

NoSQL Databases

The Redis key-value store

Laszlo Szathmary
University of Debrecen
Faculty of Informatics

(last update: 2018-12-06 [yyyy-mm-dd])

2018-2019, 1st semester

Redis

<http://redis.io>

Redis is an open-source, key-value store, where data are stored in the memory. Among the key-value store systems, Redis is the most well-known.

Some of its properties:

1. implemented in C
2. first version was released in 2009
3. supports multiple platforms
4. there are interfaces to lots of programming languages (e.g. C, C++, C#, Erlang, Go, Haskell, Java, Lua, Perl, PHP, Python, R, Ruby, Scala, etc.)
5. easy to learn

It is developed actively. Available under a BSD license.

In a very simplified way, Redis is a hash map, where you assign values to keys. However, and here is where Redis differs from other key-value stores, in Redis a value doesn't have to be a string!

The value can be:

- a string
- a list
- a set
- an ordered set (the elements are ordered by a *score*)
- a hash

On the server side, high level operations are available, e.g. sorting a list, set operations (union, intersection, difference), etc.

Persistence

Redis stores its whole data in the memory. However:

1. Upon quit, the database can be saved to disk.
2. Upon start, the saved database can be loaded.

Furthermore, the system performs automatic saves from time to time (*snapshotting*). The interval between saves can be set. Snapshotting can be complete or incremental.

Performance

It was implemented in C (very fast); works in the memory (low number of I/O operations).

It uses one single process, and it uses one thread only (*single threaded*)
=> each operation is atomic.

Transaction

Every operation is atomic. If we want to group several operations in one atomic group, it can be done with transactions.

Installation from source

<http://redis.io/download>

```
# find the newest version
$ wget http://download.redis.io/releases/redis-2.8.19.tar.gz
$ tar xvzf redis-2.8.19.tar.gz
$ cd redis-2.8.19
$ make
$ make test
# optional, under Ubuntu it installs to /usr/local/bin
$ sudo make install
```

Installation with a package manager #1

Installation under **Ubuntu** Linux:

```
$ sudo apt install redis
```

Among others, it will install the „redis-server“ and „redis-tools“ packages.

„redis-server“ contains the server

„redis-tools“ contains the command-line client

Installation with a package manager #2

Installation under **Manjaro** Linux :

```
$ sudo pacman -S redis
```

Manjaro is based on Arch Linux, and it's getting more and more popular. Similarly to Ubuntu, it's easy to install, easy to use. Manjaro is a rolling-release distro.

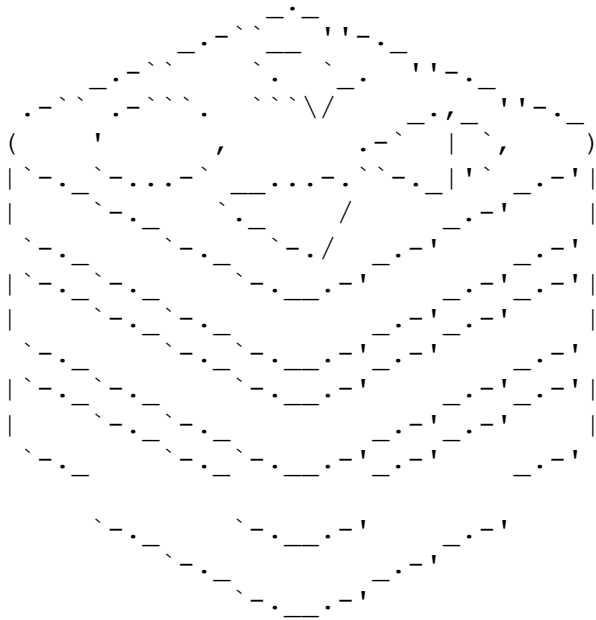
New versions of packages get into Manjaro repositories very quickly.

Here, it's enough to install just one package, which includes both the server and the command-line client.

