Complete Guide to Redis Commands

Last Updated : 19 Dec, 2024

Redis is an open-source, advanced key-value store and an apt solution for building high-performance, scalable web applications. Redis has three main peculiarities that set it apart:

- Redis holds its database entirely in the memory, using the disk only for persistence.
- Redis has a relatively rich set of data types compared to many key-value data stores.
- Redis can replicate data to any number of slaves.

Table of Content

1. KEY-VALUE OPERATIONS

1.1) SET:

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our Cookie Policy & Privacy Policy

Got It !

Syntax:

```
SET key value
```

Example:

```
SET user:123 name “John Doe” email “john@example.com” age 30
```

Time Complexity: O(1)

1.2) GET:

Retrieves the value associated with the specified key.

Syntax:

```
GET key
```

Example:

```
GET user:123:name
```

Time Complexity: O(1)

1.4) DEL:

Deletes one or more keys and their associated values.

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our Cookie Policy & Privacy Policy

```
DEL key [key …]
```

Example:

```
DEL username
```

Time Complexity: O(N)

1.4) EXISTS:

Checks if the specified key exists in the database.

Syntax:

```
EXISTS key
```

Example:

```
EXISTS user:123
```

Time Complexity: O(1)

1.5) TTL:

Get the remaining time to live of a key in seconds.

Syntax:

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our Cookie Policy & Privacy Policy

```
TTL key
```

Example:

```
TTL mykey
```

Time Complexity: O(1)

1.6) EXPIRE :

Set a key’s time to live in seconds.

Syntax:

```
EXPIRE key seconds
```

Example:

```
EXPIRE mykey 60
```

Time Complexity: O(1)

1.7) INCR

Increments the number stored at a key by one.

Syntax

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our Cookie Policy & Privacy Policy

```
INCR key
```

Time complexity: O(1)

Increments the number stored at key by one. If the key does not exist, it is set to 0 before performing the operation. An error is returned if the key contains a value of the wrong type or contains a string that can not be represented as integer. This operation is limited to 64 bit signed integers.

1.8) DECR

Decrements the number stored at key by one. If the key does not exist, it is set to 0 before performing the operation. An error is returned if the key contains a value of the wrong type or contains a string that can not be represented as integer. This operation is limited to 64 bit signed integers.

Syntax

```
DECR key
```

Time complexity: O(1)

1.9) APPEND

The amortized time complexity is O(1) assuming the appended value is small and the already present value is of any size, since the dynamic string library used by Redis will double the free space available on every reallocation.
If key already exists and is a string, this command appends the value at the end of the string. If key

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our Cookie Policy & Privacy Policy

Syntax

```
APPEND key value
```

Time complexity: O(1).

2. LISTS:

2.1) LPUSH

adds a new element to the head of a list;

Syntax

```
LPUSH key element [element ...]
```

Time complexity:O(1) for each element added, so O(N) to add N elements when the command is called with multiple arguments.

2.2) RPUSH

adds to the tail.

Syntax

```
RPUSH key element [element ...]
```

Time complexity:

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our Cookie Policy & Privacy Policy

2.3) LPOP

removes and returns an element from the head of a list.

Syntax

```
LPOP key [count]
```

Time complexity:

O(N) where N is the number of elements returned

2.4) RPOP

Does the same but from the tails of a list.

Syntax

```
RPOP key [count]
```

Time complexity:

O(N) where N is the number of elements returned

2.5) LLEN

returns the length of a list.

Syntax

```
LLEN key
```

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our Cookie Policy & Privacy Policy

2.5) LRANGE

Returns the specified elements of the list stored at key. The offsets start and stop are zero-based indexes, with 0 being the first element of the list (the head of the list), 1 being the next element and so on.

Syntax

```
LRANGE key start stop
```

Time complexity:
O(S+N) where S is the distance of start offset from HEAD for small lists, from nearest end (HEAD or TAIL) for large lists; and N is the number of elements in the specified range.

3. SETS:

3.1) “sadd” command:

Creates a set, and adds element to it.

Syntax:

```
sadd set_name set_element
```

Example:

```
sadd SocialMedia Facebook Twitter WhatsApp.
```

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our Cookie Policy & Privacy Policy

Explanation:

We created a set namely “SocialMedia” and added 3 unique elements into it as “Facebook”, “Twitter”, “WhatsApp”.

