eBook Gratuit

APPRENEZ redis

eBook gratuit non affilié créé à partir des **contributeurs de Stack Overflow.**

#redis

Table des matières

À propos

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: redis

It is an unofficial and free redis ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official redis.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapitre 1: Commencer avec Redis

Remarques

Cette section fournit une vue d'ensemble de ce qu'est Redis et des raisons pour lesquelles un développeur peut vouloir l'utiliser.

Il devrait également mentionner tous les grands sujets au sein de Redis, et établir un lien avec les sujets connexes. La documentation de Redis étant nouvelle, vous devrez peut-être créer des versions initiales de ces rubriques connexes.

Versions

Version	Date de sortie
3.2.3	2016-08-02
3.2.2	2016-07-28

Examples

Vue d'ensemble

Redis est une base de données distante en mémoire qui offre des performances élevées, une réplication et un modèle de données unique pour produire une plate-forme permettant de résoudre les problèmes. Redis est une structure de données en mémoire ouverte (sous licence BSD), utilisée comme base de données, cache et courtier de messages. Il est classé en tant que magasin clé-valeur NoSQL. Il prend en charge des structures de données telles que des chaînes, des hachages, des listes, des ensembles, des ensembles triés avec des requêtes de plage, des bitmaps, des hyperloglogs et des index géospatiaux avec des requêtes radius. Prenant en charge cinq types différents de structures de données,

1. STRING (Fonctionne sur toute la chaîne, les parties, les entiers et les flottants)
2. LISTE (Push ou pop articles des deux extrémités)
3. SET (Ajouter, extraire, supprimer, vérifier, intersecter, union, différence, etc.)
4. HASH (stocker, coller, supprimer dans le hachage)
5. ZSET (identique à celui défini mais de manière ordonnée)
6. GEO (Ajouter, mettre à jour, supprimer la latitude et la longitude, entrer dans un redius donné)

Redis intègre la réplication, les scripts Lua, l'expulsion LRU, les transactions et différents niveaux de persistance sur disque (sync / async).

Avant la version 3, Redis fonctionnait en mode maître-esclave et exigeait que Redis-Sentinel

fournisse une haute disponibilité. Seul maître accepte les écritures et synchronise les données sur ses esclaves en les falsifiant.

À partir de la version 3, Redis fonctionne et recommande le mode multi-maître où les fonctionnalités de basculement, de partitionnement / partitionnement et de redimensionnement sont intégrées. Redis-Sentinel n'est pas requis à partir de la version 3. Pour que le cluster redis exploite un minimum de 3 nœuds / processus maîtres est requis.

Les fonctionnalités supplémentaires sont la réplication, la persistance et le partitionnement côté client. Redis s'adapte à une grande variété de problèmes qui peuvent être naturellement associés à ce que Redis propose, vous permettant de résoudre vos problèmes sans avoir à effectuer le travail conceptuel requis par d'autres bases de données.

Interface de ligne de commande Redis

`redis-cli` est le programme d'interface de ligne de commande Redis qui permet d'envoyer des commandes à Redis et de lire les réponses envoyées par le serveur, directement depuis le terminal. L'utilisation de la ligne de commande de base est ci-dessous:

Accès aux redis:

```
$ redis-cli
127.0.0.1:6379>
```

Accès aux redis avec authentification:

```
$ redis-cli -a myPassword
127.0.0.1:6379>
```

Sélectionnez la base de données et affichez la taille de la base de données (le numéro de base de données par défaut est 0):

```
127.0.0.1:6379> dbsize
(integer) 2
127.0.0.1:6379> select 1
OK
127.0.0.1:6379[1]> dbsize
(integer) 20
```

Obtenir des informations et des statistiques sur le serveur:

```
127.0.0.1:6379> info
redis_version:2.4.10
redis_git_sha1:00000000
redis_git_dirty:0
arch_bits:64
multiplexing_api:epoll
gcc_version:4.4.6
process_id:947
uptime_in_seconds:873394
uptime_in_days:10
```

```
lru_clock:118108
used_cpu_sys:19.55
used_cpu_user:397.46
used_cpu_sys_children:0.00
used_cpu_user_children:0.00
connected_clients:1
connected_slaves:0
client_longest_output_list:0
client_biggest_input_buf:0
blocked_clients:0
used_memory:14295792
used_memory_human:13.63M
used_memory_rss:19853312
used_memory_peak:14295760
used_memory_peak_human:13.63M
mem_fragmentation_ratio:1.39
mem_allocator:jemalloc-2.2.5
loading:0
aof_enabled:0
changes_since_last_save:0
bgsave_in_progress:0
last_save_time:1468314087
bgrewriteaof_in_progress:0
total_connections_received:2
total_commands_processed:2
expired_keys:0
evicted_keys:0
keyspace_hits:0
keyspace_misses:0
pubsub_channels:0
pubsub_patterns:0
latest_fork_usec:0
vm_enabled:0
role:master
db0:keys=2,expires=0
db1:keys=20,expires=0
```

En sortant du redis-cli:

```
127.0.0.1:6379> exit
```

Redis "Hello World"

Vous devez d'abord installer et démarrer votre serveur Redis, consultez le lien ci-dessous qui peut vous aider à installer Redis sur votre serveur ou votre ordinateur local.

Installation et configuration

Maintenant, ouvrez votre invite de commande et exécutez la commande `redis-cli` :

Pour enregistrer le premier jeu> SET 'nom du fichier' puis 'valeur'

```
127.0.0.1:6379> SET hkey "Hello World!"
```

Appuyez sur Entrée, vous devriez voir

```
OK
```

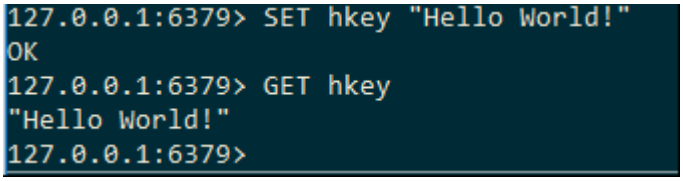
Puis entrez:

```
GET hkey
```

tu devrais voir:

```
"Hello World!"
```

Exemple de sortie d'écran:



Installez Redis en utilisant Docker

Il est simple de commencer à utiliser Redis en utilisant docker:

```
docker pull redis
docker run -p 6379:6379 --rm --name redis redis
```

Maintenant, vous avez une instance en cours d'exécution sur le port `6397`

Attention: Toutes les données seront supprimées lorsque Redis sera arrêté.

Pour connecter le redis-cli, démarrez un autre menu fixe:

```
docker run -it --link redis:redis --rm redis redis-cli -h redis -p 6379
```

Vous pouvez maintenant jouer avec votre docker redis.

Redis installtion sur Windows, avec l'exemple Node.js

Redis dispose d'un port Windows fourni par «Microsoft Open Technologies». Vous pouvez utiliser le programme d'installation msi trouvé sur: https://github.com/MSOpenTech/redis/releases

Une fois l'installation terminée, vous pouvez voir que "Redis" est un service Windows (et son statut doit être "Démarré")

Pour écrire un exemple "Hello world" utilisant Redis dans Node.js (dans Windows également), vous pouvez utiliser le module npm suivant: https://www.npmjs.com/package/redis

exemple de code:

```
var redis = require('redis'),
```

```
    client = redis.createClient();

client.set('mykey', 'Hello World');
client.get('mykey', function(err,res){
    console.log(res);
});
```

Lire Commencer avec Redis en ligne: https://riptutorial.com/fr/redis/topic/1724/commencer-avec-redis

Chapitre 2: Comment se connecter à Redis en Java en utilisant Jedis

Introduction

Il existe plus de dix bibliothèques client différentes à utiliser avec Redis en Java. Jedis est l'un des clients les plus populaires.