Starting Redis manually

Manually, the server can be started with the command `redis-server`:

```
$ redis-server
[11650] 29 Sep 13:44:31.595 # Warning: no config file specified, using the default config. In
order to specify a config file use redis-server /path/to/redis.conf
[11650] 29 Sep 13:44:31.596 # Unable to set the max number of files limit to 10032 (Operation
not permitted), setting the max clients configuration to 3984.
```



Redis 2.6.16 (00000000/0) 64 bit

← version

Running in stand alone mode

Port: 6379

PID: 11650

← default port

<http://redis.io>

```
[11650] 29 Sep 13:44:31.596 # Server started, Redis version 2.6.16
[11650] 29 Sep 13:44:31.599 * DB loaded from disk: 0.003 seconds
[11650] 29 Sep 13:44:31.599 * The server is now ready to accept connections on port 6379
```


Starting Redis automatically #1

under Ubuntu

Under Ubuntu, the installer does everything for us, thus Redis will start at the next boot.

If we installed Redis from source, then here is a guide that explains how to register the server to start at each boot:

<https://www.digitalocean.com/community/articles/how-to-install-and-use-redis>

The easiest way to check if the server is running is to try to connect to it with the command-line shell:

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

If we don't get any error, then the server is running.

Starting Redis automatically #2

under Manjaro

To enable starting the server at boot, execute this command:

```
$ systemctl enable redis.service
```

With the following command you can start the server. If you restart your machine, this command won't be necessary:

```
$ systemctl start redis.service
```

Use the command „redis-cli” to check if the server is running:

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

Data structures

Redis can use **five** different data structures. Out of these five, only one is considered a typical key-value structure.

Key: arbitrary string, e.g.: `users:jabba`

The colon has no special meaning. It's a common practice to use it as a delimiter.

Value: the value assigned to the key

In most cases, Redis treats it as a series of bytes and doesn't care about its content.

```
$ redis-cli
redis 127.0.0.1:6379> set users:jabba "Jabba Laci"
OK
redis 127.0.0.1:6379> get users
(nil)
redis 127.0.0.1:6379> get users:jabba
"Jabba Laci"
redis 127.0.0.1:6379>
```



command-line client

Data structures

„The key is everything, the value is nothing.“

That is, you cannot write a query on values. For ex. a JSON value is just a normal string for Redis.

Redis is great for some tasks, but it doesn't provide a general solution for every problem. Redis doesn't want to replace RDBM systems.

However, it can **complement** RDBM systems.

Every command in Redis belongs to a specific data structure. The available commands are listed here: <http://redis.io/commands> .

1. string
2. hash
3. list
4. set
5. ordered set



see examples later


Web admin interface

Redis Commander is a Node.js web application, somewhat similar to PHPMysqlAdmin.

<http://nearinfinity.github.io/redis-commander/>

Installation

```
$ sudo npm install -g redis-commander
$ redis-commander
path.existsSync is now called `fs.existsSync`.
listening on 8081
Redis Connection 127.0.0.1:6379 Using Redis DB #0
```

 **redis Commander**

Refresh

Commands

Add Server

127.0.0.1:6379:0

users:* (1)

abc jabba

Add New Key...

Disconnect

Name	Value
# server	
Redis version	2.6.16
Redis git sha1	00000000
Redis git dirty	0
Redis mode	standalone
Os	Linux 3.8.0-31-generic x86_64
Arch bits	64
Multiplexing api	epoll
Gcc version	4.7.3
Process	11650
Run	6d06dab587abe60085d9f5c5e38e378336bea373
Tcp port	6379
Uptime in seconds	1161

127.0.0.1:6379:0

Current Instance: 127.0.0.1:6379:0

redis>

Redis Commander's web interface (localhost:8081)

GUI admin interface

Another admin interface is Redis Desktop Manager, which is a native desktop application written in C++.

