EBook Gratis

APRENDIZAJE redis

Free unaffiliated eBook created from
Stack Overflow contributors.

#redis

Tabla de contenido

Acerca de

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.

The content is released under Creative Commons BY-SA, and the list of contributors to each chapter are provided in the credits section at the end of this book. Images may be copyright of their respective owners unless otherwise specified. All trademarks and registered trademarks are the property of their respective company owners.

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

Capítulo 1: Empezando con redis

Observaciones

Esta sección proporciona una descripción general de qué es Redis y por qué un desarrollador puede querer usarlo.

También debe mencionar cualquier tema importante dentro de Redis y vincular a los temas relacionados. Dado que la documentación de Redis es nueva, es posible que deba crear versiones iniciales de los temas relacionados.

Versiones

Versión	Fecha de lanzamiento
3.2.3	2016-08-02
3.2.2	2016-07-28

Examples

Visión general

Redis es una base de datos remota en memoria que ofrece alto rendimiento, replicación y un modelo de datos único para producir una plataforma para resolver problemas. Redis es un código abierto (licencia BSD), estructura de datos en memoria, utilizado como base de datos, caché y agente de mensajes. Se clasifica como un almacén de valor-clave NoSQL. Es compatible con estructuras de datos como cadenas, hashes, listas, conjuntos, conjuntos ordenados con consultas de rango, mapas de bits, hiperloglogs e índices geoespaciales con consultas de radio. Soportando cinco tipos diferentes de estructuras de datos,

1. STRING (Operar en toda la cadena, partes, enteros y flotantes)
2. LISTA (Empuje o saque elementos de ambos extremos)
3. SET (agregar, obtener, eliminar, verificar, intersectar, unir, diferencia, etc.)
4. HASH (tienda, fatch, eliminar en hash)
5. ZSET (igual a lo establecido pero en forma ordenada)
6. GEO (agregar, actualizar, eliminar latitud y longitud, obtener dentro de redius dado)

Redis tiene replicación incorporada, scripts Lua, desalojo de LRU, transacciones y diferentes niveles de persistencia en el disco (sync / async).

Antes de la versión 3, Redis funciona en modo maestro-esclavo y requería que Redis-Sentinel proporcionara alta disponibilidad. Solo el maestro acepta escrituras y sincroniza los datos con sus esclavos forzando.

A partir de la versión 3, Redis trabaja y recomienda el modo multi-master donde se incorporan las funciones de conmutación por error, fragmentación / partición y de retardo. Redis-Sentinel no se requiere de la versión 3. Para que el clúster redis funcione, se requieren un mínimo de 3 nodos / procesos maestros.

Las características adicionales son replicación, persistencia y fragmentación del lado del cliente. Redis se adapta a una amplia variedad de problemas que pueden asignarse naturalmente a lo que Redis ofrece, lo que le permite resolver sus problemas sin tener que realizar el trabajo conceptual requerido por otras bases de datos.

Interfaz de línea de comandos redis

`redis-cli` es el programa de interfaz de línea de comandos de Redis que permite enviar comandos a Redis y leer las respuestas enviadas por el servidor, directamente desde el terminal. El uso básico de la línea de comando está abajo:

Acceso a redis:

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

Acceso a redis con autenticación:

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

Seleccione la base de datos y muestre el tamaño de la base de datos (el número predeterminado de la base de datos es 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
```

Obtenga información y estadísticas sobre el servidor:

```
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
```

Saliendo de la redis-cli:

```
127.0.0.1:6379> exit
```

Redis "hola mundo"

Primero necesita instalar e iniciar su servidor Redis, verifique el enlace a continuación que puede ayudarlo a instalar redis en su servidor o máquina local.

Instalación y configuración

Ahora abra su línea de comandos y ejecute el comando `redis-cli` :

Para guardar el primer conjunto> SET 'keyname' luego 'valor'

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

Presiona Enter que deberías ver

```
OK
```

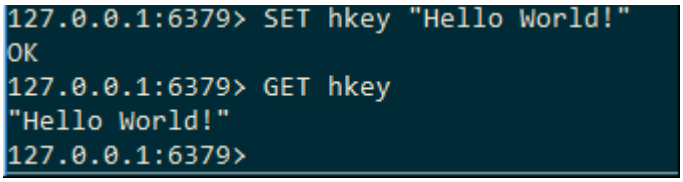
Luego ingrese:

```
GET hkey
```

debería ver:

```
"Hello World!"
```

Ejemplo de salida de pantalla:



Instala Redis utilizando Docker

Es simple comenzar a usar Redis usando la ventana acoplable:

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

Ahora tienes la instancia en ejecución en el puerto `6397`

Atención: se eliminarán todos los datos, cuando se detendrá Redis.

Para conectar el redis-cli, inicie otra ventana acoplable:

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

Ahora puedes jugar con tu redis docker.

Instalación de Redis en Windows, con ejemplo de Node.js

Redis tiene un puerto de Windows proporcionado por 'Microsoft Open Technologies'. Puede usar el instalador msi que se encuentra en: https://github.com/MSOpenTech/redis/releases

Una vez completada la instalación, puede ver que 'Redis' es un servicio de Windows (y su estado debería ser "Iniciado")

Para escribir un ejemplo de 'Hola mundo' que use Redis en Node.js (también en Windows) puede usar el siguiente módulo npm: https://www.npmjs.com/package/redis

Ejemplo de código:

```
var redis = require('redis'),
    client = redis.createClient();

client.set('mykey', 'Hello World');
```

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

Lea Empezando con redis en línea: https://riptutorial.com/es/redis/topic/1724/empezando-con-redis

Capítulo 2: Almacenamiento de persistencia Redis

Introducción

Redis admite dos modos principales de persistencia: RDB y AOF. El modo de persistencia RDB toma una instantánea de su base de datos en un momento determinado. En el modo RDB, Redis desactiva un proceso para conservar la base de datos en el disco. AOF registra cada operación ejecutada contra el servidor en un registro de reproducción que puede procesarse en el inicio para restaurar el estado de la base de datos.

Examples

Deshabilitar todo el almacenamiento de persistencia en Redis

Hay dos tipos de modos de almacenamiento persistente en Redis: AOF y RDB. Para deshabilitar temporalmente RDB, ejecute los siguientes comandos en la línea de comandos de Redis:

```
config set save ""
```

para deshabilitar temporalmente AOF, ejecute lo siguiente desde la línea de comandos de Redis:

```
config set appendonly no
```

Los cambios persistirán hasta que se reinicie el servidor, luego el servidor volverá a los modos configurados en el archivo redis.conf del servidor.

El comando `CONFIG REWRITE` se puede usar para modificar el archivo redis.conf para reflejar cualquier cambio dinámico en la configuración.

Obtener estado de almacenamiento persistente

El siguiente código obtendrá la configuración actual para el estado de almacenamiento persistente. Estos valores pueden modificarse dinámicamente, por lo que pueden diferir de la configuración en redis.conf:

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

Lea Almacenamiento de persistencia Redis en línea:
https://riptutorial.com/es/redis/topic/7871/almacenamiento-de-persistencia-redis

Capítulo 3: Apoyo

Introducción

La copia de seguridad de una instancia de Redis remota se puede lograr con la replicación. Esto es útil si desea tomar una instantánea de un conjunto de datos antes de actualizar, eliminar o cambiar una base de datos de Redis.

Examples

Copia de seguridad de una instancia de Redis remota en una instancia local

En la máquina donde desea realizar la copia de seguridad, vaya a la CLI de Redis:

```
redis-cli
```

¿Contraseña?

Si su Redis DB maestra (la que desea replicar) tiene una contraseña:

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

Iniciar la replicación

Ejecute lo siguiente para comenzar la replicación:

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

Para verificar la replicación se está ejecutando:

```
INFO replication
```

Y deberías ver una salida como esta:

```
# 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
```

Tenga en cuenta que el `master_link_status` debe estar `up` .

Comprobando el progreso de sincronización

Cuando se complete la sincronización, la `INFO replication` debería mostrar:

```
master_sync_in_progress:0
```

Para comprobar que el conjunto de datos se ha sincronizado, puede comparar el tamaño de la base de datos:

```
DBSIZE
```

Guardando un volcado de datos en el disco

Para guardar la base de datos en el disco de forma asíncrona:

```
BGSAVE
CONFIG GET dir
```

Luego debe encontrar un archivo `dump.rdb` en el directorio listado por el comando config.

Detener la replicación

Puedes detener la replicación con:

```
SLAVEOF NO ONE
```

Referencia: Guía de replicación de Redis.

Lea Apoyo en línea: https://riptutorial.com/es/redis/topic/9369/apoyo

Capítulo 4: Cómo conectarse a Redis en Java usando Jedis

Introducción

Hay más de diez bibliotecas de clientes diferentes para usar con Redis en Java. Uno de los clientes más populares es Jedis .

Observaciones

Más información:

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

Examples

Obteniendo Jedis

La biblioteca Jedis generalmente se agrega al proyecto Java utilizando un sistema de administración de dependencias integrado en el entorno de compilación del proyecto. Dos sistemas de construcción Java populares son Maven y Gradle.

Usando gradle

Para agregar la biblioteca Jedis a un proyecto de Gradle, necesitará configurar un repositorio y agregar una dependencia. El siguiente fragmento de código muestra cómo agregar la versión 2.9.0 de la biblioteca Jedis a un proyecto de Gradle.

```
repositories {
    mavenCentral()
}

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

Usando maven

Para agregar Jedis a un proyecto de Maven, debe agregar una dependencia a su lista de dependencias y proporcionar las coordenadas de la biblioteca. El siguiente fragmento de código se agregaría a su archivo pom.xml:

```
<dependencies>
```

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

Conectando a Redis

Usando una piscina

La mayoría de los códigos querrán conectarse a Redis utilizando un conjunto de objetos de conexión compartidos. Conectarse a Redis usando un grupo implica dos bloques de código diferentes. En el momento de la inicialización, su aplicación necesita crear el grupo de conexiones:

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

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

El `JedisPoolConfig` proporciona opciones para ajustar el grupo.

A medida que la aplicación procesa su carga de trabajo, deberá obtener una conexión del grupo compartido utilizando el siguiente código:

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

        ...
    }
```

La mejor práctica es obtener el objeto de conexión `Jedis` de la agrupación dentro de un bloque try-with-resources.

Sin piscinas

En algunos casos, como una aplicación simple o una prueba de integración, es posible que no desee tratar con grupos compartidos y, en su lugar, crear el objeto de conexión `Jedis` directamente. Eso se puede lograr con el siguiente código:

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

    . . .
}
```

Una vez más, la mejor práctica es crear el objeto cliente Jedis dentro de un bloque try-with-resources.

Ejecutando Comandos básicos de Get / Set

Una vez que haya establecido una conexión con Redis, puede obtener y establecer valores utilizando el objeto de conexión `Jedis` :

Obtener

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

Conjunto

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

Comandos de ejecución

Para ejecutar un comando Redis utilizando Jedis, realice llamadas de método contra el objeto `Jedis` que creó desde el grupo. Jedis expone los comandos de Redis como llamadas de método, algunos ejemplos son:

```
- 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 quisiera establecer el valor de una clave de cadena en Redis, usaría un bloque de código similar a:

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

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

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

Lea Cómo conectarse a Redis en Java usando Jedis en línea:
https://riptutorial.com/es/redis/topic/9712/como-conectarse-a-redis-en-java-usando-jedis

Capítulo 5: Conectando a redis usando Python

Introducción

Conectarse a Redis en Python requiere el uso de una biblioteca cliente. Existen muchas bibliotecas de clientes diferentes para Python, pero **redis-py** es uno de los clientes más populares en uso.

Una vez que instale su biblioteca cliente, puede acceder a Redis en su aplicación importando el módulo apropiado, estableciendo una conexión y luego ejecutando un comando.

Observaciones

Para conectarse en redis con python necesita instalar un **cliente** . Se puede instalar con pip usando:

```
pip install redis
```

esto instalará **redis-py**

Opcionalmente, es posible que desee instalar **hiredis-py** que delegue el análisis de los mensajes de protocolo al cliente de C hiredis. Esto puede proporcionar una mejora significativa del rendimiento en muchas situaciones. Puedes instalar hiredis con pip ejecutando:

```
pip install hiredis
```

Examples

Añadir elemento a la lista

```
import redis

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

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

Añadiendo campos a un hash

Hay dos funciones principales en Redis (HSET y HMSET) para agregar campos a una clave de hash. Ambas funciones están disponibles en redis-py.