3.2) “smembers” command:

Shows all the elements, present in that set.

Syntax:

```
smembers set_name
```

Example:

```
smembers SocialMedia
```

Explanation:

Previously, we stored “Facebook”, “Twitter”, “WhatsApp” in SocialMedia set. Hence, it’s displaying these elements.

3.3) “scard” command:

Shows no. of elements, present in that set.

Syntax:

```
scard set_name
```

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our Cookie Policy & Privacy Policy

```
scard SocialMedia
```

Explanation:

There ae 3 elements in this set, hence its showing 3 as output

3.4) “sismember” command:

Checks if that element exists in that set, if yes then returns 1

Syntax:

```
sismember set_name set_element
```

Example:

```
sismember SocialMedia Twitter
```

Explanation: Returned 1, “Twitter” is present in SocialMedia set.

3.5) “sdiff” command:

Shows difference of two sets by displaying those elements, elements that are in set 1 butt not in set 2.

Syntax:

```
sdiff set_name1 set_name2
```

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our Cookie Policy & Privacy Policy

```
sdiff FavSub1 FavSub2
```

Explanation:

FavSub1 and FavSub2 are two sets that contain favorite subjects of User1 and User2, respectively. Elements of FavSub1 and FavSub2 are {Computer Science, Data Science} and {Computer Science, Math's}. The ouput of this command is showing diffrence of two sets as their elements that is the elements that are in FavSub1 but not in FavSub2. Hence, output is Data Science.

3.6) "sdiffstore" command:

Stores elements that differ in two sets, elements that are in set1 but not in set2 in a new set
Syntax:

```
sdiffstore set_name3 set_name1 set_name2
```

Example:

```
sdiffstore NewSet FavSub1 FavSub2
```

Explanation:

As the elements that differ in FavSub1 and FavSub2 is only Data Science. Hence, we made a new set namely "NewSet" that stored this element.

3.7) "sinter" command:

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our Cookie Policy & Privacy Policy

```
sinter set_name1 set_name2
```

Example:

```
sinter FavSub1 FavSub2
```

Explanation:

Displaying Computer Science, as its the only common element present in both sets

3.8) “sinterstore” command:

Stores elements that are present in both sets in a new set.

Syntax:

```
sinterstore set_name4 set_name1 set_name2
```

Example:

```
sinterstore SamePinch FavSub1 FavSub2
```

Explanation:

SamePinch is a new set created that stores the elements that are common in FavSub1 and FavSub2 that is Computer Science

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our Cookie Policy & Privacy Policy

Shows all the unique elements that are there overall including those 2 sets as union of sets

Syntax:

```
sunion set_name1 set_name2
```

Example:

```
sunion FavSub1 FavSub2
```

Explanation: Displaying Computer Science, Maths and Data Science as these are the unique elements in whole as union of sets

3.10) “sunionstore” command:

Stores elements that are union of two sets in a new set

Syntax:

```
sunionstore set_name5 set_name1 set_name2
```

Example:

```
sunionstore Sub FavSub1 FavSub2
```

Explanation: Sub is a new set created that stores all the unique elements of whole in these 2 sets.

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our Cookie Policy & Privacy Policy

Syntax:

```
srem set_name element
```

Example:

```
srem Sub Math’s
```

Explanation:

Removed Math’s element from the set “Sub”, hence output after executing srem commands is Data Science and Computer Science.

4. HASHES:

4.1) HSET (Hash Set):

Sets the value of a field in a Hash.
If the field does not exist, it creates the field and assigns the value.

Syntax:

```
HSET <key> <field> <value>
```

4.2) HGET (Hash Get):

Retrieves the value of a field in a Hash.

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our Cookie Policy & Privacy Policy

```
HGET <key> <field>
```

4.3) HDEL (Hash Delete):

Deletes one or more fields from a Hash.

Syntax:

```
HDEL <key> <field1> [<field2> … <fieldN>]
```

Example for HDEL:

Input:

```
HDEL user:123 age
HGET user:123 age
```

Output:

```
(nil)
```

4.4) HGETALL (Hash Get All):

Retrieves all fields and values of a Hash.

