EBook Gratuito

APPRENDIMENTO redis

Free unaffiliated eBook created from
Stack Overflow contributors.

#redis

Sommario

Di ... 1

Capitolo 1: Iniziare con redis ... 2

Osservazioni ... 2

Versioni ... 2

Examples ... 2

Panoramica ... 2

Redis interfaccia a riga di comando ... 3

Redis "Hello World" ... 4

Installa Redis usando Docker ... 5

Installazione di Redis su Windows, con esempio Node.js ... 5

Capitolo 2: Come connettersi a Redis in Java usando Jedis ... 6

introduzione ... 6

Osservazioni ... 6

Examples ... 6

Ottenere Jedis ... 6

Connessione a Redis ... 7

Esecuzione dei comandi Get / Set di base ... 7

Esecuzione di comandi ... 8

Capitolo 3: Connessione a redis con Python ... 9

introduzione ... 9

Osservazioni ... 9

Examples ... 9

Aggiungi un elemento alla lista ... 9

Aggiunta di campi a un hash ... 9

Impostazione di una connessione a Redis ... 10

Creare una transazione ... 10

Esecuzione diretta dei comandi ... 10

Capitolo 4: di riserva ... 11

introduzione ... 11

Examples ... 11

Backup di un'istanza Redis remota in un'istanza locale 11
Parola d'ordine? 11
Inizia la replica 11
Controllo dell'avanzamento della sincronizzazione 12
Salvataggio di un dump di dati su disco 12
Arrestare la replica 12
Capitolo 5: Geo 13
introduzione 13
Sintassi 13
Examples 13
GEOADD 13
GEODIST 13
Capitolo 6: Installazione e configurazione 14
Examples 14
Installazione di Redis 14
Iniziare Redis 14
Controlla se Redis sta funzionando 14
Accedi a Redis Cli 14
Redis tipi di dati 14
Installazione ed esecuzione di Redis Server su Windows 15
Capitolo 7: Lua Scripting 17
introduzione 17
Examples 17
Comandi per lo scripting 17
Capitolo 8: Pub / Sub 18
introduzione 18
Sintassi 18
Osservazioni 18
Examples 18
Pubblica e sottoscrivi con redis 18
Capitolo 9: Redis List Datatype 20

introduzione 20
Sintassi 20
Osservazioni 20
Examples 20
Aggiungere elementi a un elenco 20
Ottenere oggetti da una lista 20
Dimensione di una lista 21
Capitolo 10: Redis Persistence Storage 22
introduzione 22
Examples 22
Disabilita tutta la memoria di persistenza in Redis 22
Ottieni lo stato di archiviazione della persistenza 22
Capitolo 11: Redis Set Datatype 23
introduzione 23
Sintassi 23
Osservazioni 23
Examples 23
Dimensione di un set 23
Aggiungere elementi a un set 23
Test per l'iscrizione 24
Capitolo 12: Redis String datatype 25
introduzione 25
Sintassi 25
Examples 25
Lavorare con String come numeri interi 25
Lavorare con le stringhe come numeri in virgola mobile 26
Capitolo 13: Set ordinati 27
introduzione 27
Sintassi 27
Osservazioni 27
Examples 27

Aggiunta di elementi a un set ordinato 27
Conteggio degli oggetti in un set ordinato 28
Capitolo 14: Tasti di Redis 30
introduzione 30
Sintassi 30
Osservazioni 30
Examples 31
Chiavi valide 31
Schemi di denominazione delle chiavi 31
Elenco di tutte le chiavi 32
TTL e scadenza chiave 32
Cancellare chiavi 33
Scansione di Redis Keyspace 33
Titoli di coda 35

Di

Capitolo 1: Iniziare con redis

Osservazioni

Questa sezione fornisce una panoramica di cosa è Redis e perché uno sviluppatore potrebbe volerlo utilizzare.

Dovrebbe anche menzionare qualsiasi argomento di grandi dimensioni all'interno di Redis e collegarsi agli argomenti correlati. Poiché la documentazione di Redis è nuova, potrebbe essere necessario creare versioni iniziali di tali argomenti correlati.

Versioni

Versione	Data di rilascio
3.2.3	2016/08/02
3.2.2	2016/07/28

Examples

Panoramica

Redis è un database remoto in memoria che offre prestazioni elevate, replica e un modello di dati univoco per produrre una piattaforma per la risoluzione dei problemi. Redis è una struttura dati in-memory open source (con licenza BSD), utilizzata come database, cache e broker di messaggi. È classificato come archivio valori-chiave NoSQL. Supporta strutture di dati come stringhe, hash, elenchi, set, serie ordinate con query di intervallo, bitmap, hyperloglog e indici geospaziali con query radius. Supportando cinque diversi tipi di strutture dati,

1. STRING (Funziona su tutta la stringa, parti, interi e galleggianti)
2. LISTA (spingere o rilasciare elementi da entrambe le estremità)
3. SET (Aggiungi, recupera, rimuovi, verifica, intersecano, unione, differenza ecc.)
4. HASH (store, fatch, remove in hash)
5. ZSET (uguale al set ma in modo ordinato)
6. GEO (Aggiungi, aggiorna, elimina latitudine e longitudine, entra all'interno di Redius dato)

Redis ha replica integrata, scripting Lua, sfratto LRU, transazioni e diversi livelli di persistenza on-disk (sync / async).

Prima della versione 3, Redis funziona in modalità master-slave e richiede Redis-Sentinel per fornire un'alta disponibilità. Solo il master accetta le scritture e sincronizza i dati con gli schiavi mediante la foratura.

Dalla versione 3, Redis funziona e consiglia la modalità multi-master in cui le funzioni di failover,

sharding / paritioning e resharding sono integrate. Redis-Sentinel non è richiesto dalla versione-3. Per far funzionare il cluster redis sono richiesti almeno 3 nodi / processi principali.

Altre funzionalità sono la replica, la persistenza e lo sharding sul lato client. Redis supporta un'ampia varietà di problemi che possono essere naturalmente associati a ciò che Redis offre, consentendoti di risolvere i tuoi problemi senza dover eseguire il lavoro concettuale richiesto da altri database.