Remarques

Informations complémentaires:

- Clients Java Redis
- Jedis Github Repository
- Jedis Documentation / Wiki

Examples

Obtenir Jedis

La bibliothèque Jedis est généralement ajoutée au projet Java à l'aide d'un système de gestion des dépendances intégré à l'environnement de construction du projet. Maven et Gradle sont deux systèmes Java populaires.

Utiliser Gradle

Pour ajouter la bibliothèque Jedis à un projet Gradle, vous devez configurer un référentiel et ajouter une dépendance. L'extrait suivant montre comment ajouter la version 2.9.0 de la bibliothèque Jedis à un projet Gradle.

```
repositories {
    mavenCentral()
}

dependencies {
    compile 'redis.clients:jedis:2.9.0'
}
```

Utiliser Maven

Pour ajouter Jedis à un projet Maven, vous devez ajouter une dépendance à votre liste de dépendances et fournir les coordonnées de la bibliothèque. L'extrait suivant serait ajouté à votre fichier pom.xml:

```
<dependencies>
```

```
    <dependency>
        <groupId>redis.clients</groupId>
        <artifactId>jedis</artifactId>
        <version>2.9.0</version>
    </dependency>
</dependencies>
```

Connexion à Redis

Utiliser un pool

La plupart du code voudra se connecter à Redis en utilisant un pool d'objets de connexion partagés. La connexion à Redis via un pool implique deux blocs de code différents. Au moment de l'initialisation, votre application doit créer le pool de connexions:

```
    JedisPoolConfig poolCfg = new JedisPoolConfig();
    poolCfg.setMaxTotal(3);

    pool = new JedisPool(poolCfg, hostname, port, 500, password, false);
```

`JedisPoolConfig` fournit des options pour régler le pool.

Au fur et à mesure que votre application traite sa charge de travail, vous devez vous connecter au pool partagé à l'aide du code suivant:

```
    try (Jedis jedis = pool.getResource()) {

        ...
    }
```

La meilleure pratique consiste à extraire l'objet de connexion `Jedis` du pool dans un bloc try-with-resources.

Sans piscine

Dans certains cas, tels qu'une application simple ou un test d'intégration, vous ne souhaitez peut-être pas gérer des pools partagés et créer directement l'objet de connexion `Jedis` . Cela peut être accompli avec le code suivant:

```
try (Jedis jedis = new Jedis(hostname, port)) {
    jedis.connect();
    jedis.auth(password);
    jedis.select(db);

    . . .
}
```

Encore une fois, la meilleure pratique consiste à créer l'objet client Jedis dans un bloc try-with-resources.

Exécution de commandes Get / Set de base

Une fois que vous avez établi une connexion à Redis, vous pouvez obtenir et définir des valeurs à l'aide de l'objet de connexion `Jedis` :

Obtenir

```
String value = jedis.get(myKey);
```

Ensemble

```
jedis.put(myKey, "some value");
```

Exécution des commandes

Pour exécuter une commande Redis à l'aide de Jedis, vous effectuez des appels de méthode contre l'objet `Jedis` créé à partir du pool. Jedis expose les commandes Redis comme des appels de méthode, par exemple:

```
- String get(String key)
- Long geoadd(String key, double longitude, double latitude, String member)
- List<String> hmget(String key, String... fields)
- Long hsetnx(String key, String field, String value)
```

Si vous vouliez définir la valeur d'une clé String dans Redis, vous utiliseriez un bloc de code similaire à:

```
try (Jedis jedis = pool.getResource()) {

   String myKey = "users:20";
   String myValue = "active";

   jedis.set(myKey, myValue);
}
```

Lire Comment se connecter à Redis en Java en utilisant Jedis en ligne:
https://riptutorial.com/fr/redis/topic/9712/comment-se-connecter-a-redis-en-java-en-utilisant-jedis

Chapitre 3: Connexion à redis en utilisant Python

Introduction

La connexion à Redis en Python nécessite l'utilisation d'une bibliothèque client. Il existe de nombreuses bibliothèques client pour Python, mais **redis-py** est l'un des clients les plus populaires.

Une fois la bibliothèque client installée, vous pouvez accéder à Redis dans votre application en important le module approprié, en établissant une connexion, puis en exécutant une commande.

Remarques

Pour vous connecter sur redis avec python, vous devez installer un **client** . Vous pouvez installer avec pip en utilisant:

```
pip install redis
```

cela va installer **redis-py**

Vous pouvez éventuellement installer **hiredis-py** qui délègue l'analyse des messages de protocole au client C hiredis. Cela peut apporter une amélioration significative des performances dans de nombreuses situations. Vous pouvez installer hiredis avec pip en exécutant:

```
pip install hiredis
```

Examples

Ajouter un élément à la liste

```
import redis

r = redis.StrictRedis(host='localhost', port=6379, db=0)

r.lpush('myqueue','myelement')
```

Ajout de champs à un hachage

Il y a deux fonctions principales dans Redis (HSET et HMSET) pour ajouter des champs à une clé de hachage. Les deux fonctions sont disponibles en redis-py.

Utiliser HSET:

```
import redis

r = redis.StrictRedis(host='myserver', port=6379, db=0)
r.hset('my_key', 'field0', 'value0')
```

En utilisant HMSET:

```
import redis

r = redis.StrictRedis(host='myserver', port=6379, db=0)
r.hmset('my_key', {'field0': 'value0', 'field1':'value1', 'field2':'value2'}
```

Configuration d'une connexion à Redis

Le client **redis-py** fournit deux classes `StrictRedis` et `Redis` pour établir une connexion de base à une base de données Redis. La classe `Redis` est fournie pour la rétrocompatibilité et les nouveaux projets doivent utiliser la classe `StrictRedis` .

L'une des méthodes recommandées pour établir une connexion consiste à définir les paramètres de connexion dans un dictionnaire et à transmettre le dictionnaire au constructeur `StrictRedis` à l'aide de la syntaxe `**` .

```
conn_params = {
    "host": "myredis.somedomain.com",
    "port": 6379,
    "password": "sekret",
    "db": 0
}

r = redis.StrictRedis(**config)
```

Créer une transaction

Vous pouvez établir une transaction en appelant la méthode de `pipeline` sur `StrictRedis` . Les commandes redis exécutées sur la transaction sont effectuées dans un seul bloc.

```
# defaults to transaction=True
tx = r.pipeline()
tx.hincrbyfloat(debit_account_key, 'balance', -amount)
tx.hincrbyfloat(credit_account_key, 'balance', amount)
tx.execute()
```

Exécuter des commandes directement

Redis-py fournit la méthode `execute_command` pour appeler directement les opérations Redis. Cette fonctionnalité peut être utilisée pour accéder à tous les modules qui peuvent ne pas avoir une interface prise en charge dans le client redis-py. Par exemple, vous pouvez utiliser la commande `execute_command` pour répertorier tous les modules chargés sur un serveur Redis:

```
r.execute_command('MODULE', 'LIST')
```

Lire Connexion à redis en utilisant Python en ligne:
https://riptutorial.com/fr/redis/topic/9103/connexion-a-redis-en-utilisant-python

Chapitre 4: Ensembles triés

Introduction

Le type de données Sorted Set dans Redis est une version ordonnée du type de données Set. Un ensemble trié Redis consiste en une collection de membres uniques. Chaque membre de l'ensemble trié peut être considéré comme une paire composée du membre et d'un score. Le score est utilisé pour classer les membres dans l'ensemble par ordre croissant.

