété intellectuelle

I. Manipuler des données clé-valeur sous Redis

Le détail de chaque commande Redis (sa description, des exemples et la complexité) est disponible sur : `https://redis.io/commands/`

Vous pouvez travailler sur redis (sans l'installer sur votre machine) en utilisant la console en ligne `https://onecompiler.com/redis` où vous pouvez saisir et exécuter plusieurs instructions à la fois.

A. Exercice 1

1. Créer la clé `User` avec comme valeur `Dario`
2. Tester la commande suivante pour renommer la clé :
   `RENAME User User:1`
3. Tester les commandes suivantes pour renommer la clé et analyser leur résultat (vous pouvez vous aider en aller voir dans la documentation référencée plus haut) :
   ```
   SET User:2 Toto
   RENAME User:1 User:2
   GET User:1
   EXISTS User:1
   GET User:2
   SET User:3 Toto
   RENAMENX User:3 User:2
   GET User:3
   GET User:2
   RENAMENX User:3 User:1
   GET User:1
   EXISTS User:3
   ```
4. Créer, avec une seule commande, les 2 clés suivantes (cf. commande `MSET`)
   a. `User:3` avec comme valeur `Maud`
   b. `User:4` avec comme valeur `Xavier`
5. Lister toutes les clés définies dans la base (cf. commande `KEYS`)
6. Lister toutes les clés commençant par `User`
7. Lister les valeurs des différentes clés (cf. command `MGET`)
8. Ajouter un '`e`' à la fin de valeur de la clé (cf. command `SETRANGE`)
9. Tester la commande : `SET User:3 "Aude" NX`
10. Créer une clé temporaire `User:5` avec comme valeur `Toto` de durée de vie 30 secondes (cf. commande `SET` *`option`* `EX`)
11. Vérifier la durée de vie de la clé `User:5` (cf. command `TTL`)
12. Vérifier la durée de vie de la clé `User:1`

13. Créer les clés suivantes :
    a. `User:6` avec pour valeur `Toto`
    b. `User:6:City` pour valeur `Paris`
    c. `User:6:Age` avec pour valeur 25
    d. `User:6:Activity` avec pour valeur `Tutorial` et une expiration dans 10 minutes (avec la commande `SET`)
    e. Vérifier la durée de vie de la clé `User:6:Activity`
14. Ajouter, à la valeur de `User:6:Activity`, la valeur `" Redis"` (cf. commande `APPEND`) et vérifier la valeur de `User:6:Activity`
15. Tester les 2 instructions suivantes et analyser leur résultat :

```
MSETNX User:6:Note 20 User:4:Time 5
MSETNX User:6:Note 20 User:4:Validation Redis
MSETNX User:5:Note 20 User:4:Validation Redis
```

16. Créer un utilisateur avec des champs similaires en utilisant la commande `HSET`.
17. Tester les commandes `HKEYS` et `HVALS` sur ce nouvel utilisateur.
18. Tester la commande suivante pour insérer une valeur en JSON :

```
SET User:8 {"nom":"Titi","age":23,"Activity":["Redis","MongoDB"]}
```

19. Créer une liste `Cours` en y ajouter la valeur `NoSQL` (cf. commande `LPUSH`)
20. Ajouter la valeur `MicroServices` dans la liste `Cours` (cf. commande `LPUSH`)
21. Lister les éléments de la liste (cf. commande `LRANGE`)
22. Exécuter la commande `MULTI`
23. Créer une liste `Cours2` en y ajouter la valeur `NoSQL` (cf. commande `LPUSH`)
24. Ajouter la valeur `MicroServices` dans la liste `Cours` (cf. commande `LPUSH`)
25. Lister les éléments de la liste (cf. commande `LRANGE`)
26. Exécuter la commande `EXEC`

B. Exercice 2[1]

Antoine et Thomas organisent une soirée. Antoine veut inviter : Sophie, Etienne, Hélène, Paul, Charles, Ethan et Pauline. Thomas lui souhaite inviter : Etienne, Paul, Catherine, Ethan, Pauline, Sophia et Mickaël.

Ayant un espace limité, ils souhaitent inviter uniquement leurs amis communs. Qui sont-ils ?

La semaine suivante, ils souhaitent organiser une soirée avec les personnes qui n'ont pas été invité à la première. À l'aide de la documentation, établissez une liste des invités d'Antoine et une liste des invités de Thomas dans de nouveaux `SET`.

Quelle-est la liste finale des invités pour cette seconde soirée ? Une fois établie, exportez-là dans un nouveau set avec l'aide de la documentation.

[1] Repris de https://github.com/mbaumanndev/nosql-course-redis

II. Manipuler des données clé-valeur sous PostgreSQL

Pour créer une base de données en ligne sous PostgreSQL, vous pouvez utiliser le site : `https://sqlfiddle.com`. Ce site permet de créer en ligne une base de données temporaire et de l'interroger en SQL. Il faut sélectionner le SGBD PostgreSQL.

Le détail des commandes permettant de gérer des paires clé-valeur sous PostgreSQL est disponible à l'adresse suivante :
`https://www.postgresql.org/docs/current/hstore.html`
Vous pouvez vous inspirer du tutoriel suivant pour la suite en plus de la documentation PostgreSQL : `https://www.postgresqltutorial.com/postgresql-tutorial/postgresql-hstore/`

Exercice 3

1. Si vous utilisez `dbfiddle`, saisissez l'instruction suivante (à gauche) :
   `CREATE EXTENSION hstore;`
   pour installer l'extension permettant de gérer des couples clé-valeur sous PostgreSQL.
   Sous `sqlfiddle`, cette instruction est inutile, l'extension étant déjà installée.
2. Créer une relation (table) nommée utilisateur[2] avec une clé primaire artificielle nommé `id` et un champ data de type `hstore`.
3. Insérer 4 nuplets représentant 3 utilisateurs : le 1er (identifié par 1) doit avoir un couple clé-valeur (`nom,Dario`), le 2ème (identifié par 2) le couple (`nom, Toto`), (`nom, Maud`), et le 3me (identifié par 3) le couple (`nom, Xavier`).
4. Faire une requête permettant d'afficher le nom de l'utilisateur 1.
5. Créer une requête permettant d'ajouter un nouveau utilisateur associé à 3 couples clé-valeur (`nom,Toto), (Ville, Paris), (Age,25).`
6. Afficher l'id et la valeur de la clé `Ville` de chaque nuplet.
7. Ajouter un couple (`Activite,Redis`) à l'utilisateur 4.
8. Afficher toutes les clés des couples clé-valeur stockés dans la relation `utilisateur`.
9. Afficher toutes les valeurs des couples clé-valeur stockés dans la relation `utilisateur`.
10. Exécuter la requête suivante :
    ```
    SELECT id,(EACH(data) ).* FROM utilisateur;
    ```
11. Exécuter la requête suivante :
    ```
    SELECT key, count(*) FROM
      (SELECT (each(data)).key FROM utilisateur) AS stat
      GROUP BY key
      ORDER BY count DESC, key;
    ```
12. Afficher le contenu du champ `data` sous la forme de document *json*.

[2] NB : PostgreSQL n'autorise pas de nommer une relation `User`.