Utilizando HSET:

```
import redis

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

Utilizando HMSET:

```
import redis

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

Configuración de una conexión a Redis

El cliente **redis-py** proporciona dos clases `StrictRedis` y `Redis` para establecer una conexión básica a una base de datos Redis. La clase `Redis` se proporciona para la compatibilidad con versiones anteriores y los nuevos proyectos deben usar la clase `StrictRedis` .

Una de las formas recomendadas para establecer una conexión, es definir los parámetros de conexión en un diccionario y pasar el diccionario al constructor `StrictRedis` usando la sintaxis `**` .

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

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

Creando una transacción

Puede establecer una transacción llamando al método de `pipeline` en `StrictRedis` . Los comandos redis ejecutados contra la transacción se realizan en un solo bloque.

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

Ejecutando Comandos Directamente

Redis-py proporciona el método `execute_command` para invocar directamente las operaciones de Redis. Esta funcionalidad se puede usar para acceder a cualquier módulo que no tenga una interfaz compatible en el cliente redis-py. Por ejemplo, puede usar el `execute_command` para enumerar todos los módulos cargados en un servidor Redis:

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

Lea Conectando a redis usando Python en línea:
https://riptutorial.com/es/redis/topic/9103/conectando-a-redis-usando-python

Capítulo 6: Conjuntos ordenados

Introducción

El tipo de datos del conjunto ordenado en Redis es una versión ordenada del tipo de datos del conjunto. Un conjunto clasificado de Redis consiste en una colección de miembros únicos. Cada miembro del conjunto clasificado puede considerarse como un par que consiste en el miembro y una puntuación. La puntuación se utiliza para ordenar los miembros dentro del conjunto en orden ascendente.

Sintaxis

- Clave de ZADD [NX | XX] [CH] [INCR] miembro de puntuación [miembro de puntuación ...]
- Tecla ZCARD
- Tecla ZCOUNT min max max
- Tecla ZLEXCOUNT min max max

Observaciones

La documentación oficial de los Conjuntos Clasificados se puede encontrar en el sitio Redis.io .

Los conjuntos ordenados a veces se denominan zsets. Si usa el comando TIPO en una clave de conjunto ordenada, se devolverá el valor zset.

Examples

Agregar elementos a un conjunto ordenado

Redis proporciona el comando ZADD para agregar elementos a un conjunto ordenado. La forma básica del comando ZADD es especificar el conjunto, el elemento a agregar y su puntuación. Por ejemplo, si quisiera construir un juego ordenado de mi comida favorita (de menos a la mayoría), podría usar cualquiera de los siguientes:

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

o alternativamente:

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

La función ZADD funciona de manera muy similar a la función de conjunto sin clasificar SADD. El resultado del comando ZADD es la cantidad de elementos que se agregaron. Así que después de

crear mi set como el de arriba, si intenté hacer una cerveza ZADD nuevamente:

```
ZADD favs 4 beer
```

Obtendría un resultado de 0, si decidiera que me gusta el chocolate más que la cerveza, podría ejecutar:

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

para actualizar mis preferencias, pero todavía obtendría un resultado de retorno de 0 ya que tanto la cerveza como el chocolate ya están en el set.

Contando elementos en un conjunto ordenado

Redis proporciona tres comandos para contar los elementos dentro de un conjunto ordenado: ZCARD, ZCOUNT, ZLEXCOUNT.

El comando ZCARD es la prueba básica para la cardinalidad de un conjunto. (Es análogo al comando SCARD para conjuntos). ZCARD devuelve el recuento de los miembros de un conjunto. Ejecutando el siguiente código para agregar elementos a un conjunto:

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

corriendo ZCard:

```
zcard favs
```

devuelve un valor de 4.

Los comandos ZCOUNT y ZLEXCOUNT le permiten contar un subconjunto de los elementos en un conjunto ordenado basado en un rango de valores. ZCOUNT le permite contar artículos dentro de un rango particular de puntajes y ZLEXCOUNT le permite contar el número de artículos dentro de un rango lexográfico particular.

Usando nuestro set de arriba:

```
zcount favs 2 5
```

devolvería un 3, ya que hay tres elementos (pizza, chocolate, cerveza) que tienen una puntuación de entre 2 y 5 inclusive.

ZLEXCOUNT está diseñado para funcionar con conjuntos en los que cada elemento tiene la misma puntuación, forzando y ordenando los nombres de los elementos. Si creamos un conjunto como:

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

podríamos usar ZLEXCOUNT para obtener el número de elementos en un rango lexográfico particular (esto se hace mediante una comparación de bytes usando la función memcpy).

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

volvería 2, ya que dos elementos (manzana, cerveza) caen dentro del rango manzana (inclusive) y chocolate (exclusivo). Podríamos alternativamente hacer ambos extremos inclusive:

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

y obtener el resultado 3.

Lea Conjuntos ordenados en línea: https://riptutorial.com/es/redis/topic/9111/conjuntos-ordenados

Capítulo 7: Geo

Introducción

Redis proporciona el tipo de datos GEO para trabajar con datos indexados geoespaciales.

Sintaxis

- GEOADD clave longitud latitud miembro [longitud latitud miembro ...]
- GEODIST clave miembro1 miembro2 [unidad]

Examples

GEOADD

El comando GEOADD permite a un usuario agregar información geoespacial (nombre del elemento, longitud, latitud) a una clave en particular.

El comando GEOADD se puede usar para agregar un solo elemento a una clave:

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

o múltiples elementos a una clave:

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

Geodist

El comando GEODIST permite a un usuario determinar la distancia entre dos miembros dentro de un índice geoespacial al especificar las unidades.