Syntaxe

- Touche ZADD [NX | XX] [CH] [INCR] membre du score [score du membre ...]
- Clé ZCARD
- Touche ZCOUNT min max
- Touche ZLEXCOUNT min max

Remarques

La documentation officielle des kits triés est disponible sur le site Redis.io .

Les ensembles triés sont parfois appelés ensembles zsets. Si vous utilisez la commande TYPE sur une clé de jeu triée, la valeur zset sera renvoyée.

Examples

Ajout d'éléments à un ensemble trié

Redis fournit la commande ZADD pour ajouter des éléments à un ensemble trié. La forme de base de la commande ZADD consiste à spécifier l'ensemble, l'élément à ajouter et son score. Par exemple, si je voulais construire un ensemble ordonné de mon plat préféré (du moins au plus grand), je pourrais utiliser l'un des deux:

```
zadd favs 1 apple
zadd favs 2 pizza
zadd favs 3 chocolate
zadd favs 4 beer
```

Ou bien:

```
zadd favs 1 apple 2 pizza 3 chocolate 4 beer
```

La fonction ZADD fonctionne de manière très similaire à la fonction SADD non triée. Le résultat de la commande ZADD est le nombre d'éléments ajoutés. Donc, après avoir créé mon set comme ci-dessus, si j'ai essayé de refaire de la bière ZADD:

```
ZADD favs 4 beer
```

J'obtiendrais un résultat 0, si je décidais que j'aimais mieux le chocolat que la bière, je pourrais exécuter:

```
ZADD favs 3 beer 4 chocolate
```

pour mettre à jour mes préférences, mais j'obtiendrais quand même un résultat de 0 puisque la bière et le chocolat sont déjà dans le jeu.

Comptage d'articles dans un ensemble trié

Redis fournit trois commandes pour compter les éléments dans un jeu trié: ZCARD, ZCOUNT, ZLEXCOUNT.

La commande ZCARD est le test de base pour la cardinalité d'un ensemble. (Il est analogue à la commande SCARD pour les ensembles.). ZCARD renvoie le nombre de membres d'un ensemble. Exécuter le code suivant pour ajouter des éléments à un ensemble:

```
zadd favs 1 apple
zadd favs 2 pizza
zadd favs 3 chocolate
zadd favs 4 beer
```

ZCard en cours d'exécution:

```
zcard favs
```

renvoie une valeur de 4.

Les commandes ZCOUNT et ZLEXCOUNT vous permettent de compter un sous-ensemble des éléments d'un ensemble trié en fonction d'une plage de valeurs. ZCOUNT vous permet de compter des éléments dans une plage particulière de scores et ZLEXCOUNT vous permet de compter le nombre d'éléments dans une plage lexographique particulière.

En utilisant notre ensemble ci-dessus:

```
zcount favs 2 5
```

renverrait un 3, car il y a trois articles (pizza, chocolat, bière) qui ont des scores compris entre 2 et 5 inclus.

ZLEXCOUNT est conçu pour fonctionner avec des ensembles où chaque élément a le même score, forçant et ordonnant les noms d'élément. Si nous avons créé un ensemble comme:

```
zadd favs 1 apple
zadd favs 1 pizza
zadd favs 1 chocolate
zadd favs 1 beer
```

nous pourrions utiliser ZLEXCOUNT pour obtenir le nombre d'éléments dans une plage lexographique particulière (ceci est fait par une comparaison par octets en utilisant la fonction memcpy).

```
zlexcount favs [apple (chocolate
```

retournerait 2, car deux éléments (pomme, bière) entrent dans la gamme des pommes (inclus) et du chocolat (exclusif). Nous pourrions alternativement faire les deux bouts inclus:

```
zlexcount favs [apple [chocolate
```

et obtenez le résultat 3.

Lire Ensembles triés en ligne: https://riptutorial.com/fr/redis/topic/9111/ensembles-tries

Chapitre 5: Géo

Introduction

Redis fournit le type de données GEO pour travailler avec des données indexées géospatiales.

Syntaxe

- GEOADD key longitude latitude member [longitude latitude membre ...]
- GEODIST membre clé1 membre2 [unité]

Examples

GEOADD

La commande GEOADD permet à un utilisateur d'ajouter des informations géospatiales (nom de l'élément, longitude, latitude) à une clé particulière.

La commande GEOADD peut être utilisée pour ajouter un seul élément à une clé:

```
GEOADD meetup_cities -122.43 37.77 "San Francisco"
```

ou plusieurs éléments à une clé:

```
GEOADD meetup_cities -122.43 37.77 "San Francisco" -104.99 39.74 "Denver"
```

GEODIST

La commande GEODIST permet à un utilisateur de déterminer la distance entre deux membres d'un index géospatial tout en spécifiant les unités.

Pour trouver la distance entre deux villes de rencontre:

```
GEODIST meetup_cities "San Francisco" "Denver" mi
```

Lire Géo en ligne: https://riptutorial.com/fr/redis/topic/9091/geo

Chapitre 6: Installation et configuration

Examples

Installation de Redis

```
wget http://download.redis.io/redis-stable.tar.gz
tar xvzf redis-stable.tar.gz
cd redis-stable
make
```

Démarrer Redis

```
redis-server
```

Vérifiez si Redis fonctionne

```
redis-cli ping
```

Cela devrait retourner `PONG`

Accès à Redis Cli

En supposant que vous exécutez le serveur redis sur localhost, vous pouvez taper commande

```
redis-cli
```

Après que cette commande apparaisse, redis invite de ligne de commande

```
127.0.0.1:6379>
```

Redis types de données

Voici la liste de toutes les structures de données prises en charge par Redis:

- **Chaînes binaires sécurisées**
- **Listes** : collections d'éléments de chaînes triés en fonction de l'ordre d'insertion.
- **Ensembles** : collections d'éléments de chaîne uniques non triés.
- **Ensembles triés** : similaires aux ensembles mais où chaque élément de chaîne est associé à une valeur numérique flottante, appelée score.
- **Hash** : sont des cartes composées de champs associés à des valeurs.
- **HyperLogLogs** : il s'agit d'une structure de données probabiliste utilisée pour estimer la cardinalité d'un ensemble.