Installation

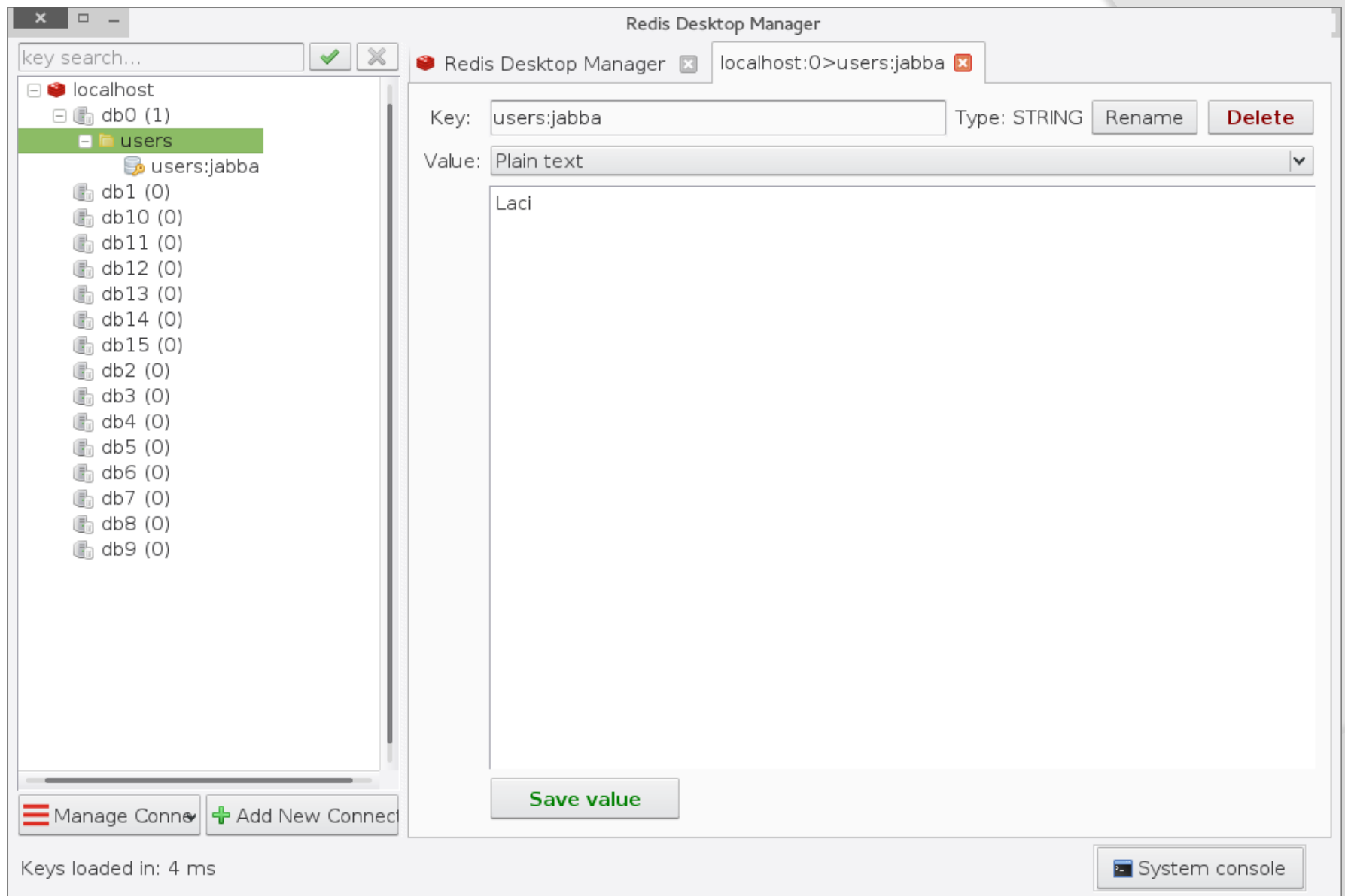
<http://redisdesktop.com/>

```
$ sudo snap install redis-desktop-manager
```

How to launch it:

```
$ redis-desktop-manager.rdm
```

Make sure that `/snap/bin` is in your PATH.



