COSC 416 - Special Topics in Databases
Assignment 7 - Redis

Connecting
The redis database is located here:
Host: gpu1.ddl.ok.ubc.ca
Port: 50030

Installation
- Installing Redis: http://redis.io/topics/quickstart

Documentation
- Main Documentation: http://redis.io/documentation
- Commands: http://redis.io/commands
- Try Redis (scripting not supported): http://try.redis.io

Command Line Usage
$ redis-cli --help

Using your Database
Redis supports multiple databases via the SELECT command. Each group will be assigned a database number that they will use to select their database after each connection. Database 0 will be kept open for everyone if they want to try interfacing their chat clients together in part II of the assignment.

$ redis-cli -n 1 GET mykey

To access redis on GPU1 use the command:
/srv/redis/redis-2.6.10/src/redis-cli

Part I (10 marks)

Task #1 (1 mark)

Add your name to the database keyed to your student ID.
HINTS: http://redis.io/commands#string

Task #2 (1 mark)

- Add a list of 5 databases we're going to cover in this course using "416" as the key, in the order they are covered in class.
- Trim MongoDB from the list using the LTRIM command then remove your least favorite

db from the list (You should now have a list of size 3)
HINTS: http://redis.io/commands#list

Task #3 (1 mark)

Make a new set called C that contains the common element in these 2 sets.
A: “a, b, c”
B: “c, d, e, f”
HINTS: http://redis.io/commands#set

Task #4 (1 mark)

- Create a sorted set of size 5 of your favorite databases then promote the third element in your list to becoming the top of your list.
- Retrieve your list along with the scores.

HINTS: http://redis.io/commands#sorted_set

Task #5 (1 mark)

- Create a hash table that stores a user profile with
  - ID
  - First Name
  - Last Name
  - birthday
  - email
- Delete the email then return all values.

HINTS: http://redis.io/commands#hash

Task #6 (1 mark)

- Subscribe to the “COSC416A” channel
- Publish the message “Hello World” in the COSC416A channel

(Use 2 different Terminals or get a friend to send the message)
HINTS: http://redis.io/commands#pubsub

Task #7 (1 mark)

- Make Redis delete your name that you created in Task #1 in 10 seconds.
- Remove the expiration timer

HINTS: http://redis.io/commands#generic

Task #8 (3 mark)

- Make the database delete your name that you created in Task #1 in 10 ***milli***seconds
- **Remove the expiration timer before it gets deleted**

HINTS: http://redis.io/topics/transactions

Part II (10 marks)

Write a Lua script that will increment a value without using the INCR command, and returns the updated value. The key must be passed in by the client (not hard coded into the script).

In Lua you can call Redis commands with the redis.call function like this:

redis.call("set", "mykey", "myval")
local value = redis.call("get", "mykey") -- always declare variables as 'local'
return value – will return "myval" to client

It should function like this:
SET mykey 1
> "1"
EVAL <your script> mykey
> "2"

BONUS: Modify the script so that it accepts multiple keys and values, and increments each key by the corresponding value.

It should function like this:
MSET mykey1 1 mykey2 2
EVAL <your script> 2 mykey1 mykey2 1 2
> "mykey1"
> "2"
> "mykey2"
> "4"

HINTS:
http://redis.io/commands#scripting
http://redis.io/commands/eval
redis-cli --eval script.lua mykey1 mykey2 , 1 2

NOTE: Remember that Redis returns everything as strings, including script arguments. In your script, you may need to use the **tonumber()** Lua function to convert a string to a number.

Part III (30 marks)

PHP Template: https://www.dropbox.com/s/w9riji6zv6apygz/RedisChat-assignment.zip
Java template: https://github.com/smithbower/JRedisChatAssignment

NOTE: For the Java example, if you are connecting to gpu1.ddl.ok.ubc.ca, you must use port

50030. You must change the Jedis connection constructors [Jedis blah = new Jedis(host)] to contain the port, i.e. [Jedis blah = new Jedis(host, 50030)].

Create a basic Redis chat client using Redis Publish/Subscribe. The client should be able to do the following things:

- A user should be able to identify him or herself to the server using some command.
- A user can send a chat message, which is heard by everyone.
- A user can join and leave a channel.
- A user can send a chat message to a specific channel, which is heard only by users who have joined that channel.
- A user can send a chat message to a specific person, which is heard only by that person.
- A user can ask the server for information about another user.

As a guide (and if you want to interface with other chat clients), you should implement the following protocol. Both a PHP web-template and a Java client have been written for you (you just need to plug in the required Redis commands). You are free to write your own implementation however.

Each user has his or her own hash object containing the following fields:
(key) user:<name>

- **(string) name**
- **(integer) age**
- **(string) sex**
- **(string) location**

Each user also has a set object which contains the channels he or she is subscribed to.
(key) channels:<name>

- **(string) channel:416**
- **(string) channel:cosc234**
- **(string) channel:all**
- **…**

Each message stored as a serialized JSON object. Example:

```
{
        “name”: “paul”,
        “channel”: “channel:416”,
        “message”: “fml”
}
```

The following commands should be made available:

- **/me** *[name] [age] [sex] [location]* Identifies the user to the server
- **/join** *[channel]* Subscribes the user to the given channel
- **/chat** *[channel] [message]* Broadcasts a message on the given channel. If no channel is specified, defaults to “all”.
- **/tell** *[user] [message]* Sends a message to a user’s private channel (named after them).
- **/leave** *[channel]* Leaves a given channel.
- **/whois** *[user]* Gets server information on a given user.