Basé sur la documentation officielle de ***redis.io***

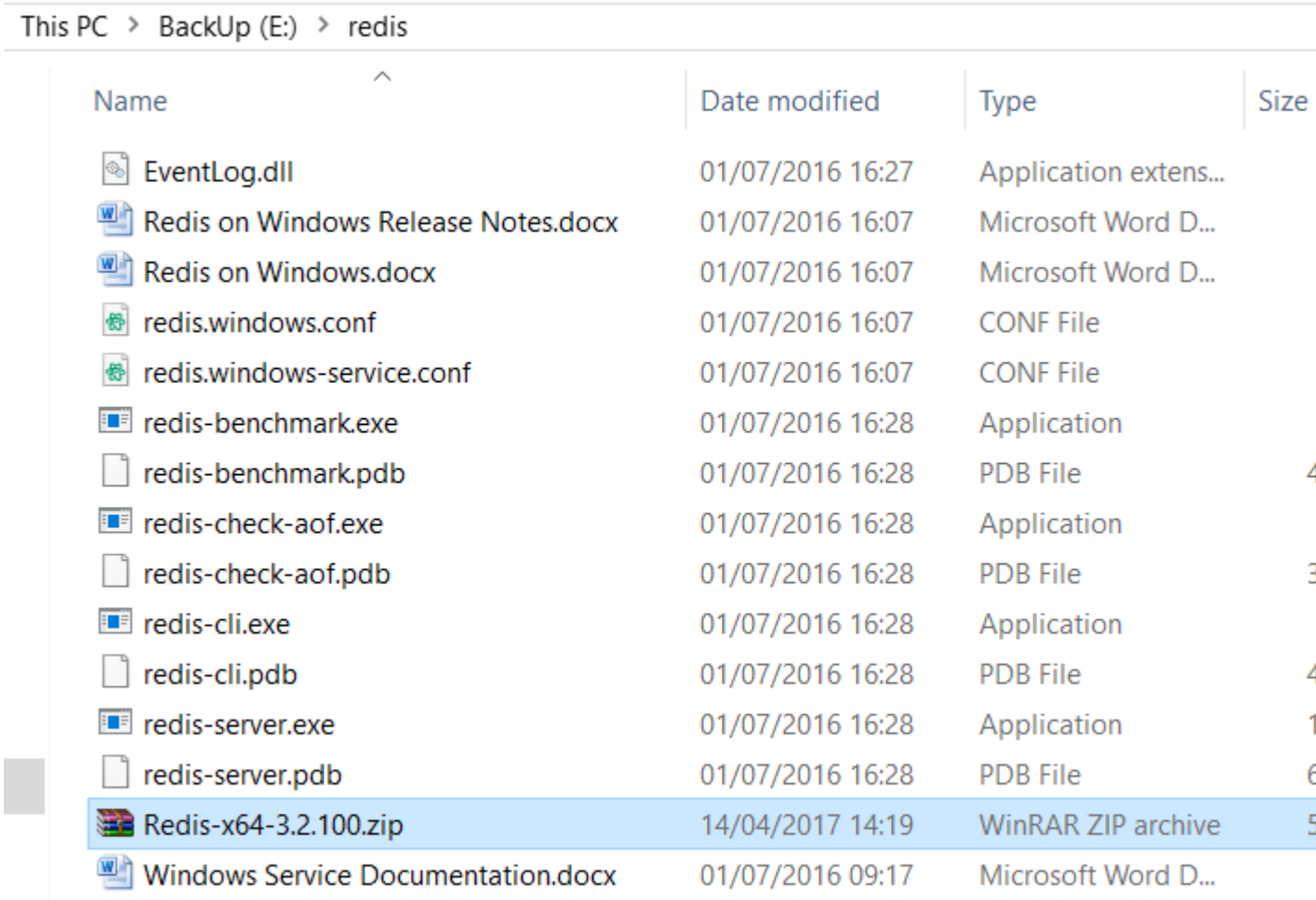
Installation et exécution de Redis Server sous Windows

Remarque: Le projet Redis ne prend pas officiellement en charge Windows.

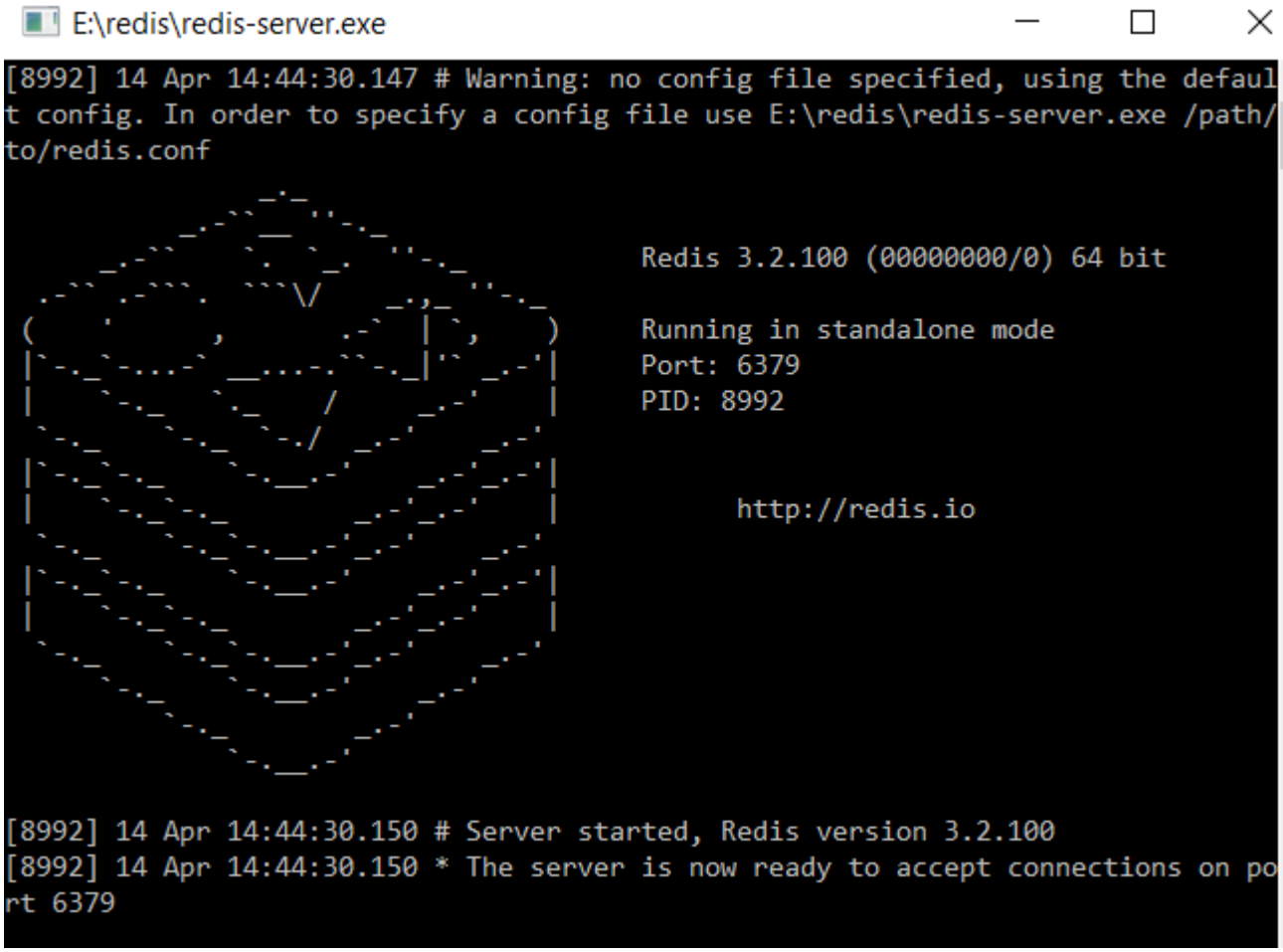
Cependant, le groupe **Microsoft Open Tech** développe et maintient ce port Windows ciblant Win64. **Redis.io/download officiel**

Vous pouvez choisir de télécharger différentes versions ou la dernière version de Redis github.com/MSOpenTech/redis/releases

1. **Téléchargez** soit le fichier .msi ou .zip, ce tutoriel vous permettra de télécharger le dernier fichier zip
   Redis-x64-3.2.100.zip .
2. **Extrayez le fichier zip** dans le répertoire préparé.