Redis Desktop Manager's interface

Using Redis from applications

Redis can be used from various programming languages. Let's see some Python examples.

First, we need to install the package `redis` :

```
sudo pip3 install redis -U
```

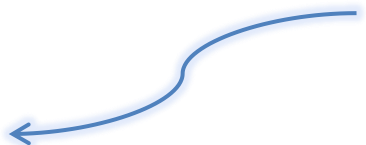
Verify if installation was successful:

```
[08:34:09] ~ $ python3
Python 3.5.2 (default, Nov 23 2017, 16:37:01)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import redis
>>> █
```

All right, import didn't raise any exception.

Connecting to the server

```
1  #!/usr/bin/env python
2
3  import sys
4  import redis
5
6  r = redis.Redis("localhost", 6379)
7  #r = redis.Redis()    # use default values
8
9  def check_server():
10     try:
11         r.info()
12     except redis.exceptions.ConnectionError:
13         print >>sys.stderr, "Error: cannot connect to redis server. Is the server running?"
14         sys.exit(1)
15
16
17 def main():
18     check_server()
19     print "The server is up and running."
20
21 #####
22
23 if __name__ == "__main__":
24     main()
```



Examples #1

```
>>> import redis
>>> r = redis.Redis()    # default: localhost, port 6379
>>> r.set("name", "jabba")
True
>>> r.get("name")
'jabba'

# the list is called "test"
# rpush: right push, i.e. put an element on its right side (tail)
>>> r.rpush("test", 24)
1L
>>> r.rpush("test", 67)
2L
>>> r.rpush("test", 9)
3L
# list all the elements (-1 is the index of the last element)
>>> r.lrange("test", 0, -1)
['24', '67', '9']
# number of elements
>>> r.llen("test")
3
# delete the list if you don't need it anymore
>>> r.delete("test")
1
```

Examples #2

```
>>> import redis
>>> r = redis.Redis()    # default: localhost, port 6379
>>> r.incr("goku:powerlevel")
1
>>> r.incr("goku:powerlevel")
2
```

It's a common task to store some counter in Redis. Since every operation in Redis is atomic, and Redis uses just one thread, Redis can be used by several processes concurrently.

A producer / consumer application in Redis is trivial.

Examples #3

```
>>> import redis
>>> r = redis.Redis()
>>> r.set("name", "jabba")
True
>>> r.get("name")
'jabba'
>>> r.expire("name", 10)
True
>>> r.ttl("name")
6L
>>> r.ttl("name")
4L
>>> r.ttl("name")
>>> r.get("name")
>>> █
```

the key / value pair is to be
deleted in 10 seconds
(automatically)

ttl: time to live

Unix timestamps are also accepted, e.g. `r.expireat("name", 1356933600)`

Redis is an ideal choice for caching data.

Alternatives

<http://www.memcached.org/>

The most well-known alternative is *Memcached*.

Memcached provides a large hash table, where one can store key / value pairs. Usually, dynamic web sites use it for caching data and objects, reducing the time of reading databases.

Caching example:

```
function get_foo(foo_id)
  foo = memcached_get("foo:" . foo_id)
  return foo if defined foo

  foo = fetch_foo_from_database(foo_id)
  memcached_set("foo:" . foo_id, foo)
  return foo
end
```

First, we try to read the element from the cache.

If not found, then read it from the database and put it in the cache.

When called again, the data can be returned from the cache.

Links

- Redis HQ
<http://redis.io>
- Redis commands
<http://redis.io/commands>
- Karl Seguin: The Little Redis Book
<http://openmymind.net/2012/1/23/The-Little-Redis-Book/>
- Getting Started: Redis and Python
<http://degizmo.com/2010/03/22/getting-started-redis-and-python/>
- Redis Python client: redis-py
<https://github.com/andymccurdy/redis-py>
- Python wrapper for Redis datatypes
https://github.com/Doist/redis_wrap