Redis interfaccia a riga di comando

`redis-cli` è il programma di interfaccia a riga di comando di Redis che consente di inviare comandi a Redis e leggere le risposte inviate dal server, direttamente dal terminale. L'utilizzo della riga di comando di base è di seguito:

Accesso ai redis:

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

Accesso ai redis con autenticazione:

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

Seleziona il database e mostra le dimensioni del database (il numero di database predefinito è 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
```

Ottieni informazioni e statistiche sul server:

```
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
```

Uscendo dal redis-cli:

```
127.0.0.1:6379> exit
```

Redis "Hello World"

Per prima cosa è necessario installare e avviare il server Redis, controllare il seguente link che può aiutare a installare redis sul proprio server o sulla macchina locale.

Installazione e configurazione

Ora apri il prompt dei comandi ed esegui il comando `redis-cli` :

Per salvare il primo set> SET 'keyname' quindi 'value'

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

Premi Invio dovresti vedere

```
OK
```

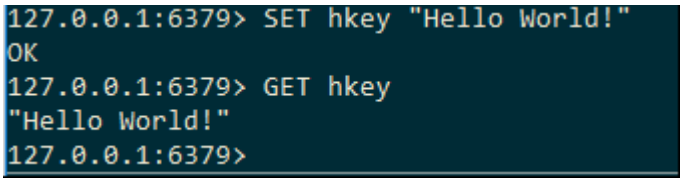
Quindi inserire:

```
GET hkey
```

tu dovresti vedere:

```
"Hello World!"
```

Esempio di output della schermata:



Installa Redis usando Docker

È semplice iniziare a utilizzare Redis tramite la finestra mobile:

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

Ora hai istanza in esecuzione sulla porta `6397`

Attenzione: tutti i dati saranno cancellati, quando Redis sarà fermato.

Per connettere redis-cli, avviare un'altra finestra mobile:

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

Ora puoi giocare con la tua finestra mobile Redis.

Installazione di Redis su Windows, con esempio Node.js

Redis ha una porta Windows fornita da "Microsoft Open Technologies". È possibile utilizzare l'installer msi trovato su: https://github.com/MSOpenTech/redis/releases

Una volta completata l'installazione, è possibile vedere "Redis" è un servizio di Windows (e lo stato dovrebbe essere "Avviato")

Per scrivere un esempio di 'Hello world' che utilizza Redis in Node.js (anche in Windows) puoi utilizzare il seguente modulo npm: https://www.npmjs.com/package/redis

esempio di codice:

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

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

Leggi Iniziare con redis online: https://riptutorial.com/it/redis/topic/1724/iniziare-con-redis

Capitolo 2: Come connettersi a Redis in Java usando Jedis

introduzione

Esistono più di dieci diverse librerie client da utilizzare con Redis in Java. Uno dei clienti più popolari è Jedis .

Osservazioni

Ulteriori informazioni:

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

Examples

Ottenere Jedis

La libreria Jedis viene generalmente aggiunta al progetto Java utilizzando un sistema di gestione delle dipendenze integrato nell'ambiente di generazione del progetto. Due famosi sistemi di build Java sono Maven e Gradle.

Usando Gradle

Per aggiungere la libreria Jedis a un progetto Gradle, è necessario configurare un repository e aggiungere una dipendenza. Il seguente frammento mostra come aggiungere la versione 2.9.0 della libreria Jedis a un progetto Gradle.

```
repositories {
    mavenCentral()
}

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

Usando Maven

Per aggiungere Jedis a un progetto Maven, devi aggiungere una dipendenza all'elenco delle dipendenze e fornire le coordinate della libreria. Il seguente frammento verrà aggiunto al tuo file pom.xml:

```
<dependencies>
```

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

Connessione a Redis

Usare una piscina

La maggior parte del codice vorrà connettersi a Redis usando un pool di oggetti di connessione condivisi. Il collegamento a Redis tramite pool richiede due blocchi di codice diversi. Al momento dell'inizializzazione, l'applicazione deve creare il pool di connessioni:

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

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

`JedisPoolConfig` fornisce opzioni per l'ottimizzazione del pool.

Poiché l'applicazione elabora il suo carico di lavoro, sarà necessario ottenere una connessione dal pool condiviso usando il seguente codice:

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

        ...
    }
```

La migliore pratica è ottenere l'oggetto connessione `Jedis` dal pool all'interno di un blocco try-with-resources.

Senza piscine

In alcuni casi, ad esempio una semplice applicazione o un test di integrazione, potresti non voler gestire i pool condivisi e invece creare direttamente l'oggetto di connessione `Jedis` . Questo può essere ottenuto con il seguente codice:

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

    . . .
}
```

Di nuovo, la procedura migliore è creare l'oggetto client Jedis all'interno di un blocco try-with-resources.

Esecuzione dei comandi Get / Set di base

Una volta stabilita una connessione a Redis, è possibile ottenere e impostare valori utilizzando l'oggetto di connessione `Jedis` :

Ottenere

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

Impostato

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

Esecuzione di comandi

Per eseguire un comando Redis usando Jedis, si effettuano chiamate di metodo contro l'oggetto `Jedis` creato dal pool. Jedis espone i comandi Redis come chiamate di metodo, alcuni esempi sono:

```
- 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)
```

Se si desidera impostare il valore di una chiave String in Redis, si utilizzerà un blocco di codice simile a:

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

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

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

Leggi Come connettersi a Redis in Java usando Jedis online:
https://riptutorial.com/it/redis/topic/9712/come-connettersi-a-redis-in-java-usando-jedis

Capitolo 3: Connessione a redis con Python

introduzione

La connessione a Redis in Python richiede l'uso di una libreria client. Esistono molte librerie client diverse per Python, ma **redis-py** è uno dei client più popolari in uso.

Una volta installata la libreria client, è possibile accedere a Redis nell'applicazione importando il modulo appropriato, stabilendo una connessione e quindi eseguendo un comando.

Osservazioni

Per connetterti su redis con python devi installare un **client** . Puoi installare con pip usando:

```
pip install redis
```

questo installerà **redis-py**

Facoltativamente, è possibile installare **hiredis-py** che deleghi l'analisi dei messaggi di protocollo al client C hiredis. Ciò può fornire un significativo miglioramento delle prestazioni in molte situazioni. È possibile installare hiredis con pip eseguendo:

```
pip install hiredis
```

Examples

Aggiungi un elemento alla lista

```
import redis

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

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

Aggiunta di campi a un hash

Ci sono due funzioni principali in Redis (HSET e HMSET) per aggiungere campi a una chiave hash. Entrambe le funzioni sono disponibili in redis-py.

Utilizzo di HSET:

```
import redis

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

Utilizzando HMSET:

```
import redis

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

Impostazione di una connessione a Redis

Il client **redis-py** fornisce due classi `StrictRedis` e `Redis` per stabilire una connessione di base a un database Redis. La classe `Redis` viene fornita per la compatibilità con le versioni precedenti e i nuovi progetti devono utilizzare la classe `StrictRedis` .

Uno dei modi consigliati per stabilire una connessione consiste nel definire i parametri di connessione in un dizionario e passare il dizionario al costruttore `StrictRedis` utilizzando la sintassi `**` .

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

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

Creare una transazione

È possibile stabilire una transazione chiamando il metodo `pipeline` su `StrictRedis` . I comandi redis eseguiti contro la transazione vengono eseguiti in un unico blocco.

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

Esecuzione diretta dei comandi

Redis-py fornisce il metodo `execute_command` per richiamare direttamente le operazioni Redis. Questa funzionalità può essere utilizzata per accedere a tutti i moduli che potrebbero non avere un'interfaccia supportata nel client redis-py. Ad esempio, è possibile utilizzare `execute_command` per elencare tutti i moduli caricati in un server Redis:

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

Leggi Connessione a redis con Python online:
https://riptutorial.com/it/redis/topic/9103/connessione-a-redis-con-python

Capitolo 4: di riserva

introduzione

Il backup di un'istanza Redis remota può essere ottenuto con la replica. Ciò è utile se si desidera acquisire un'istantanea di un set di dati prima di aggiornare, eliminare o modificare un database Redis.

Examples

Backup di un'istanza Redis remota in un'istanza locale

Sulla macchina su cui desideri eseguire il backup, passa a CLI Redis:

```
redis-cli
```

Parola d'ordine?

Se il master Redis DB (quello che si desidera replicare) ha una password:

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

Inizia la replica

Esegui quanto segue per iniziare la replica:

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

Per verificare la replica è in corso la corsa:

```
INFO replication
```

E dovresti vedere un risultato come questo:

```
# 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
```

Si noti la `master_link_status` dovrebbe essere `up` .

Controllo dell'avanzamento della sincronizzazione

Al termine della sincronizzazione, la `INFO replication` dovrebbe mostrare:

```
master_sync_in_progress:0
```

Per verificare che il set di dati sia stato sincronizzato è possibile confrontare la dimensione del database:

```
DBSIZE
```

Salvataggio di un dump di dati su disco

Per salvare il DB su disco in modo asincrono:

```
BGSAVE
CONFIG GET dir
```

Quindi dovresti trovare un file `dump.rdb` nella directory elencata dal comando config.

Arrestare la replica

È possibile interrompere la replica con:

```
SLAVEOF NO ONE
```

Riferimento: guida alla replica di Redis

Leggi di riserva online: https://riptutorial.com/it/redis/topic/9369/di-riserva

Capitolo 5: Geo

introduzione

Redis fornisce il tipo di dati GEO per lavorare con dati indicizzati geospaziali.

Sintassi

- GEOADD key longitude latitude member [longitude latitude member ...]
- Chiave GEODIST membro1 membro2 [unità]

Examples

GEOADD

Il comando GEOADD consente a un utente di aggiungere informazioni geospaziali (nome oggetto, longitudine, latitudine) a una chiave particolare.

Il comando GEOADD può essere utilizzato per aggiungere un singolo elemento a una chiave:

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

o più elementi per una chiave:

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

GEODIST

Il comando GEODIST consente a un utente di determinare la distanza tra due membri all'interno di un indice geospaziale mentre specifica le unità.

Per trovare la distanza tra due città meetup:

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

Leggi Geo online: https://riptutorial.com/it/redis/topic/9091/geo

Capitolo 6: Installazione e configurazione

Examples

Installazione di Redis

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

Iniziare Redis

```
redis-server
```

Controlla se Redis sta funzionando

```
redis-cli ping
```

Questo dovrebbe restituire `PONG`

Accedi a Redis Cli

Supponendo di eseguire il server redis su localhost, è possibile digitare command