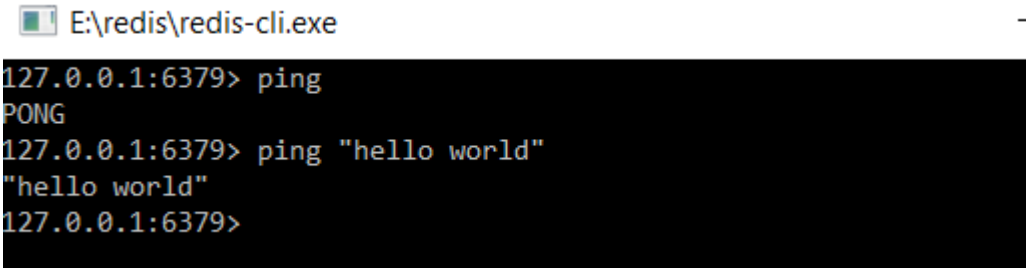


3. **Exécutez redis-server.exe** , vous pouvez soit exécuter directement redis-server.exe en cliquant ou en exécutant via l'invite de commandes.



4. **Exécutez redis-cli.exe** après avoir exécuté avec succès le serveur redis. Vous pouvez y accéder et tester les commandes en exécutant redis-cli.exe

La commande **PING** est utilisée pour tester si une connexion est toujours active.



Vous pouvez maintenant commencer à utiliser Redis, veuillez vous référer à plus de commandes dans les documentations officielles

Lire Installation et configuration en ligne: https://riptutorial.com/fr/redis/topic/2898/installation-et-configuration

Chapitre 7: Lua Scripting

Introduction

Redis fournit quelques mécanismes pour étendre les fonctionnalités de la base de données. Un mécanisme consiste à utiliser des scripts LUA côté serveur qui peuvent être exécutés pour manipuler des données. Les scripts Lua peuvent être utiles pour effectuer des opérations coûteuses ou implémenter des opérations atomiques nécessitant une logique.

Examples

Commandes pour les scripts

Redis propose sept opérations différentes pour travailler avec des scripts:

- Opérations d'évaluation (EVAL, EVALSHA)
- Opérations SCRIPT (DEBUG, EXISTS, FLUSH, KILL, LOAD)

La commande EVAL évalue un script fourni sous la forme d'un argument de chaîne sur le serveur. Les scripts peuvent accéder aux clés Redis spécifiées nommées comme arguments dans la commande et aux paramètres de chaîne supplémentaires que l'utilisateur souhaite transmettre au script.

Par exemple, la commande:

```
EVAL "return {KEYS[1],KEYS[2],ARGV[1],ARGV[2]}" 2 key1 key2 first second
```

provoque l'exécution d'un script Lua défini par l'utilisateur qui renvoie simplement les valeurs fournies. L'appel est impliqué avec 2 clés Redis (clé1 et clé2) et deux paramètres.

Une autre façon d'exécuter un script Lua consiste à le charger d'abord dans la base de données, puis à l'exécuter en utilisant un hachage SHA du script .:

```
> script load "return {KEYS[1],KEYS[2],ARGV[1],ARGV[2]}"
"a42059b356c875f0717db19a51f6aaca9ae659ea"
> evalsha "a42059b356c875f0717db19a51f6aaca9ae659ea" 2 key1 key2 foo bar
1) "key1"
2) "key2"
3) "foo"
4) "bar"
```

La commande de chargement de script charge le script et le stocke dans la base de données. Une signature sha du script est renvoyée pour pouvoir être référencée par les futurs appels. La fonction EVALSHA prend le sha et exécute le script correspondant à partir de la base de données.

Lire Lua Scripting en ligne: https://riptutorial.com/fr/redis/topic/9112/lua-scripting

Chapitre 8: Pub / Sub

Introduction

Redis fournit une implémentation du modèle de messagerie Publish / Subscribe (Pub / Sub). Au lieu d'envoyer des messages à des destinataires spécifiques, les éditeurs envoient des messages aux destinataires intéressés via un mécanisme d'indirection. Les récepteurs spécifient l'intérêt pour des messages particuliers. Dans Redis, cette fonctionnalité est accessible via les commandes PUBLISH et SUBSCRIBE sur les canaux.

Syntaxe

- SUBSCRIBE channel [canal ...]
- UNSUBSCRIBE [channel [channel ...]]
- Message de canal de publication
- Modèle PSUBSCRIBE [modèle ...]
- PUNSUBSCRIBE [modèle [modèle ...]]

Remarques

Pour gérer la publication / sous dans redis, vous devez avoir **un client pour vous abonner et un client différent pour la publication** . Les deux ne peuvent pas être gérés par le même client. Bien que toutes les autres commandes puissent toujours être traitées avec le même client.

Examples

Publiez et abonnez-vous avec redis

Redis a publié / abonné pour envoyer des messages. Ceci est géré en vous abonnant à un canal et en publiant sur le canal. Oui, les abonnés seront abonnés à une ou plusieurs chaînes. L'éditeur n'a pas besoin de savoir qui sont tous des abonnés. Au lieu de cela, l'éditeur publiera sur un canal spécifique. Tous les abonnés abonnés à cette chaîne recevront le message. Ce découplage des éditeurs et des abonnés peut permettre une plus grande évolutivité et une topologie de réseau plus dynamique.

Exemple: L' utilisateur s'abonne à 2 chaînes, disons foo & boo

```
SUBSCRIBE foo boo
```

Dans la console de redis-client1:

```
127.0.0.1:6379> SUBSCRIBE foo boo
Reading messages... (press Ctrl-C to quit)
1) "subscribe"
```

```
2) "foo"
3) (integer) 1
1) "subscribe"
2) "boo"
3) (integer) 2
```

Il va commencer à écouter le message. Lors de la publication, vous obtiendrez des données pour le canal correspondant.

Par exemple: Lorsque vous voulez envoyer un message à tous les abonnés connectés avec boo, vous devez publier sur ce canal.

```
PUBLISH boo "Hello Boo"
```

Dans la console de redis-client1:

```
1) "message"
2) "boo" //channel name
3) "Hello Boo" //Actual data
```

Pour vous désabonner du canal à tout moment, utilisez

```
UNSUBSCRIBE // to unsubscribe from all channels
UNSUBSCRIBE foo // to unsubscribe from specific channel
```

Pouvez-vous vous abonner en fonction du modèle aussi. Lorsque le nom du canal n'est pas sûr / souhaitez vous abonner en fonction du modèle, utilisez **PSUBSCRIBE** .

De même pour se désabonner en fonction de l'utilisation du modèle **PUNSUBSCRIBE**

Lire Pub / Sub en ligne: https://riptutorial.com/fr/redis/topic/5071/pub---sub

Chapitre 9: Redis Keys

Introduction

L'espace de clés Redis peut être considéré comme une table de hachage ou des clés de mappage de dictionnaire pour les structures de données de la base de données.

Redis fournit un large éventail de commandes qui fonctionnent avec des clés pour gérer l'espace de clés, y compris la possibilité de supprimer des clés, d'inspecter les métadonnées des clés, de rechercher des clés et de modifier certaines propriétés des clés.