Para encontrar la distancia entre dos ciudades de encuentros:

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

Lea Geo en línea: https://riptutorial.com/es/redis/topic/9091/geo

Capítulo 8: Instalación y configuración

Examples

Instalando redis

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

A partir de Redis

```
redis-server
```

Compruebe si Redis está funcionando

```
redis-cli ping
```

Esto debería devolver `PONG`

Acceso Redis Cli

Suponiendo que está ejecutando el servidor redis en localhost, puede escribir el comando

```
redis-cli
```

Después de este comando aparece el símbolo de línea de comandos redis

```
127.0.0.1:6379>
```

Tipos de datos redis

La siguiente es la lista de todas las estructuras de datos soportadas por Redis:

- **Cuerdas binarias seguras**
- **Listas** : colecciones de elementos de cadena ordenados según el orden de inserción.
- **Conjuntos** : colecciones de elementos de cadena únicos, sin clasificar.
- **Conjuntos ordenados** : similares a los Conjuntos pero donde cada elemento de cadena está asociado a un valor de número flotante, llamado puntuación.
- **Hashes** : son mapas compuestos de campos asociados con valores.
- **HyperLogLogs** : es una estructura de datos probabilística que se utiliza para estimar la cardinalidad de un conjunto.

*Basado en la documentación oficial de **redis.io.***

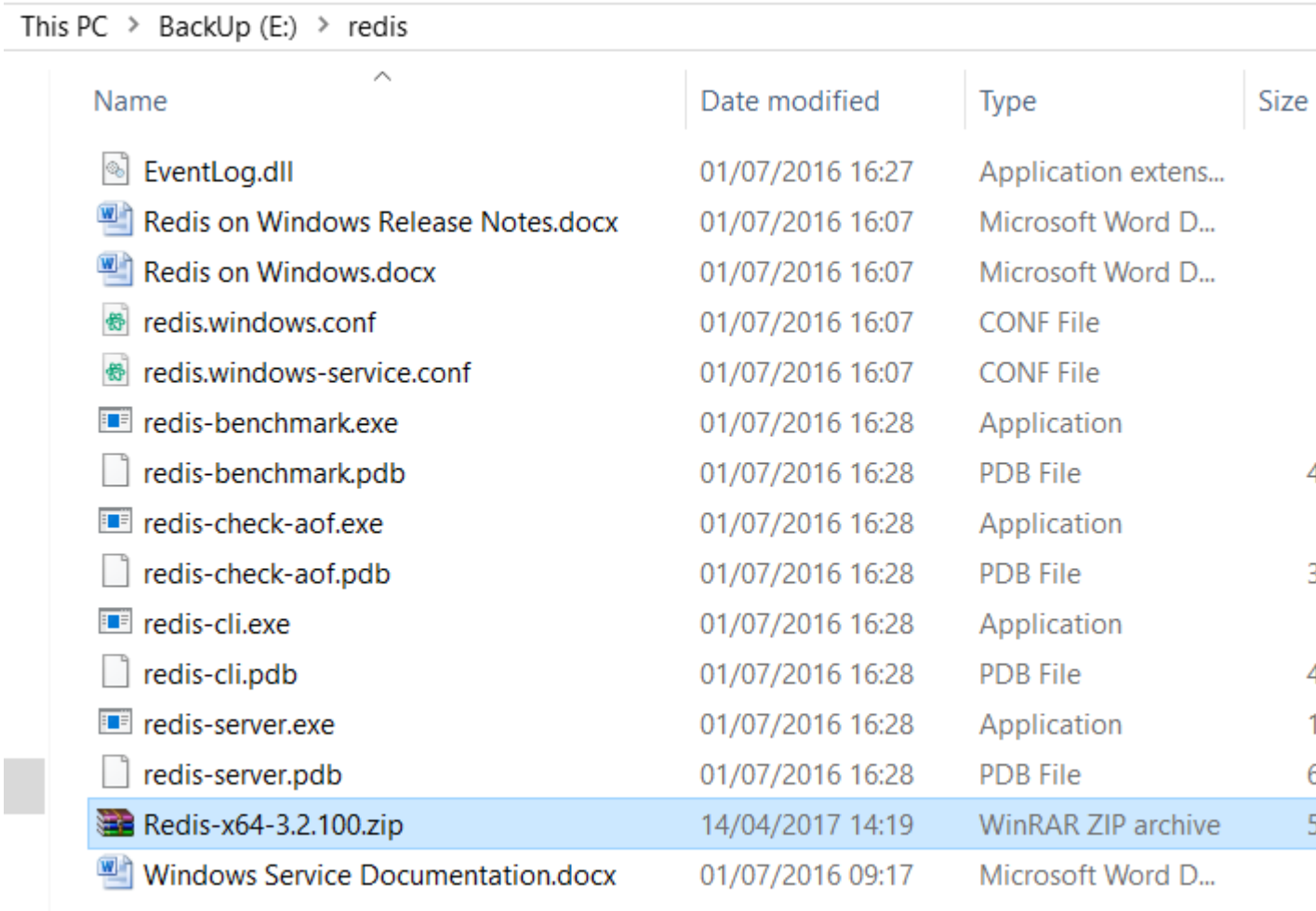
Instalación y ejecución de Redis Server en Windows

Nota: El proyecto Redis no es oficialmente compatible con Windows.