```
redis-cli
```

Dopo questo comando appare redis prompt della riga di comando

```
127.0.0.1:6379>
```

Redis tipi di dati

Di seguito è riportato l'elenco di tutte le strutture dati supportate da Redis:

- **Stringhe binarie**
- **Elenchi** : raccolte di elementi stringa ordinati secondo l'ordine di inserimento.
- **Imposta** : raccolte di elementi di stringa unici, non ordinati.
- **Set ordinati** : simili agli Insiemi ma in cui ogni elemento stringa è associato a un valore numerico mobile, chiamato punteggio.
- **Hash** : sono mappe composte da campi associati a valori.
- **HyperLogLogs** : si tratta di una struttura dati probabilistica che viene utilizzata per stimare la cardinalità di un set.

*Basato sulla documentazione ufficiale **redis.io***

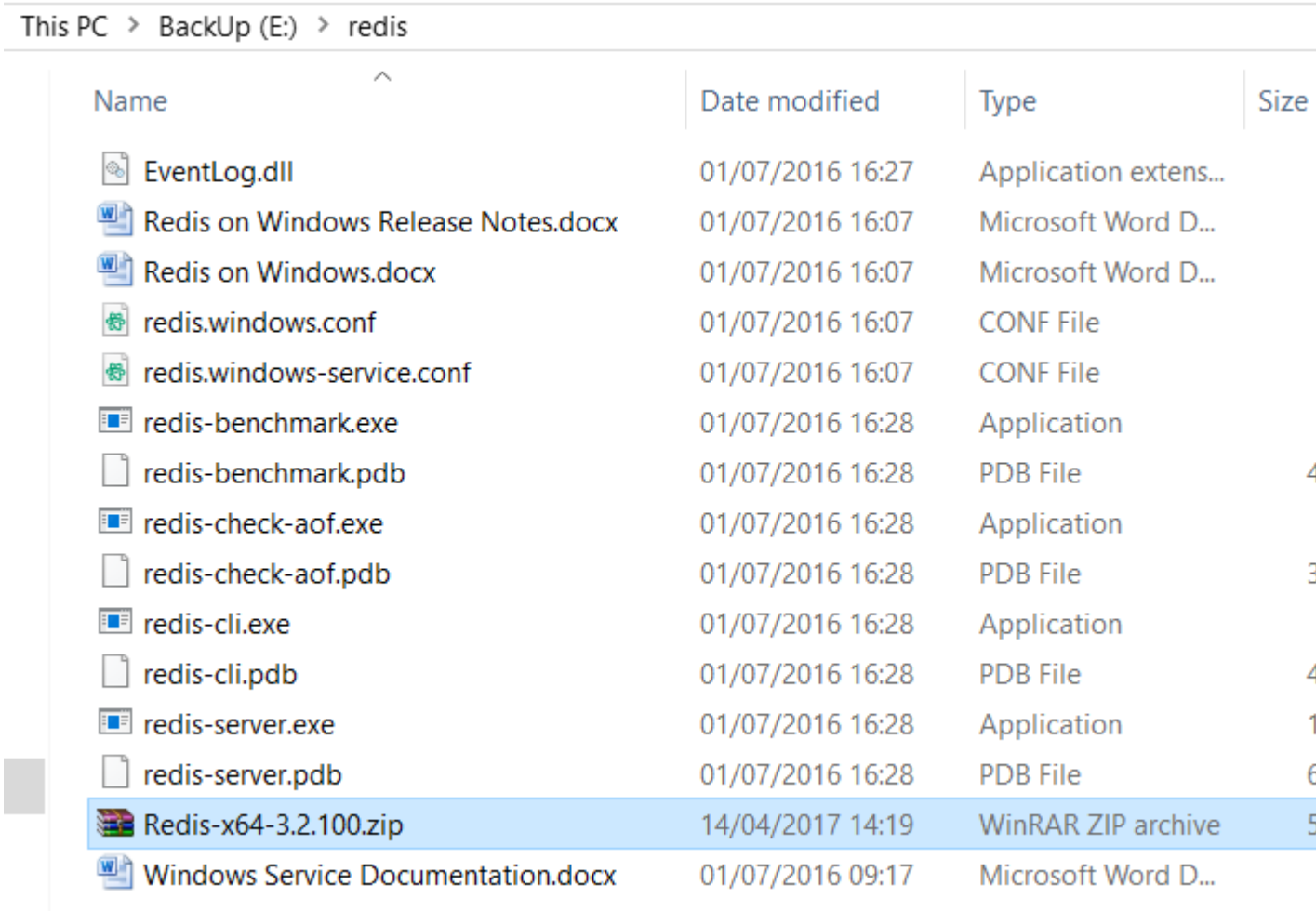
Installazione ed esecuzione di Redis Server su Windows

Nota: il progetto Redis non supporta ufficialmente Windows.

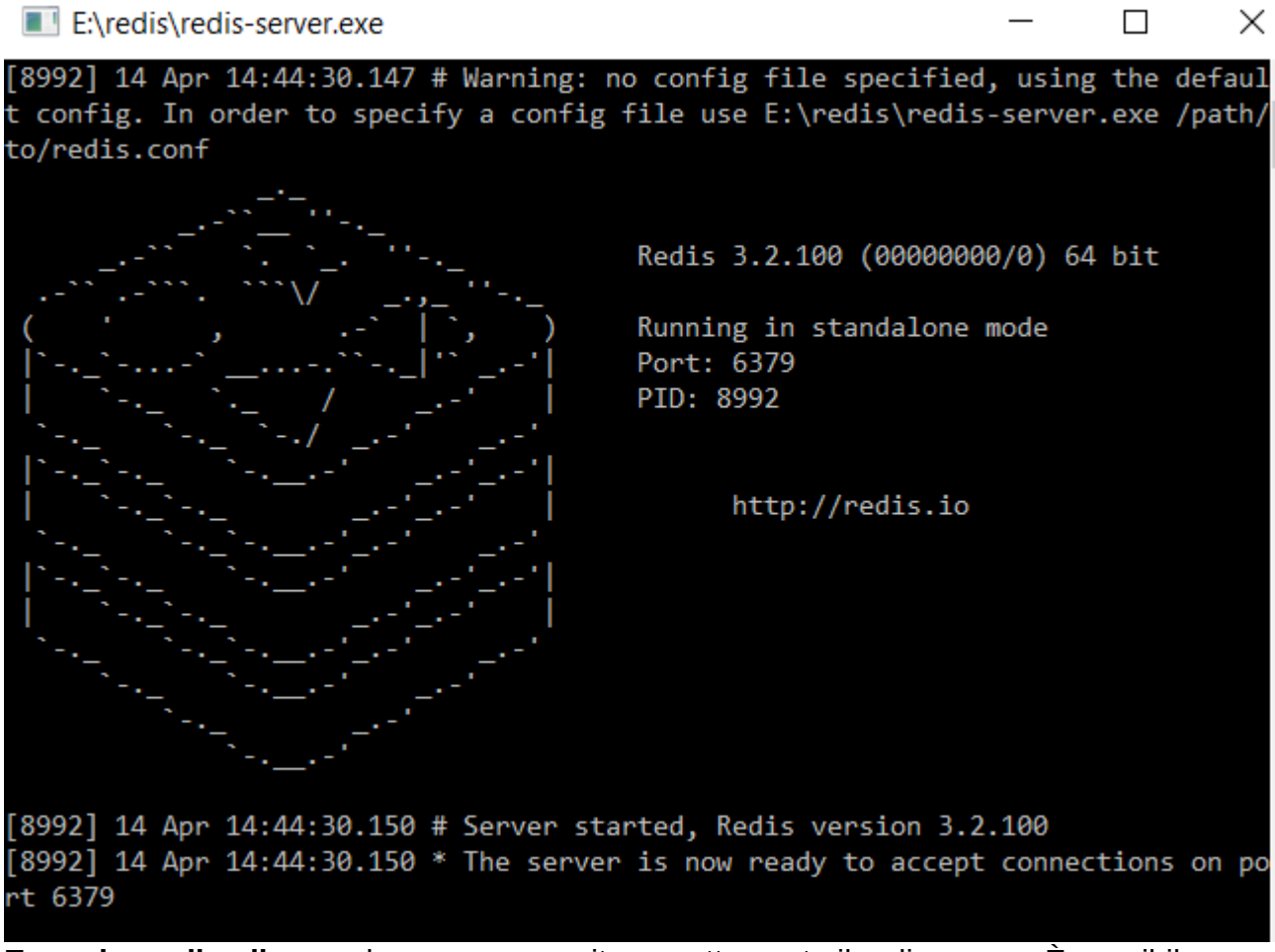
Tuttavia, il **gruppo Microsoft Open Tech** sviluppa e mantiene questa porta Windows che punta a Win64. **Redis.io/download ufficiale**

È possibile scegliere di scaricare versioni diverse o l'ultima versione di Redis github.com/MSOpenTech/redis/releases

1. **Scarica il** file .msi o .zip, questo tutorial ti consentirà di scaricare l'ultimo file zip **Redis-x64-3.2.100.zip** .
2. **Estrai il file zip** nella directory preparata.

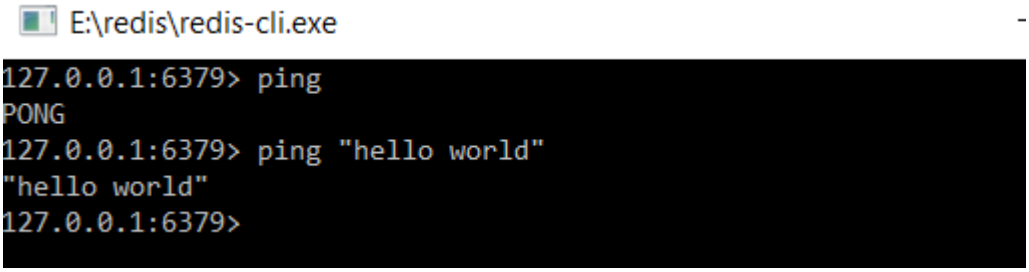


3. **Eseguire redis-server.exe** , è possibile eseguire direttamente redis-server.exe facendo clic o eseguendo tramite prompt dei comandi.



4. **Eseguire redis-cli.exe** , dopo aver eseguito correttamente il redis-server. È possibile accedervi e testare i comandi eseguendo redis-cli.exe Te

Il comando **PING** viene utilizzato per verificare se una connessione è ancora attiva.



È ora possibile iniziare a utilizzare Redis, fare riferimento a ulteriori comandi nelle documentazioni ufficiali

Leggi Installazione e configurazione online: https://riptutorial.com/it/redis/topic/2898/installazione-e-configurazione

Capitolo 7: Lua Scripting

introduzione

Redis fornisce un paio di meccanismi per estendere la funzionalità del database. Un meccanismo è l'utilizzo di script LUA sul lato server che possono essere eseguiti per manipolare i dati. Gli script Lua possono essere utili per eseguire operazioni costose o per implementare operazioni atomiche che richiedono la logica.

Examples

Comandi per lo scripting

Redis offre sette diverse operazioni per lavorare con gli script:

- Operazioni Eval (EVAL, EVALSHA)
- Operazioni SCRIPT (DEBUG, EXISTS, FLUSH, KILL, LOAD)

Il comando EVAL valuta uno script fornito come argomento stringa sul server. Gli script possono accedere alle chiavi Redis specificate denominate come argomenti del comando e ai parametri di stringa aggiuntivi che l'utente desidera passare allo script.

Ad esempio, il comando:

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

causa l'esecuzione di uno script Lua definito dall'utente che restituisce semplicemente i valori forniti. La chiamata è coinvolta con 2 tasti Redis (tasto 1 e tasto 2) e due parametri.

Un altro modo per eseguire uno script Lua è caricarlo prima nel database, quindi eseguirlo usando un hash SHA dello 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"
```