Syntaxe

- Motif de touches
- Clé PERSIST
- EXPIRE clé secondes
- Horodatage de la clé EXPIREAT
- Clé TTL
- Clé millisecondes PEXPIRE
- Clé PEXPIREAT millisecondes-horodatage
- Clé PTTL
- Clé UNLINK [clé ...]
- Touche DEL [touche ...]
- Curseur SCAN [motif MATCH] [compte COUNT]

Remarques

Pour *les caractères valides dans les clés Redis* , le manuel explique cela complètement :

> Les clés Redis sont sécurisées en binaire, cela signifie que vous pouvez utiliser n'importe quelle séquence binaire en tant que clé, d'une chaîne telle que "foo" au contenu d'un fichier JPEG. La chaîne vide est également une clé valide.
>
> Quelques autres règles concernant les clés:
>
> Les clés très longues ne sont pas une bonne idée, par exemple une clé de 1024 octets est une mauvaise idée non seulement du point de vue de la mémoire, mais aussi parce que la recherche de la clé dans le jeu de données peut nécessiter plusieurs comparaisons de clés coûteuses. Même lorsque la tâche à accomplir consiste à faire coïncider l'existence d'une valeur élevée, le recours au hachage (par exemple avec SHA1) est une meilleure idée, en particulier du point de vue de la mémoire et de la bande passante.
>
> Les clés très courtes ne sont souvent pas une bonne idée. Il est inutile d'écrire "u1000flw" comme clé si vous pouvez écrire "user: 1000: followers". Ce dernier est

plus lisible et l'espace ajouté est mineur par rapport à l'espace utilisé par l'objet-clé lui-même et l'objet de valeur. Bien que les touches courtes consomment évidemment un peu moins de mémoire, votre travail consiste à trouver le bon équilibre.

Essayez de vous en tenir à un schéma. Par exemple, "object-type: id" est une bonne idée, comme dans "user: 1000". Les points ou les tirets sont souvent utilisés pour les champs à plusieurs mots, comme dans "comment: 1234: reply.to" ou "comment: 1234: reply-to".

La taille de clé maximale autorisée est de 512 Mo.

Soyez prudent lorsque vous utilisez la commande KEYS sur un système de production, cela peut entraîner de sérieux problèmes de performances. Si vous devez effectuer une recherche sur l'espace de clés, les commandes SCAN constituent une meilleure alternative.

Examples

Touches valides

Les clés Redis sont sécurisées par des binaires, de sorte que tout peut littéralement être utilisé comme clé. Les seules limites sont qu'elles doivent être inférieures à 512 Mo.

Exemples de clés valides:

```
7
++++
`~!@#$%^&*()-_=+
user:10134
search/9947372/?query=this%20is%20a%28test%29%20query
<div id="div64">

Any other string less than 512MB in size.
The raw binary content of an image or other binary file.
An entire multi-line text document.
An entire SQL query.
Any integer, hexadecimal, octal, or binary value.
Anything else you can think of less than 512MB in size.
```

Clés Redis non valides:

```
Anything larger than 512MB.
```

Schémas de nommage

Pour plus de clarté et de facilité de maintenance, il est souvent recommandé de développer un système ou un schéma permettant de nommer vos clés Redis. Voici quelques exemples de systèmes courants et maintenables pour nommer vos clés:

```
user:10134
```

```
user:10134:favorites
user:10134:friends
user:10134:friends-of-friends

user:10134
user:10134/favorites
user:10134/friends
user:10134/friends.of.friends

user/10134
user/10134/favorites
user/10134/friends
user/10134/friends of friends
```

Notez que, bien qu'autorisées, les touches plus grandes utilisent plus de mémoire et ralentissent les temps de recherche. L'utilisation d'une clé de 500 Mo n'est donc peut-être pas une bonne idée de performance. Une meilleure idée pourrait être d'utiliser un hachage SHA-1, SHA-256 ou MD5 d'un objet binaire volumineux en tant que clé à la place:

```
image/9517bb726d33efdc503a43582e6ea2eea309482b
image52e9df0577fca2ce022d4e8c86b1eccb070d37bef09dec36df2fabbfa7711f5c
```

Liste de toutes les clés

Vous pouvez répertorier toutes les clés d'une base de données Redis en exécutant les commandes suivantes à partir de redis-cli:

```
KEYS *
```

Le paramètre de KEYS est une expression de correspondance de modèle de style glob. Voici des exemples de motifs supposés:

```
h?llo matches hello, hallo and hxllo
h*llo matches hllo and heeeello
h[ae]llo matches hello and hallo, but not hillo
h[^e]llo matches hallo, hbllo, ... but not hello
h[a-b]llo matches hallo and hbllo
```

L'utilisation de la commande KEYS * peut avoir des effets néfastes sur les performances. Elle n'est donc pas recommandée contre les instances de production. Utilisez l'opération SCAN pour rechercher des clés dans le code de production.

TTL et expiration des clés

Les valeurs d'expiration d'une clé peuvent être gérées par un utilisateur en dehors des commandes de mise à jour. Redis permet à un utilisateur de déterminer l’heure actuelle de vie (TTL) d’une clé à l’aide de la commande TTL:

```
TTL key
```

Cette commande retournera la durée de vie d'une clé en secondes ou renverra les valeurs spéciales -1 ou -2. Un -1 indique que la clé est persistante (n'expire pas) et un -2 indique que la clé n'existe pas.

Une clé qui expire peut être rendue persistante à l'aide de la commande PERSIST:

```
PERSIST KEY
```

et une clé persistante peut être faite pour expirer en utilisant la commande EXPIRE:

```
EXPIRE KEY seconds
```

Expire peut également être utilisé pour modifier la durée de vie d'une clé existante. Vous pouvez également utiliser la commande EXPIREAT avec un horodatage UNIX pour définir une heure d'expiration.

Il existe des versions millisecondes des commandes TTL, EXPIRE et EXPIREAT qui sont préfixées par un P.

Suppression de clés

Redis fournit deux fonctions pour supprimer des clés de la base de données: del et unlink.

La fonction del supprime une ou plusieurs clés de la base de données. La commande del provoque Redis pour récupérer immédiatement la mémoire pour la clé supprimée sur le thread d'exécution en cours. Le temps d'exécution pour del est proportionnel au nombre d'éléments individuels supprimés de toutes les clés.

La fonction unlink agit comme la commande del, elle supprime une ou plusieurs clés de la base de données. Cependant, contrairement à la commande del, toute mémoire utilisée par ces clés est récupérée de manière asynchrone sur un autre thread.

Numérisation du Redis Keyspace

Redis fournit la commande SCAN pour parcourir les clés de la base de données correspondant à un modèle particulier. Redis prend en charge la correspondance des modèles de style glob dans la commande SCAN.

La commande SCAN fournit un itérateur à base de curseur sur l'espace de clés Redis. La séquence d'appel itérative de SCAN commence par un appel de l'utilisateur avec l'argument curseur positionné sur 0. Le résultat de cet appel est un lot d'articles et un curseur mis à jour qui est fourni au prochain appel à SCAN. Cette itération continue jusqu'à ce que Redis renvoie un curseur 0.

La fonction Python suivante illustre l'utilisation de base de SCAN:

```
def scan_keys(r, pattern):
    "Returns a list of all the keys matching a given pattern"
```

```
    result = []
    cur, keys  = r.scan(cursor=0, match=pattern, count=2)
    result.extend(keys)
    while cur != 0:
        cur, keys = r.scan(cursor=cur, match=pattern, count=2)
        result.extend(keys)

    return result
```

La commande SCAN est la méthode recommandée pour rechercher des clés dans la base de données et est recommandée par rapport à la commande `KEYS *` .