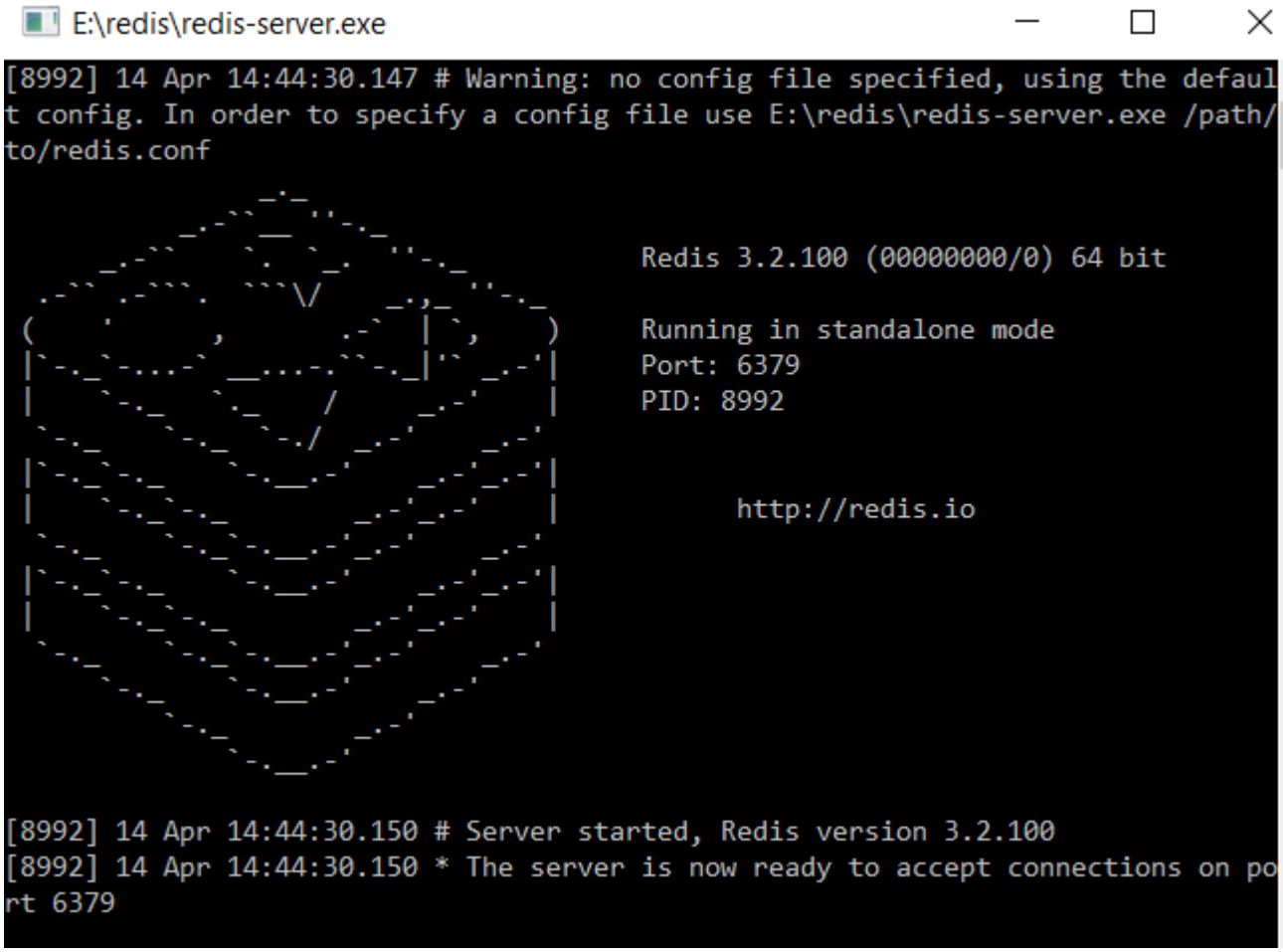
Sin embargo, el **grupo Microsoft Open Tech** desarrolla y mantiene este puerto de Windows orientado a Win64. **Oficial redis.io/download**

Puede elegir descargar diferentes versiones o la última versión de Redis
github.com/MSOpenTech/redis/releases

1. **Descargue el** archivo .msi o .zip, este tutorial le permitirá descargar el último archivo zip **Redis-x64-3.2.100.zip** .
2. **Extraiga el archivo zip** al directorio preparado.

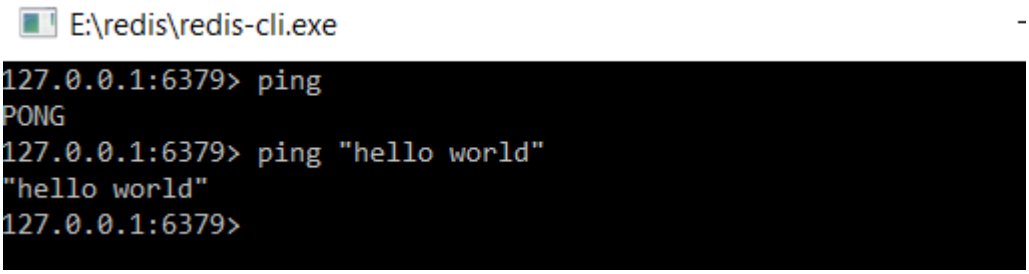


3. **Ejecute redis-server.exe** , puede ejecutar directamente redis-server.exe haciendo clic o ejecutando mediante el símbolo del sistema.



4. **Ejecute redis-cli.exe** , después de ejecutar con éxito el servidor redis. Puede acceder a él y probar los comandos ejecutando redis-cli.exe Te

El comando **PING** se usa para probar si una conexión está aún activa.



Ahora puede comenzar a utilizar Redis, consulte más comandos en las documentaciones oficiales.

Lea Instalación y configuración en línea: https://riptutorial.com/es/redis/topic/2898/instalacion-y-configuracion

Capítulo 9: Lua Scripting

Introducción

Redis proporciona un par de mecanismos para ampliar la funcionalidad de la base de datos. Un mecanismo es mediante el uso de scripts LUA del lado del servidor que se pueden ejecutar para manipular los datos. Los scripts de Lua pueden ser útiles para realizar operaciones costosas o para implementar operaciones atómicas que requieren lógica.

Examples

Comandos para scripting

Redis proporciona siete operaciones diferentes para trabajar con scripts:

- Evaluar operaciones (EVAL, EVALSHA)
- Operaciones de SCRIPT (DEBUG, EXISTS, FLUSH, KILL, LOAD)

El comando EVAL evalúa una secuencia de comandos proporcionada como un argumento de cadena al servidor. Los scripts pueden acceder a las claves de Redis especificadas como argumentos para el comando y a los parámetros de cadena adicionales que el usuario quiere pasar al script.

Por ejemplo, el comando:

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

provoca la ejecución de un script Lua definido por el usuario que simplemente devuelve los valores proporcionados. La llamada está involucrada con 2 teclas Redis (tecla 1 y tecla 2) y dos parámetros.