Il comando di caricamento dello script carica lo script e lo memorizza nel database. Viene restituita una firma sha dello script in modo che possa essere referenziata da chiamate future. La funzione EVALSHA prende lo sha ed esegue lo script corrispondente dal database.

Leggi Lua Scripting online: https://riptutorial.com/it/redis/topic/9112/lua-scripting

Capitolo 8: Pub / Sub

introduzione

Redis fornisce un'implementazione del modello di messaggistica di pubblicazione / sottoscrizione (Pub / Sub). Invece di inviare messaggi a destinatari specifici, gli editori inviano messaggi ai destinatari interessati tramite un meccanismo indiretto. I destinatari specificano interesse in particolari messaggi. In Redis è possibile accedere a questa funzionalità utilizzando i comandi PUBLISH e SUBSCRIBE sui canali.

Sintassi

- ISCRIVITI canale [canale ...]
- UNSUBSCRIBE [canale [canale ...]]
- PUBLISH messaggio del canale
- Modello PSUBSCRIBE [modello ...]
- PUNSUBSCRIBE [modello [modello ...]]

Osservazioni

Per gestire il pub / sub in redis, è necessario avere **un client per iscriversi e client diversi per la pubblicazione** . Entrambi non possono essere gestiti dallo stesso cliente. Sebbene tutti gli altri comandi possano ancora essere gestiti con lo stesso client.

Examples

Pubblica e sottoscrivi con redis

Redis ha pubblicato / sottoscritto per l'invio di messaggi. Questo viene gestito sottoscrivendo un canale e pubblicando sul canale. Sì, gli abbonati si iscriveranno a uno o più canali. L'editore non deve sapere chi sono tutti gli abbonati. Invece, l'editore pubblicherà su un canale specifico. Tutti gli abbonati che sono iscritti per quel canale riceveranno il messaggio. Questo disaccoppiamento di editori e abbonati può consentire una maggiore scalabilità e una topologia di rete più dinamica.

Esempio: l' utente è abbonato a 2 canali dire foo & boo

```
SUBSCRIBE foo boo
```

Nella console di 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
```

Inizierà ad ascoltare il messaggio. Alla pubblicazione otterrà i dati per il canale corrispondente.

Ad esempio: quando si desidera inviare un messaggio a tutti gli abbonati che sono connessi con boo, è necessario pubblicare su quel canale.

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

Nella console di redis-client1:

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

Per annullare l'iscrizione dal canale in qualsiasi momento, utilizzare

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

Può iscriversi anche in base al modello. Se il nome del canale non è sicuro / vuoi iscriverti in base al pattern, utilizza **PSUBSCRIBE** .

Allo stesso modo per annullare l'iscrizione in base al modello, utilizzare **PUNSUBSCRIBE**

Leggi Pub / Sub online: https://riptutorial.com/it/redis/topic/5071/pub---sub

Capitolo 9: Redis List Datatype

introduzione

Il tipo di dati Elenco in Redis è una raccolta ordinata di elementi a cui fa riferimento una chiave Redis. Redis ti consente di accedere e modificare un elenco per indice o per operazioni push / pop. In Redis, le due estremità di una lista sono indicate come sinistra e destra. La sinistra corrisponde al primo elemento o alla testa di una lista e la destra corrisponde all'ultimo elemento o coda di una lista.

Sintassi

- Valore chiave LPUSH [valore ...]
- Valore chiave RPUSH [valore ...]
- Chiave LPOP
- Chiave RPOP
- Chiave LLEN

Osservazioni

Maggiori dettagli sul tipo di dati List e su tutti i comandi che possono essere utilizzati insieme a essi possono essere trovati nella documentazione Redis ufficiale su Redis.io .

Examples

Aggiungere elementi a un elenco

Redis ti consente di aggiungere elementi alla destra o alla sinistra di una lista.

Se stavo lavorando con una lista, my_list e io volevamo anteporre 3 alla lista, avrei potuto farlo usando il comando LPUSH di Redis:

```
LPUSH my_list 3
```

Se volessi aggiungere 3 a my_list, preferirei utilizzare il comando RPUSH:

```
RPUSH my_list 3
```

Entrambi i comandi LPUSH e RPUSH creeranno automaticamente un nuovo elenco per te se la chiave fornita non esiste. Due comandi alternativi LPUSHX e RPUSHX possono essere utilizzati per operare solo sulla chiave dell'elenco, se già esiste.

Ottenere oggetti da una lista

Redis fornisce i comandi LPOP e RPOP come controparte dei comandi LPUSH e RPUSH per il recupero degli elementi di dati.

Se stavo lavorando con un elenco my_list che conteneva già diversi elementi di dati, posso ottenere il primo elemento nell'elenco usando il comando LPOP:

```
LPOP my_list
```

Il risultato di questo comando restituirà il valore del primo elemento dall'elenco e lo rimuoverà da my_list. Per esempio, se avessi la lista [1, 3, 2, 4] e ho applicato LPOP ad essa, avrei la lista [3, 2, 4] in memoria in seguito.

Allo stesso modo, posso rimuovere dalla fine della lista usando RPOP:

```
RPOP my_list
```

restituirebbe il valore per l'ultimo elemento della lista e quindi rimuoverlo da my_list. Usando il nostro esempio [1, 2, 3, 4] dopo aver chiamato RPOP su questo elenco, l'elenco in memoria sarebbe [1, 2, 3].

Dimensione di una lista

La dimensione di un elenco Redis può essere deter minata usando il comando LLEN. Se ho un elenco di quattro elementi memorizzato nella chiave my_list, posso ottenere la dimensione usando:

```
LLEN my_list
```

che restituirà 4.

Se un utente specifica una chiave che non esiste per LLEN, restituirà uno zero, ma se viene utilizzata una chiave che punta a un elemento di un tipo di dati diverso, verrà restituito un errore.

Leggi Redis List Datatype online: https://riptutorial.com/it/redis/topic/9107/redis-list-datatype

Capitolo 10: Redis Persistence Storage

introduzione

Redis supporta due principali modalità di persistenza: RDB e AOF. La modalità di persistenza RDB acquisisce un'istantanea del database in un determinato momento. In modalità RDB, Redis esegue la foratura di un processo per mantenere il database su disco. AOF registra ogni operazione eseguita sul server in un registro di riproduzione che può essere elaborato all'avvio per ripristinare lo stato del database.

Examples

Disabilita tutta la memoria di persistenza in Redis

Esistono due tipi di modalità di memorizzazione persistenti in Redis: AOF e RDB. Per disabilitare temporaneamente RDB, eseguire i seguenti comandi sulla riga di comando Redis:

```
config set save ""
```

per disattivare temporaneamente AOF eseguire quanto segue dalla riga di comando Redis:

```
config set appendonly no
```

Le modifiche permarranno fino al riavvio del server, quindi il server tornerà a qualsiasi modalità configurata nel file redis.conf del server.

Il comando `CONFIG REWRITE` può essere utilizzato per modificare il file redis.conf per riflettere eventuali modifiche dinamiche alla configurazione.

Ottieni lo stato di archiviazione della persistenza

Il codice seguente otterrà la configurazione corrente per lo stato di archiviazione permanente. Questi valori possono essere modificati dinamicamente, quindi potrebbero differire dalla configurazione in redis.conf:

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

Leggi Redis Persistence Storage online: https://riptutorial.com/it/redis/topic/7871/redis-persistence-storage

Capitolo 11: Redis Set Datatype

introduzione

Redis supporta un set di dati analogo a set matematici per la modellazione dei dati nel database. Gli insiemi sono un tipo di dati composto costituito da un gruppo di membri unici e non ordinati. Imposta il supporto per l'aggiunta e la rimozione di membri, le operazioni sulle dimensioni, nonché le operazioni di combinazione che richiedono due set e generano un terzo set. I set in Redis sono simili agli Sets nella maggior parte dei linguaggi di programmazione.

Sintassi

- Membro chiave SADD [membro ...]
- Membro chiave di SISMEMBER
- Tasto SCARD
- Membro chiave SADD [membro ...]

Osservazioni

La documentazione completa sul tipo di dati set Redis è disponibile su Redis.io .

Examples

Dimensione di un set

La dimensione di un set può essere determinata usando il comando SCARD. SCARD restituirà la cardinalità di un set o il numero di membri nel set. Ad esempio, se avessi un set Redis my_set memorizzato nel database che sembrava (Apple, Orange, Banana), avrei potuto ottenere le dimensioni usando il seguente codice:

```
SCARD my_set
```

Nel caso del mio esempio impostato, ciò restituirebbe 3. Se l'utente esegue un comando SCARD su una chiave che non esiste, Redis restituirà 0.

Aggiungere elementi a un set

Il comando di base di Redis per aggiungere un elemento a un set è SADD. Prende una chiave e uno o più membri e li aggiunge al set memorizzato nella chiave specificata.

Ad esempio, diciamo che volevo creare un set con gli oggetti mela, pera e banana. Potrei eseguire una delle seguenti operazioni:

```
SADD fruit apple
```

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

o

```
SADD fruit apple pear banana
```

Dopo aver eseguito entrambi, avrò il frutto impostato con 3 elementi.

Il tentativo di aggiungere un elemento già presente nel set non avrà alcun effetto. Dopo aver impostato il mio set di frutta usando il codice sopra, se provo ad aggiungere nuovamente Apple:

```
SADD fruit apple
```

Redis tenterà di aggiungere la mela al set di frutta, ma dal momento che è già nel set nulla cambierà.

Il risultato del comando SADD è sempre il numero di elementi Redis aggiunti a un set. Quindi, tentando di aggiungere nuovamente apple, verrà restituito un risultato pari a 0.

Gli elementi membri in Redis distinguono tra maiuscole e minuscole, quindi Apple e Apple vengono considerati due elementi separati.

Test per l'iscrizione

Redis fornisce il comando SISMEMBER per verificare se un determinato elemento è già membro di un set. Utilizzando il comando SISMEMBER posso verificare e vedere se la mela è già un membro del mio set di frutta.

Se costruisco il mio frutto dall'esempio precedente, posso controllare e vedere se contiene apple usando il seguente test:

```
SISMEMBER fruit apple
```

SISMEMBER restituirà un 1 poiché l'articolo è già lì.

Se provassi a vedere se il cane è un membro del mio set di frutta:

```
SISMEMBER fruit dog
```

Redis restituirà uno 0 poiché il cane non è nella serie di frutta.

Se un utente tenta di utilizzare il comando SISMEMBER con una chiave che non esiste, Redis restituirà uno 0 che non indica l'appartenenza, ma se si utilizza SISMEMBER con una chiave che contiene già un tipo di dati non impostato, Redis restituirà un errore.

Leggi Redis Set Datatype online: https://riptutorial.com/it/redis/topic/9109/redis-set-datatype

Capitolo 12: Redis String datatype

introduzione

Redis fornisce un tipo di dati stringa che viene utilizzato per associare i dati a una particolare chiave. La stringa Redis è il tipo di dati più semplice disponibile in Redis e uno dei primi tipi di dati con cui gli utenti imparano a lavorare.

Le stringhe sono spesso associate a dati di testo, ma le stringhe di Redis sono più simili ai buffer che possono essere utilizzate per memorizzare una vasta gamma di dati diversi. Le stringhe Redis possono essere utilizzate per rappresentare numeri interi, numeri in virgola mobile, bitmap, testo e dati binari.

Sintassi

- Valore chiave SET [EX secondi] [PX millisecondi] [NX | XX]
- Tasto INCR
- Incremento chiave INCRBY
- Incremento chiave INCRBYFLOAT
- Tasto DECR
- DECRBY decremento chiave

Examples

Lavorare con String come numeri interi

Diversi comandi consentono di lavorare con stringhe che rappresentano valori interi.

Un utente può impostare il valore intero di una chiave usando il comando:

```
SET intkey 2
```

Il comando set creerà la chiave se necessario o aggiornerà se già esiste.

Il valore di una chiave intera può essere aggiornato sul server utilizzando i comandi INCR o INCRBY. INCR aumenterà il valore di una chiave di 1 e INCRBY aumenterà il valore della chiave per il valore di passo fornito.