Lire Redis Keys en ligne: https://riptutorial.com/fr/redis/topic/3916/redis-keys

Chapitre 10: Redis List Type de données

Introduction

Le type de données List dans Redis est une collection ordonnée d'éléments référencés par une clé Redis. Redis vous permet d'accéder et de modifier une liste par index ou par opérations push / pop. Dans Redis, les deux extrémités d'une liste sont appelées gauche et droite. La gauche correspond au premier élément ou à la tête d'une liste et la droite correspond au dernier élément ou à la queue d'une liste.

Syntaxe

- Valeur clé LPUSH [valeur ...]
- Valeur clé RPUSH [valeur ...]
- Clé LPOP
- Clé RPOP
- Clé LLEN

Remarques

Vous trouverez plus de détails sur le type de données List et toutes les commandes pouvant être utilisées conjointement avec celles-ci dans la documentation officielle Redis sur Redis.io .

Examples

Ajout d'éléments à une liste

Redis vous permet d'ajouter des éléments à droite ou à gauche d'une liste.

Si je travaillais avec une liste, my_list et que je voulais ajouter 3 à la liste, je pourrais le faire en utilisant la commande Redis LPUSH:

```
LPUSH my_list 3
```

Si je voulais ajouter 3 à my_list, j'utiliserais plutôt la commande RPUSH:

```
RPUSH my_list 3
```

Les commandes LPUSH et RPUSH créeront automatiquement une nouvelle liste pour vous si la clé fournie n'existe pas. Deux commandes alternatives LPUSHX et RPUSHX peuvent être utilisées pour fonctionner uniquement sur la clé de liste si elle existe déjà.

Obtenir des éléments d'une liste

Redis fournit les commandes LPOP et RPOP en contrepartie des commandes LPUSH et RPUSH pour l'extraction d'éléments de données.

Si je travaillais avec une liste my_list contenant plusieurs éléments de données, je peux obtenir le premier élément de la liste à l'aide de la commande LPOP:

```
LPOP my_list
```

Le résultat de cette commande renvoie la valeur du premier élément de la liste et le supprime de my_list. Par exemple, si j'avais la liste [1, 3, 2, 4] et que je lui appliquais le LPOP, j'aurais la liste [3, 2, 4] en mémoire par la suite.

De même, je peux supprimer de la fin de la liste en utilisant RPOP:

```
RPOP my_list
```

renverrait la valeur du dernier élément de la liste, puis la supprimerait de my_list. En utilisant notre exemple, [1, 2, 3, 4] après avoir appelé RPOP sur cette liste, la liste en mémoire serait [1, 2, 3].

Taille d'une liste

La taille d'une liste Redis peut être déterminée à l'aide de la commande LLEN. Si j'ai une liste de quatre éléments stockée dans la clé my_list, je peux obtenir la taille en utilisant:

```
LLEN my_list
```

qui reviendra 4.

Si un utilisateur spécifie une clé qui n'existe pas pour LLEN, il retournera un zéro, mais si une clé est utilisée pour désigner un élément d'un type de données différent, une erreur sera renvoyée.

Lire Redis List Type de données en ligne: https://riptutorial.com/fr/redis/topic/9107/redis-list-type-de-donnees

Chapitre 11: Redis Set Datatype

Introduction

Redis prend en charge un type de données défini analogue aux ensembles mathématiques pour la modélisation des données dans la base de données. Les ensembles sont un type de données composé constitué d'un groupe de membres uniques non ordonnés. Les ensembles prennent en charge l'ajout et la suppression de membres, les opérations de taille ainsi que les opérations de combinaison qui prennent deux ensembles et génèrent un troisième ensemble. Les ensembles dans Redis sont similaires aux ensembles dans la plupart des langages de programmation.

Syntaxe

- Membre clé de SADD [membre ...]
- Membre clé de SISMEMBER
- Touche SCARD
- Membre clé de SADD [membre ...]

Remarques

La documentation complète sur le type de données Redis est disponible sur Redis.io .

Examples

Taille d'un ensemble

La taille d'un ensemble peut être déterminée à l'aide de la commande SCARD. SCARD renvoie la cardinalité d'un ensemble ou le nombre de membres dans l'ensemble. Par exemple, si un ensemble Redis définissait my_set dans la base de données qui ressemblait à (Apple, Orange, Banana), je pourrais obtenir la taille en utilisant le code suivant:

```
SCARD my_set
```

Dans le cas de mon exemple défini, cela renverrait 3. Si l'utilisateur exécute une commande SCARD sur une clé qui n'existe pas, Redis retournera 0.

Ajout d'éléments à un ensemble

La commande de base Redis pour ajouter un élément à un ensemble est SADD. Il prend une clé et un ou plusieurs membres et les ajoute à l'ensemble stocké à la clé donnée.

Par exemple, disons que je voulais créer un ensemble avec les éléments pomme, poire et banane. Je pourrais exécuter l'un des suivants:

```
SADD fruit apple
SADD fruit pear
SADD fruit banana
```

ou

```
SADD fruit apple pear banana
```

Après l'exécution, j'aurai le fruit avec 3 éléments.

Si vous essayez d'ajouter un élément qui se trouve déjà dans l'ensemble, cela n'aura aucun effet. Après avoir configuré mon set de fruits en utilisant le code ci-dessus, si j'essaie d'ajouter à nouveau Apple:

```
SADD fruit apple
```

Redis tentera d'ajouter de la pomme à la série de fruits, mais comme elle est déjà dans le jeu, rien ne changera.

Le résultat de la commande SADD est toujours le nombre d'éléments ajoutés par Redis à un ensemble. Donc, en essayant de rajouter Apple, le résultat sera 0.

Les éléments de membre dans Redis sont sensibles à la casse, donc Apple et Apple sont traités comme deux éléments distincts.

Test d'adhésion

Redis fournit la commande SISMEMBER pour tester si un élément particulier est déjà membre d'un ensemble. En utilisant la commande SISMEMBER, je peux tester et voir si Apple est déjà membre de mon set de fruits.

Si je construis mon jeu de fruits à partir de l'exemple précédent, je peux vérifier et voir s'il contient Apple en utilisant le test suivant:

```
SISMEMBER fruit apple
```

SISMEMBER renverra un 1 puisque l'élément est déjà là.

Si j'ai essayé de voir si le chien fait partie de mon groupe de fruits:

```
SISMEMBER fruit dog
```

Redis renverra un 0 car le chien n'est pas dans la série de fruits.

Si un utilisateur tente d'utiliser la commande SISMEMBER avec une clé qui n'existe pas, Redis renvoie un 0 indiquant l'absence d'appartenance, mais si vous utilisez SISMEMBER avec une clé contenant déjà un type de données non défini, Redis renvoie une erreur.

Lire Redis Set Datatype en ligne: https://riptutorial.com/fr/redis/topic/9109/redis-set-datatype

Chapitre 12: Sauvegarde

Introduction

La sauvegarde d'une instance Redis distante peut être réalisée avec la réplication. Ceci est utile si vous souhaitez prendre un instantané d'un ensemble de données avant de mettre à niveau, de supprimer ou de modifier une base de données Redis.