Otra forma de ejecutar un script Lua es cargarlo primero en la base de datos y luego ejecutarlo usando un hash SHA del 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"
```

El comando de carga de script carga el script y lo almacena en la base de datos. Se devuelve una firma sha del script para que pueda ser referenciada por futuras llamadas. La función EVALSHA toma el sha y ejecuta el script correspondiente desde la base de datos.

Lea Lua Scripting en línea: https://riptutorial.com/es/redis/topic/9112/lua-scripting

Capítulo 10: Pub / Sub

Introducción

Redis proporciona una implementación del patrón de mensajería de publicación / suscripción (publicación / suscripción). En lugar de enviar mensajes a receptores específicos, los editores envían mensajes a receptores interesados a través de algún mecanismo de direccionamiento indirecto. Los receptores especifican el interés en mensajes particulares. En Redis se accede a esta funcionalidad utilizando los comandos PUBLISH y SUBSCRIBE en los canales.

Sintaxis

- SUSCRIBIRSE canal [canal ...]
- UNSUBSCRIBE [canal [canal ...]]
- PUBLICAR mensaje de canal
- Patrón de PSUBSCRIBE [patrón ...]
- PUNSUBSCRIBE [patrón [patrón ...]]

Observaciones

Para manejar el pub / sub en redis, necesita tener **un cliente para suscribirse y otro cliente para publicar** . Ambos no pueden ser manejados por el mismo cliente. Aunque todos los demás comandos se pueden manejar con el mismo cliente.

Examples

Publicar y suscribirse con redis

Redis tiene publicación / suscripción para enviar mensajes. Esto se maneja suscribiéndose a un canal y publicando en el canal. Sí, los suscriptores se suscribirán a uno o más canales. El editor no necesita saber quiénes son todos suscriptores. En su lugar, el editor publicará en un canal específico. Todos los suscriptores que están suscritos a ese canal recibirán el mensaje. Este desacoplamiento de editores y suscriptores puede permitir una mayor escalabilidad y una topología de red más dinámica.

Ejemplo: el usuario se está suscribiendo a 2 canales, digamos foo & boo

```
SUBSCRIBE foo boo
```

En la consola 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
```

Comenzará a escuchar el mensaje. En la publicación obtendremos datos para el canal correspondiente.

Por ejemplo: cuando desee enviar un mensaje a todos los suscriptores que están conectados con boo, debe publicar en ese canal.

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

En la consola de redis-client1:

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

Para cancelar la suscripción del canal en cualquier momento, use

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

Puede suscribirse basado en el patrón también. Cuando el nombre del canal no está seguro / desea suscribirse según el patrón, use **PSUBSCRIBE** .

Al igual que **anular la** suscripción basado en el uso de patrones **PUNSUBSCRIBE**

Lea Pub / Sub en línea: https://riptutorial.com/es/redis/topic/5071/pub---sub

Capítulo 11: Redis List Datatype

Introducción

El tipo de datos de lista en Redis es una colección ordenada de elementos a los que se hace referencia mediante una clave de Redis. Redis le permite acceder y modificar una lista por índice o mediante operaciones push / pop. En Redis, los dos extremos de una lista se conocen como la izquierda y la derecha. La izquierda corresponde al primer elemento o encabezado de una lista y la derecha responde al último elemento o cola de una lista.

Sintaxis

- Valor de la clave LPUSH [valor ...]
- Valor de clave RPUSH [valor ...]
- Tecla LPOP
- Tecla RPOP
- Tecla LLEN

Observaciones

Puede encontrar más detalles sobre el tipo de datos de la Lista y todos los comandos que se pueden usar junto con ellos en la documentación oficial de Redis en Redis.io .

Examples

Agregar elementos a una lista

Redis le permite agregar elementos a la derecha oa la izquierda de una lista.

Si estaba trabajando con una lista, my_list y quería añadir 3 a la lista, podría hacerlo usando el comando Redis LPUSH:

```
LPUSH my_list 3
```

Si quisiera agregar 3 a my_list, en su lugar usaría el comando RPUSH:

```
RPUSH my_list 3
```

Tanto el comando LPUSH como el comando RPUSH crearán automáticamente una nueva lista si la clave provista no existe. Se pueden usar dos comandos alternativos LPUSHX y RPUSHX para operar solo con la tecla de lista si ya existe.

Obtención de elementos de una lista

Redis proporciona los comandos LPOP y RPOP como contrapartida de los comandos LPUSH y RPUSH para recuperar elementos de datos.

Si estaba trabajando con una lista my_list que ya tenía varios elementos de datos, puedo obtener el primer elemento de la lista mediante el comando LPOP:

```
LPOP my_list
```

El resultado de este comando devolverá el valor del primer elemento de la lista y lo eliminará de my_list. Por ejemplo, si tuviera la lista [1, 3, 2, 4] y le aplicara LPOP, tendría la lista [3, 2, 4] en la memoria después.

De manera similar, puedo eliminar del final de la lista usando RPOP:

```
RPOP my_list
```

devolvería el valor del último elemento de la lista y luego lo eliminaría de my_list. Usando nuestro ejemplo, [1, 2, 3, 4] después de llamar a RPOP en esta lista, la lista en la memoria sería [1, 2, 3].

Tamaño de una lista

El tamaño de una lista de Redis puede determinarse utilizando el comando LLEN. Si tengo una lista de cuatro elementos almacenada en la clave my_list, puedo obtener el tamaño usando:

```
LLEN my_list
```

que devolverá 4.

Si un usuario especifica una clave que no existe en LLEN, devolverá un cero, pero si se usa una clave que apunta a un elemento de un tipo de datos diferente, se devolverá un error.

Lea Redis List Datatype en línea: https://riptutorial.com/es/redis/topic/9107/redis-list-datatype

Capítulo 12: Redis Set Datatype

Introducción

Redis admite un tipo de datos de conjunto análogo a los conjuntos matemáticos para modelar datos en la base de datos. Los conjuntos son un tipo de datos compuesto que consiste en un grupo de miembros únicos y no ordenados. Los conjuntos admiten agregar y eliminar miembros, operaciones de tamaño, así como operaciones de combinación que toman dos conjuntos y generan un tercer conjunto. Los conjuntos en Redis son similares a los Conjuntos en la mayoría de los lenguajes de programación.

Sintaxis

- Miembro clave de SADD [miembro ...]
- Miembro clave de SISMEMBER
- Tecla SCARD
- Miembro clave de SADD [miembro ...]

Observaciones

La documentación completa sobre el tipo de datos del conjunto de Redis se puede encontrar en Redis.io .

Examples

Tamaño de un conjunto

El tamaño de un conjunto se puede determinar utilizando el comando SCARD. SCARD devolverá la cardinalidad de un conjunto o el número de miembros en el conjunto. Por ejemplo, si tuviera un Redis configurado my_set almacenado en la base de datos que parecía (Apple, Orange, Banana), podría obtener el tamaño utilizando el siguiente código:

```
SCARD my_set
```

En el caso de mi conjunto de ejemplos, esto devolvería 3. Si el usuario ejecuta un comando SCARD en una clave que no existe, Redis devolverá 0.