```
INCR intkey
INCRBY intkey 2
```

Se il valore della chiave specificato in INCR o INCRBY non può essere espresso come numero intero, Redis restituirà un errore. Se la chiave non esiste, la chiave verrà creata e l'operazione verrà applicata al valore predefinito di 0.

I comandi DECR e DECRBY funzionano in ordine inverso per decrementare il valore.

Lavorare con le stringhe come numeri in virgola mobile

Redis ti permettono di usare il tipo di dati String per memorizzare numeri in virgola mobile.

Un utente può impostare il valore float di una chiave usando il comando:

```
SET floatkey 2.0
```

Il comando set creerà la chiave se necessario o aggiornerà se già esiste.

Il valore della chiave può essere aggiornato sul server utilizzando il comando INCRBYFLOAT. INCRBYFLOAT aumenterà il valore di una chiave con il valore di incremento fornito.

```
INCRBYFLOAT floatkey 2.1
```

Se il valore della chiave specificato in INCRBYFLOAT non può essere espresso come virgola mobile, Redis restituirà un errore. Se la chiave non esiste, la chiave verrà creata e l'operazione verrà applicata al valore predefinito di 0.0.

Le chiavi possono essere decrementate passando un incremento negativo al comando INCRBYFLOAT.

Leggi Redis String datatype online: https://riptutorial.com/it/redis/topic/9507/redis-string-datatype

Capitolo 13: Set ordinati

introduzione

Il tipo di dati Set Sorted in Redis è una versione ordinata del tipo di dati Set. Un set ordinato di Redis consiste in una raccolta di membri unici. Ogni membro nel set ordinato può essere pensato come una coppia composta dal membro e da un punteggio. Il punteggio viene utilizzato per ordinare i membri all'interno del set in ordine crescente.

Sintassi

- Tasto ZADD [NX | XX] [CH] [INCR] membro del punteggio [membro del punteggio ...]
- Tasto ZCARD
- ZCOUNT chiave min max
- ZLEXCOUNT chiave min max

Osservazioni

La documentazione ufficiale per i set ordinati è disponibile sul sito Redis.io .

Gli insiemi ordinati sono a volte indicati come zsets. Se si utilizza il comando TYPE su una chiave set ordinata, verrà restituito il valore zset.

Examples

Aggiunta di elementi a un set ordinato

Redis fornisce il comando ZADD per aggiungere elementi a un set ordinato. La forma base del comando ZADD è specificare il set, l'elemento da aggiungere e il punteggio. Ad esempio, se volessi costruire un set ordinato del mio cibo preferito (dal meno al più), potrei usare uno dei seguenti:

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

o in alternativa:

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

La funzione ZADD funziona in modo molto simile alla funzione SADD impostata non ordinata. Il risultato del comando ZADD è il numero di elementi che sono stati aggiunti. Quindi, dopo aver creato il mio set come sopra, se avessi tentato nuovamente con la birra ZADD:

```
ZADD favs 4 beer
```

Avrei ottenuto un risultato 0, se avessi deciso che mi piace il cioccolato meglio della birra, potrei eseguire:

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

per aggiornare le mie preferenze, ma otterrei comunque un risultato di ritorno 0 poiché sia la birra che il cioccolato sono già nel set.

Conteggio degli oggetti in un set ordinato

Redis fornisce tre comandi per contare gli oggetti all'interno di un set ordinato: ZCARD, ZCOUNT, ZLEXCOUNT.

Il comando ZCARD è il test di base per la cardinalità di un set. (È analogo al comando SCARD per gli insiemi). ZCARD restituisce il conteggio dei membri di un set. Eseguendo il seguente codice per aggiungere elementi a un set:

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

ZCard in esecuzione:

```
zcard favs
```

restituisce un valore di 4.

I comandi ZCOUNT e ZLEXCOUNT consentono di contare un sottoinsieme degli elementi in un set ordinato in base a un intervallo di valori. ZCOUNT ti consente di contare gli elementi all'interno di un particolare intervallo di punteggi e ZLEXCOUNT ti consente di contare il numero di elementi all'interno di un particolare intervallo lessografico.

Usando il nostro set sopra:

```
zcount favs 2 5
```

restituire un 3, dal momento che ci sono tre elementi (pizza, cioccolato, birra) che hanno punteggi compresi tra 2 e 5 inclusi.

ZLEXCOUNT è progettato per funzionare con insiemi in cui ogni articolo ha lo stesso punteggio, costringendo e ordinando i nomi dei nomi. Se abbiamo creato un set come:

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

potremmo usare ZLEXCOUNT per ottenere il numero di elementi in particolare nell'intervallo lexografico (ciò avviene mediante confronto byte-saggio usando la funzione memcpy).

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

ritornerebbe 2, poiché due elementi (mela, birra) rientrano nel range mela (incluso) e cioccolato (esclusivo). Potremmo alternativamente fare entrambe le fini inclusive:

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

e ottieni il risultato 3.

Leggi Set ordinati online: https://riptutorial.com/it/redis/topic/9111/set-ordinati

Capitolo 14: Tasti di Redis

introduzione

Lo spazio delle chiavi di Redis può essere pensato come una tabella di hash o un dizionario che mappa le chiavi alle strutture di dati nel database.

Redis fornisce una vasta gamma di comandi che funzionano con le chiavi per gestire lo spazio delle chiavi, inclusa la possibilità di rimuovere le chiavi, ispezionare i metadati chiave, cercare le chiavi e modificare determinate proprietà delle chiavi.

Sintassi

- Modello CHIAVI
- Tasto PERSIST
- EXPIRE secondi chiave
- Timestamp del tasto EXPIREAT
- Chiave TTL
- I millisecondi chiave di PEXPIRE
- PexPIREAT key milliseconds-timestamp
- Tasto PTTL
- Tasto UNLINK [tasto ...]
- Tasto CANC [tasto ...]
- Cursore SCAN [pattern MATCH] [COUNT count]

Osservazioni

Per i *caratteri validi nei tasti Redis* , il manuale lo spiega completamente :

> Le chiavi Redis sono binary safe, questo significa che puoi usare qualsiasi sequenza binaria come una chiave, da una stringa come "foo" al contenuto di un file JPEG. Anche la stringa vuota è una chiave valida.
>
> Alcune altre regole sui tasti:
>
> Le chiavi molto lunghe non sono una buona idea, ad esempio una chiave di 1024 byte è una cattiva idea non solo per quanto riguarda la memoria, ma anche perché la ricerca della chiave nel set di dati può richiedere diversi costosi confronti-chiave. Anche quando il compito da svolgere è quello di abbinare l'esistenza di un grande valore, ricorrere all'hashing (per esempio con SHA1) è un'idea migliore, specialmente dal punto di vista della memoria e della larghezza di banda.
>
> Le chiavi molto corte spesso non sono una buona idea. Non ha molto senso scrivere "u1000flw" come chiave se invece è possibile scrivere "user: 1000: followers". Quest'ultimo è più leggibile e lo spazio aggiunto è minore rispetto allo spazio utilizzato

dall'oggetto chiave stesso e dall'oggetto valore. Mentre i tasti brevi ovviamente consumano un po 'meno di memoria, il tuo compito è trovare il giusto equilibrio.

Cerca di seguire uno schema. Ad esempio "object-type: id" è una buona idea, come in "user: 1000". I punti o i trattini vengono spesso utilizzati per i campi a più parole, come in "commento: 1234: reply.to" o "commento: 1234: risposta a".

La dimensione massima della chiave consentita è 512 MB.

Fare attenzione quando si utilizza il comando KEYS su un sistema di produzione, può causare seri problemi di prestazioni. Se è necessario eseguire una ricerca sullo spazio delle chiavi, i comandi SCAN rappresentano un'alternativa migliore.

Examples

Chiavi valide

Le chiavi Redis sono sicure per il codice binario, quindi letteralmente tutto può essere usato come chiave. Le uniche limitazioni sono che devono essere inferiori a 512 MB.

Esempi di chiavi valide:

```
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.
```

Tasti Redis non validi:

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

Schemi di denominazione delle chiavi

Per chiarezza e manutenibilità, si consiglia spesso di sviluppare un sistema o uno schema per nominare le proprie chiavi Redis. Ecco alcuni esempi di sistemi comuni e manutenibili per la denominazione delle chiavi:

```
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
```

Tieni presente che, sebbene consentito, i tasti più grandi utilizzano più memoria e comportano tempi di ricerca più lenti, pertanto l'utilizzo di una chiave da 500 MB potrebbe non essere una buona idea per le prestazioni. Un'idea migliore potrebbe essere quella di utilizzare un hash SHA-1, SHA-256 o MD5 di un oggetto binario di grandi dimensioni come chiave invece:

```
image/9517bb726d33efdc503a43582e6ea2eea309482b
image52e9df0577fca2ce022d4e8c86b1eccb070d37bef09dec36df2fabbfa7711f5c
```

Elenco di tutte le chiavi

È possibile elencare tutte le chiavi in un database Redis eseguendo i seguenti comandi da redis-cli:

```
KEYS *
```

Il parametro su KEYS è un'espressione di corrispondenza del modello glob-style. Esempi di pattern supporter includono:

```
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'utilizzo del comando KEYS * può avere effetti negativi sulle prestazioni, pertanto non è consigliato rispetto alle istanze di produzione. Utilizzare l'operazione SCAN per cercare le chiavi nel codice di produzione.

TTL e scadenza chiave

I valori di scadenza di una chiave possono essere gestiti da un utente al di fuori dei comandi di aggiornamento. Redis consente a un utente di determinare l'ora corrente da vivere (TTL) di una chiave utilizzando il comando TTL:

```
TTL key
```

Questo comando restituirà il TTL di una chiave in secondi o restituirà i valori speciali -1 o -2. A -1 indica che la chiave è persistente (non scadrà) e a -2 indica che la chiave non esiste.

Una chiave in scadenza può essere resa persistente usando il comando PERSIST:

```
PERSIST KEY
```

e una chiave persistente può essere fatta scadere usando il comando EXPIRE:

```
EXPIRE KEY seconds
```

La scadenza può anche essere utilizzata per modificare il TTL di una chiave esistente. In alternativa, è possibile utilizzare il comando EXPIREAT con un timestamp UNIX per impostare un tempo di scadenza.

Esistono versioni in millisecondi di comandi TTL, EXPIRE e EXPIREAT preceduti da un P.

Cancellare chiavi

Redis fornisce due funzioni per la rimozione delle chiavi dal database: del e unlink.

La funzione rimuove una o più chiavi dal database. Il comando del comando fa sì che Redis richieda immediatamente la memoria per la chiave cancellata sul thread corrente di esecuzione. Il tempo di esecuzione per del è proporzionale al numero di singoli elementi cancellati da tutte le chiavi.

La funzione di scollegamento funziona come il comando del comando, rimuove una o più chiavi dal database. Tuttavia, a differenza del comando del, qualsiasi memoria utilizzata da quelle chiavi viene recuperata in modo asincrono su un altro thread.

Scansione di Redis Keyspace

Redis fornisce il comando SCAN per scorrere le chiavi nel database che corrisponde a un particolare modello. Redis supporta la corrispondenza del modello stile glob nel comando SCAN.

Il comando SCAN fornisce un iteratore basato su cursore sullo spazio delle chiavi di Redis. La sequenza di chiamate iterativa a SCAN inizia con l'utente che effettua una chiamata con l'argomento del cursore impostato su 0. Il risultato di tale chiamata è un gruppo di voci e un cursore aggiornato che viene fornito alla successiva chiamata a SCAN. Questa iterazione continua fino a quando Redis non restituisce un cursore 0.

La seguente funzione di Python dimostra l'utilizzo di base di 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
```

Il comando SCAN è il metodo consigliato per cercare le chiavi nel database ed è consigliato tramite il comando `KEYS *` .

Leggi Tasti di Redis online: https://riptutorial.com/it/redis/topic/3916/tasti-di-redis

Titoli di coda

S. No	Capitoli	Contributors
1	Iniziare con redis	Ahamed Mustafa M, Alexander V., Aminadav, Community, Florian Hämmerle, Itamar Haber, Prashant Barve, RLaaa, Sagar Ranglani, sirin
2	Come connettersi a Redis in Java usando Jedis	Tague Griffith
3	Connessione a redis con Python	Gianluca D'Ardia, Tague Griffith, ystark
4	di riserva	odlp
5	Geo	Tague Griffith
6	Installazione e configurazione	Cristiana214, Gianluca D'Ardia, Sagar Ranglani
7	Lua Scripting	Tague Griffith
8	Pub / Sub	jerry, Tague Griffith
9	Redis List Datatype	Tague Griffith
10	Redis Persistence Storage	Aminadav, Tague Griffith
11	Redis Set Datatype	JonyD, Tague Griffith
12	Redis String datatype	Tague Griffith
13	Set ordinati	Tague Griffith
14	Tasti di Redis	Tague Griffith, Will