Syntax:

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our Cookie Policy & Privacy Policy

```
HGETALL <key>
```

Example for HGETALL:

Input:

```
HSET user:789 name “Charlie” country “USA”
HGETALL user:789
```

Output:

```
1) “name”
2) “Charlie”
3) “country”
4) “USA”
```

4.5) HKEYS (Hash Keys):

The HKEYS command retrieves all the field names in a hash.

Syntax:

```
HKEYS <key>
```

Example for HKEYS:

Input:

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our Cookie Policy & Privacy Policy

```
HSET user:2 username bob email bob@example.com age 25
HKEYS user:2
```

Output:

```
1) “username”
2) “email”
3) “age”
```

4.6) HVALS (Hash Values):

The HVALS command retrieves all the values in a hash.

Syntax:

```
HVALS <key>
```

Example for HVALS:

Input:

```
HSET preferences:456 theme “dark” language “en”
HVALS preferences:456
```

Output:

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our Cookie Policy & Privacy Policy

```
2) “en”
```

5. PUB/SUB:

Channels are the communication pathways in Redis Pub/Sub. Messages are published to specific channels, and subscribers listen to messages on one or more channels. Channels are identified by names, e.g., “news,” “chatroom,” “events,” etc. When a message is published on a channel, all subscribers in that channel receive the message.

5.1) PUBLISH

Publishers use the PUBLISH command to send messages to channels. The

Syntax:

```
PUBLISH channel message
```

- ***channel:*** The name of the channel to which the message will be sent.
- ***message:*** The actual message content.

Example:

```
PUBLISH chat:general “Hello, everyone!”
```

Note:

In this example, the message “Hello, everyone!” is sent to the “chat:general” channel.

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our Cookie Policy & Privacy Policy

5.2) SUBSCRIBE

Subscribers use the SUBSCRIBE command to start listening to one or more channels. The

Syntax:

```
SUBSCRIBE channel [channel …]
```

channel: The name of the channel to subscriber is going to subscribe.

Example:

```
SUBSCRIBE chat:general
```

Note:

> *In this example, the subscriber starts listening to the “chat:general” channel.*

5.3) UNSUBSCRIBE

Subscribers can unsubscribe from specific channels using the UNSUBSCRIBE command. The

Syntax:

```
UNSUBSCRIBE channel [channel …]
```

Example:

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our Cookie Policy & Privacy Policy

```
UNSUBSCRIBE chat:general
```

Note:

> *In this example, the subscriber stops listening to the “chat:general” channel.*

6. OTHER COMMANDS:

6.1) SELECT

Select the Redis logical database having the specified zero-based numeric index. New connections always use the database 0.
Selectable Redis databases are a form of namespacing: all databases are still persisted in the same RDB / AOF file.

Syntax

```
SELECT index
```

Time complexity: O(1)

6.2) FLUSHDB

Delete all the keys of the currently selected DB. This command never fails.
By default, FLUSHDB will synchronously flush all keys from the database. Starting with Redis 6.2, setting the lazyfree-lazy-user-flush configuration directive to "yes" changes the default flush mode

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our Cookie Policy & Privacy Policy

```
FLUSHDB [ASYNC | SYNC]
```

Time complexity:
O(N) where N is the number of keys in the selected database

6.3) SAVE

The SAVE commands performs a synchronous save of the dataset producing a point in time snapshot of all the data inside the Redis instance, in the form of an RDB file.

Syntax

```
SAVE
```

Time complexity: O(N) where N is the total number of keys in all databases

6.4) BGSAVE

Save the DB in background.

Syntax

```
BGSAVE [SCHEDULE]
```

Time complexity: O(1)

Redis offers us various Sets commands, and above-mentioned ones are most important and

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our Cookie Policy & Privacy Policy

requirements, in that case you can refer offical documentation of Redis. But for majority of your work with Sets in Redis, this article in itself is more than enough.

Comment More info Advertise with us

Next Article

Complete Guide of Redis Scripting

Similar Reads

Introduction to Redis

Redis is an in-memory data structure that is used for faster access to data. It is used to store data that needs to be accessed frequently and fast. It is not used for storing large amounts of data. If you want to store and retrieve large amounts of data you need to use a...