Agregar elementos a un conjunto

El comando básico de Redis para agregar un elemento a un conjunto es SADD. Toma una clave y uno o más miembros y los agrega al conjunto almacenado en la clave dada.

Por ejemplo, digamos que quería crear un set con los artículos manzana, pera y plátano. Podría ejecutar cualquiera de los siguientes:

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

o

```
SADD fruit apple pear banana
```

Después de ejecutar cualquiera de los dos, tendré el fruto establecido con 3 elementos.

Intentar agregar un elemento que ya está en el conjunto no tendrá ningún efecto. Después de configurar mi set de frutas con el código anterior, si intento agregar Apple nuevamente:

```
SADD fruit apple
```

Redis intentará agregar manzana al conjunto de frutas, pero como ya está en el conjunto, nada cambiará.

El resultado del comando SADD es siempre el número de elementos que Redis agrega a un conjunto. Así que intentar volver a agregar manzana, devolverá un resultado de 0.

Los elementos de miembros en Redis distinguen entre mayúsculas y minúsculas, por lo que Apple y Apple se tratan como dos elementos separados.

Pruebas de membresía

Redis proporciona el comando SISMEMBER para probar si un elemento en particular ya es miembro de un conjunto. Utilizando el comando SISMEMBER puedo probar y ver si Apple ya es miembro de mi conjunto de frutas.

Si construyo mi conjunto de frutas del ejemplo anterior, puedo verificar y ver si contiene manzana usando la siguiente prueba:

```
SISMEMBER fruit apple
```

SISMEMBER devolverá un 1 ya que el elemento ya está allí.

Si trato de ver si el perro es un miembro de mi juego de frutas:

```
SISMEMBER fruit dog
```

Redis devolverá un 0 ya que el perro no está en el set de frutas.

Si un usuario intenta usar el comando SISMEMBER con una clave que no existe, Redis devolverá un 0 que indica que no hay membresía, pero si usa SISMEMBER con una clave que ya tiene un tipo de datos no establecido, Redis devolverá un error.

Lea Redis Set Datatype en línea: https://riptutorial.com/es/redis/topic/9109/redis-set-datatype

Capítulo 13: Teclas redis

Introducción

El espacio de teclas Redis se puede considerar como una tabla hash o claves de mapeo de diccionario para las estructuras de datos en la base de datos.

Redis proporciona una amplia gama de comandos que trabajan con claves para administrar el espacio de claves, incluida la capacidad de eliminar claves, inspeccionar metadatos de claves, buscar claves y modificar ciertas propiedades de las claves.

Sintaxis

- Patrón de llaves
- Clave de persista
- EXPIRE segundos clave
- Clave de tiempo de EXPIREAT
- Clave TTL
- PEXPIRE clave milisegundos
- PEXPIREAT clave milisegundos-marca de tiempo
- Tecla PTTL
- Tecla DESBLOQUEAR [tecla ...]
- Tecla DEL [tecla ...]
- SCAN cursor [MATCH pattern] [COUNT count]

Observaciones

Para *los caracteres válidos en las teclas Redis* , el manual lo explica completamente :

> Las claves Redis son seguras para archivos binarios, esto significa que puedes usar cualquier secuencia binaria como una clave, desde una cadena como "foo" hasta el contenido de un archivo JPEG. La cadena vacía también es una clave válida.
>
> Algunas otras reglas sobre las claves:
>
> Las claves muy largas no son una buena idea, por ejemplo, una clave de 1024 bytes es una mala idea, no solo en cuanto a la memoria, sino también porque la búsqueda de la clave en el conjunto de datos puede requerir varias comparaciones de claves costosas. Incluso cuando la tarea en cuestión es hacer coincidir la existencia de un gran valor, recurrir al hash (por ejemplo, con SHA1) es una mejor idea, especialmente desde el punto de vista de la memoria y el ancho de banda.
>
> Las teclas muy cortas a menudo no son una buena idea. No tiene mucho sentido escribir "u1000flw" como clave si, en cambio, puede escribir "usuario: 1000: seguidores". Este último es más legible y el espacio agregado es menor en

comparación con el espacio utilizado por el propio objeto clave y el objeto de valor. Mientras que las teclas cortas obviamente consumirán un poco menos de memoria, su trabajo es encontrar el equilibrio correcto.

Trate de seguir con un esquema. Por ejemplo, "object-type: id" es una buena idea, como en "user: 1000". Los puntos o guiones se utilizan a menudo para campos de varias palabras, como en "comentario: 1234: responder.to" o "comentario: 1234: respuesta a".

El tamaño máximo permitido de la clave es de 512 MB.

Tenga cuidado al usar el comando KEYS contra un sistema de producción, ya que puede causar serios problemas de rendimiento. Si necesita hacer una búsqueda en el espacio de teclas, los comandos SCAN son una mejor alternativa.

Examples

Llaves validas

Las teclas redis son seguras para archivos binarios, por lo que, literalmente, cualquier cosa puede usarse como clave. Las únicas limitaciones son que deben ser menores a 512MB.

Ejemplos de claves válidas:

```
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.
```

Claves redis inválidas:

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

Esquemas de denominación clave

Para mayor claridad y facilidad de mantenimiento, a menudo se recomienda desarrollar un sistema o esquema para nombrar sus claves Redis. Aquí hay algunos ejemplos de sistemas comunes y mantenibles para nombrar sus claves:

```
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
```

Tenga en cuenta que, si bien está permitido, las claves más grandes utilizan más memoria y dan como resultado tiempos de búsqueda más lentos, por lo que el uso de una clave de 500 MB podría no ser una gran idea para el rendimiento. Una mejor idea podría ser usar un hash SHA-1, SHA-256 o MD5 de un objeto binario grande como una clave:

```
image/9517bb726d33efdc503a43582e6ea2eea309482b
image52e9df0577fca2ce022d4e8c86b1eccb070d37bef09dec36df2fabbfa7711f5c
```

Listado de todas las claves

Puede enumerar todas las claves en una base de datos de Redis ejecutando los siguientes comandos desde redis-cli:

```
KEYS *
```

El parámetro para KEYS es una expresión de coincidencia de patrón de estilo glob. Ejemplos de patrones soportados incluyen:

```
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
```

El uso del comando KEYS * puede tener efectos adversos en el rendimiento, por lo que no se recomienda para instancias de producción. Utilice la operación de EXPLORACIÓN para buscar claves en el código de producción.