Examples

Sauvegarde d'une instance Redis distante sur une instance locale

Sur la machine sur laquelle vous souhaitez effectuer la sauvegarde, accédez à l'interface de ligne de commande Redis:

```
redis-cli
```

Mot de passe?

Si votre base de données Redis (celle que vous souhaitez répliquer) possède un mot de passe:

```
config set masterauth <password>
```

Démarrer la réplication

Exécutez les opérations suivantes pour commencer la réplication:

```
SLAVEOF <host> <port>
```

Pour vérifier que la réplication est en cours d'exécution:

```
INFO replication
```

Et vous devriez voir la sortie comme ceci:

```
# Replication
role:slave
master_host:some-host.compute-1.amazonaws.com
master_port:6519
master_link_status:up
master_last_io_seconds_ago:3
master_sync_in_progress:0
slave_repl_offset:35492914
slave_priority:100
slave_read_only:1
```

```
connected_slaves:0
master_repl_offset:0
repl_backlog_active:0
repl_backlog_size:1048576
repl_backlog_first_byte_offset:0
repl_backlog_histlen:0
```

Notez le `master_link_status` devrait être en `up` .

Vérification de la progression de la synchronisation

Lorsque la synchronisation est terminée, la `INFO replication` doit afficher:

```
master_sync_in_progress:0
```

Pour vérifier que le jeu de données a été synchronisé, vous pouvez comparer la taille de la base de données:

```
DBSIZE
```

Enregistrement d'un vidage de données sur disque

Pour enregistrer la base de données sur le disque de manière asynchrone:

```
BGSAVE
CONFIG GET dir
```

Ensuite, vous devriez trouver un fichier `dump.rdb` dans le répertoire répertorié par la commande config.

Arrêter la réplication

Vous pouvez arrêter la réplication avec:

```
SLAVEOF NO ONE
```

Référence: Guide de réplication Redis

Lire Sauvegarde en ligne: https://riptutorial.com/fr/redis/topic/9369/sauvegarde

Chapitre 13: Stockage de persistance Redis

Introduction

Redis prend en charge deux principaux modes de persistance: RDB et AOF. Le mode de persistance RDB prend un instantané de votre base de données à un moment donné. En mode RDB, Redis supprime un processus pour conserver la base de données sur le disque. AOF enregistre chaque opération exécutée sur le serveur dans un journal de relecture pouvant être traité au démarrage pour restaurer l'état de la base de données.

Examples

Désactiver tout stockage de persistance dans Redis

Il existe deux types de modes de stockage persistants dans Redis: AOF et RDB. Pour désactiver temporairement RDB, exécutez les commandes suivantes sur la ligne de commande Redis:

```
config set save ""
```

Pour désactiver temporairement AOF, exécutez les opérations suivantes à partir de la ligne de commande Redis:

```
config set appendonly no
```

Les modifications persisteront jusqu'à ce que le serveur soit redémarré, puis le serveur reviendra à tous les modes configurés dans le fichier redis.conf du serveur.

La commande `CONFIG REWRITE` peut être utilisée pour modifier le fichier redis.conf afin de refléter toute modification dynamique de la configuration.

Obtenir l'état de stockage de la persistance

Le code suivant obtiendra la configuration actuelle pour l'état de stockage persistant. Ces valeurs peuvent être modifiées dynamiquement, elles peuvent donc différer de la configuration de redis.conf:

```
# get
config get appendonly
config get save
```

Lire Stockage de persistance Redis en ligne: https://riptutorial.com/fr/redis/topic/7871/stockage-de-persistance-redis

Chapitre 14: Type de données Redis String

Introduction

Redis fournit un type de données de chaîne utilisé pour associer des données à une clé particulière. La chaîne Redis est le type de données le plus élémentaire disponible dans Redis et l'un des premiers types de données avec lesquels les utilisateurs apprennent à travailler.

Les chaînes sont souvent associées à des données texte, mais les chaînes Redis ressemblent davantage à des tampons qui peuvent être utilisés pour stocker un large éventail de données différentes. Les chaînes Redis peuvent être utilisées pour représenter des entiers, des nombres à virgule flottante, des bitmaps, du texte et des données binaires.

Syntaxe

- Valeur de la touche SET [secondes EX] [millisecondes PX] [NX | XX]
- Clé INCR
- Incrément de clé INCRBY
- Incrément de clé INCRBYFLOAT
- Clé DECR
- Décrément de clé DECRBY

Examples

Travailler avec String en tant que nombres entiers

Plusieurs commandes vous permettent de travailler avec des chaînes représentant des valeurs entières.

Un utilisateur peut définir la valeur entière d'une clé à l'aide de la commande:

```
SET intkey 2
```

La commande set créera la clé si nécessaire ou la mettra à jour si elle existe déjà.

La valeur d'une clé entière peut être mise à jour sur le serveur à l'aide des commandes INCR ou INCRBY. INCR augmentera la valeur d'une clé de 1 et INCRBY augmentera la valeur de la clé en fonction de la valeur de pas fournie.

```
INCR intkey
INCRBY intkey 2
```

Si la valeur de la clé spécifiée pour INCR ou INCRBY ne peut pas être exprimée sous la forme d'un entier, Redis renvoie une erreur. Si la clé n'existe pas, la clé sera créée et l'opération sera appliquée à la valeur par défaut de 0.

Les commandes DECR et DECRBY fonctionnent en sens inverse pour diminuer la valeur.

Travailler avec des chaînes en tant que nombres à virgule flottante

Redis vous permet d'utiliser le type de données String pour stocker des nombres à virgule flottante.

Un utilisateur peut définir la valeur flottante d'une clé à l'aide de la commande:

```
SET floatkey 2.0
```

La commande set créera la clé si nécessaire ou la mettra à jour si elle existe déjà.

La valeur de la clé peut être mise à jour sur le serveur à l'aide de la commande INCRBYFLOAT. INCRBYFLOAT augmentera la valeur d'une clé par la valeur d'incrément fournie.

```
INCRBYFLOAT floatkey 2.1
```

Si la valeur de la clé spécifiée à INCRBYFLOAT ne peut pas être exprimée en virgule flottante, Redis renverra une erreur. Si la clé n'existe pas, la clé sera créée et l'opération sera appliquée à la valeur par défaut de 0.0.

Les clés peuvent être décrémentées en transmettant un incrément négatif à la commande INCRBYFLOAT.

Lire Type de données Redis String en ligne: https://riptutorial.com/fr/redis/topic/9507/type-de-donnees-redis-string

Crédits

S. No	Chapitres	Contributeurs
1	Commencer avec Redis	Ahamed Mustafa M, Alexander V., Aminadav, Community, Florian Hämmerle, Itamar Haber, Prashant Barve, RLaaa, Sagar Ranglani, sirin
2	Comment se connecter à Redis en Java en utilisant Jedis	Tague Griffith
3	Connexion à redis en utilisant Python	Gianluca D'Ardia, Tague Griffith, ystark
4	Ensembles triés	Tague Griffith
5	Géo	Tague Griffith
6	Installation et configuration	Cristiana214, Gianluca D'Ardia, Sagar Ranglani
7	Lua Scripting	Tague Griffith
8	Pub / Sub	jerry, Tague Griffith
9	Redis Keys	Tague Griffith, Will
10	Redis List Type de données	Tague Griffith
11	Redis Set Datatype	JonyD, Tague Griffith
12	Sauvegarde	odlp
13	Stockage de persistance Redis	Aminadav, Tague Griffith
14	Type de données Redis String	Tague Griffith