5 min read

Redis Basics

Redis and its role in System Design

Redis is an open-source, in-memory data structure store used as a database, cache, and message broker. It is widely used for its fast performance, flexibility, and ease of use. What is RedisRedis Data TypesBenefits of using RedisWorking Architecture of Redis1. Single...

12 min read

How does Redis store data?

Redis is an in-memory data store. It stores data primarily in RAM, allowing extremely fast read and write operations. The data is organized using key-value pairs, where keys are unique identifiers, and values can be of different types, including strings, lists, sets, sort

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our Cookie Policy & Privacy Policy

Complete Tutorial of Configuration in Redis

Redis configuration involves setting various options and parameters that govern the behavior of the Redis server. Configuration settings impact aspects such as networking, memory usage, persistence, security, and more. The Redis configuration file, usually named redis.con...

8 min read

Redis Cache

As we all know caching refers to storing frequently used or dealt-up data in temporary high-speed storage to reduce the latency of a system. So we do the same when happens inside a Redis cluster. Therefore, Redis Cache supercharges application performance by...

7 min read

Redis Data Structures

Complete Guide to Redis Lists

When it comes to data structures, Redis offers a wide range of options to efficiently manage and manipulate your data and one of them is list. A list is an ordered collection of values associated with a unique index. Unlike arrays, Redis lists are not limited to a fixed size and ca...

7 min read

Complete Guide on Redis Strings

Redis String is a sequence of bytes that can store a sequence of bytes, including text, Object, and binary arrays. which can store a maximum of 512 megabytes in one string. Redis String can also be used like a Redis Key for mapping a string to another string. String...

6 min read

A Complete Guide to Redis Keys

In Redis, keys are the fundamental data structure used to identify, store, and retrieve data. Each key in Redis maps to a corresponding value, and Redis supports various data types for values like strings, hashes, lists, sets, sorted sets, and more. Redis keys are used to...

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our Cookie Policy & Privacy Policy

GeeksforGeeks
Sanchhaya Education Private Limited

Corporate & Communications Address:
A-143, 7th Floor, Sovereign Corporate
Tower, Sector- 136, Noida, Uttar Pradesh
(201305)

Registered Address:
K 061, Tower K, Gulshan Vivante
Apartment, Sector 137, Noida, Gautam
Buddh Nagar, Uttar Pradesh, 201305

GET IT ON Google Play
Download on the App Store

Advertise with us

Company
- About Us
- Legal
- Privacy Policy
- In Media
- Contact Us
- Advertise with us
- GFG Corporate Solution

Languages
- Python
- Java
- C++
- PHP
- GoLang
- SQL
- R Language

DSA
- Data Structures
- Algorithms
- DSA for Beginners
- Basic DSA Problems
- DSA Roadmap
- Top 100 DSA Interview Problems

Data Science & ML
- Data Science With Python
- Data Science For Beginner
- Machine Learning
- ML Maths
- Data Visualisation

Web Technologies
- HTML
- CSS
- JavaScript
- TypeScript
- ReactJS
- NextJS
- Bootstrap

Python Tutorial
- Python Programming Examples
- Python Projects
- Python Tkinter
- Web Scraping
- OpenCV Tutorial
- Python Interview

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our Cookie Policy & Privacy Policy

GeeksforGeeks Community

Deep Learning

Computer Science

- Operating Systems
- Computer Network
- Database Management System
- Software Engineering
- Digital Logic Design
- Engineering Maths
- Software Development
- Software Testing

DevOps

- Git
- Linux
- AWS
- Docker
- Kubernetes
- Azure
- GCP
- DevOps Roadmap

System Design

- High Level Design
- Low Level Design
- UML Diagrams
- Interview Guide
- Design Patterns
- OOAD
- System Design Bootcamp
- Interview Questions

Inteview Preparation

- Competitive Programming
- Top DS or Algo for CP
- Company-Wise Recruitment Process
- Company-Wise Preparation
- Aptitude Preparation
- Puzzles

School Subjects

- Mathematics
- Physics
- Chemistry
- Biology
- Social Science
- English Grammar
- Commerce
- World GK

GeeksforGeeks Videos

- DSA
- Python
- Java
- C++
- Web Development
- Data Science
- CS Subjects

@GeeksforGeeks, Sanchhaya Education Private Limited, All rights reserved

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our Cookie Policy & Privacy Policy