TTL y clave de expiración

Los valores de caducidad de una clave pueden ser gestionados por un usuario fuera de los comandos de actualización. Redis le permite a un usuario determinar la hora actual de vida (TTL) de una tecla usando el comando TTL:

```
TTL key
```

Este comando devolverá el TTL de una tecla en segundos o devolverá los valores especiales -1 o

-2. Un -1 indica que la clave es persistente (no caducará) y un -2 indica que la clave no existe.

Una clave que caduca puede hacerse persistente usando el comando PERSIST:

```
PERSIST KEY
```

y se puede hacer que una clave persistente caduque utilizando el comando EXPIRE:

```
EXPIRE KEY seconds
```

Expire también se puede utilizar para modificar el TTL de una clave existente. Alternativamente, puede usar el comando EXPIREAT con una marca de tiempo UNIX para establecer un tiempo de caducidad.

Hay versiones de milisegundos de los comandos TTL, EXPIRE y EXPIREAT que tienen el prefijo de una P.

Borrando claves

Redis proporciona dos funciones para eliminar claves de la base de datos: del y desvinculación.

La función del elimina una o más claves de la base de datos. El comando del hace que Redis recupere inmediatamente la memoria para la clave eliminada en el hilo de ejecución actual. El tiempo de ejecución para del es proporcional al número de elementos individuales eliminados de todas las claves.

La función de desvinculación actúa como el comando del, elimina una o más claves de la base de datos. Sin embargo, a diferencia del comando del, cualquier memoria utilizada por esas claves se reclama de forma asíncrona en otro hilo.

Escaneando el espacio de teclas redis

Redis proporciona el comando SCAN para iterar sobre las claves en la base de datos que coinciden con un patrón particular. Redis admite la coincidencia de patrones de estilo glob en el comando SCAN.

El comando SCAN proporciona un iterador basado en el cursor sobre el espacio de teclas Redis. La secuencia de llamada iterativa a SCAN comienza cuando el usuario realiza una llamada con el argumento del cursor establecido en 0. El resultado de esa llamada es un lote de elementos y un cursor actualizado que se suministra a la siguiente llamada a SCAN. Esta iteración continúa hasta que Redis devuelve un cursor 0.

La siguiente función de Python demuestra el uso básico 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
```

El comando SCAN es la forma recomendada de buscar claves en la base de datos, y se recomienda sobre el comando `KEYS *` .

Lea Teclas redis en línea: https://riptutorial.com/es/redis/topic/3916/teclas-redis

Capítulo 14: Tipo de datos Redis String

Introducción

Redis proporciona un tipo de datos de cadena que se utiliza para asociar datos con una clave en particular. Las cadenas Redis son el tipo de datos más básico disponible en Redis y uno de los primeros tipos de datos con los que los usuarios aprenden a trabajar.

Las cadenas a menudo se asocian con datos de texto, pero las cadenas Redis son más bien como buffers que se pueden usar para almacenar una amplia gama de datos diferentes. Las cadenas Redis se pueden usar para representar números enteros, números de punto flotante, mapas de bits, texto y datos binarios.

Sintaxis

- Valor de la tecla SET [EX segundos] [milisegundos PX] [NX | XX]
- Tecla INCR
- INCRBY incremento de clave
- INCRBYFLOAT incremento de clave
- Tecla DECR
- DECRBY clave decremento

Examples

Trabajando con cuerdas como enteros

Varios comandos le permiten trabajar con cadenas que representan valores enteros.

Un usuario puede establecer el valor entero de una clave usando el comando:

```
SET intkey 2
```

El comando set creará la clave si es necesario o la actualizará si ya existe.

El valor de una clave entera se puede actualizar en el servidor usando los comandos INCR o INCRBY. INCR aumentará el valor de una clave en 1 e INCRBY aumentará el valor de la clave en el valor de paso proporcionado.

```
INCR intkey
INCRBY intkey 2
```

Si el valor de la clave especificada para INCR o INCRBY no se puede expresar como un entero, Redis devolverá un error. Si la clave no existe, la clave se creará y la operación se aplicará al valor predeterminado de 0.

Los comandos DECR y DECRBY funcionan a la inversa para disminuir el valor.

Trabajar con cadenas como números de punto flotante

Redis le permite usar el tipo de datos String para almacenar números de punto flotante.

Un usuario puede establecer el valor flotante de una clave con el comando:

```
SET floatkey 2.0
```

El comando set creará la clave si es necesario o la actualizará si ya existe.

El valor de la clave se puede actualizar en el servidor mediante el comando INCRBYFLOAT. INCRBYFLOAT aumentará el valor de una clave en el valor de incremento proporcionado.

```
INCRBYFLOAT floatkey 2.1
```

Si el valor de la clave especificada para INCRBYFLOAT no se puede expresar como un punto flotante, Redis devolverá un error. Si la clave no existe, la clave se creará y la operación se aplicará al valor predeterminado de 0.0.

Las claves se pueden disminuir al pasar un incremento negativo al comando INCRBYFLOAT.

Lea Tipo de datos Redis String en línea: https://riptutorial.com/es/redis/topic/9507/tipo-de-datos-redis-string

Creditos

S. No	Capítulos	Contributors
1	Empezando con redis	Ahamed Mustafa M, Alexander V., Aminadav, Community, Florian Hämmerle, Itamar Haber, Prashant Barve, RLaaa, Sagar Ranglani, sirin
2	Almacenamiento de persistencia Redis	Aminadav, Tague Griffith
3	Apoyo	odlp
4	Cómo conectarse a Redis en Java usando Jedis	Tague Griffith
5	Conectando a redis usando Python	Gianluca D'Ardia, Tague Griffith, ystark
6	Conjuntos ordenados	Tague Griffith
7	Geo	Tague Griffith
8	Instalación y configuración	Cristiana214, Gianluca D'Ardia, Sagar Ranglani
9	Lua Scripting	Tague Griffith
10	Pub / Sub	jerry, Tague Griffith
11	Redis List Datatype	Tague Griffith
12	Redis Set Datatype	JonyD, Tague Griffith
13	Teclas redis	Tague Griffith, Will
14	Tipo de datos Redis String	Tague Griffith