redis-c-examples Documentation

Release 0.1.0

Vladimir Botka

Sep 17, 2019

Contents:

1 Introduction **1**

2 Examples **3**

2.1 Create frequency graph from a log 3

2.2 List 10 most used words in a text 6

3 Indices and tables **9**

CHAPTER 1

Introduction

Redis is popular in-memory data structure store, used as a database, cache and message broker. Hiredis is a minimalistic C client library for the Redis database documented at RedisLabs .

Intention of this documentation is to create examples how to use Hiredis.

Ultimate guide to Redis commands is available at redis.io.

CHAPTER 2

Examples

2.1 Create frequency graph from a log

Task: Read /var/log/dpkg.log and create a graph to visualize how often packages are installed, upgraded and removed.

Solution: The loop (91) calls function *read_log* which reads the log line by line (33) and splits the fields date and time in minutes (35,39) and status (40,42). Third field of the log *status* is status of the dpkg operation(install, upgrade, remove ...). Command *ZINCRBY* (43) increments by 1 the score of *date* in the key *status*. As a result the database contains keys(install, upgrade, remove ...) and associated lists of *dates* sorted by score. Next loop (97) calls the function *write_csv* with first 10 keys. As a result *status.csv* files are created in the current directory with the *(date;score)* pairs.

[create-graph-01.c]

```
1  /*
2   * Task: Read /var/log/dpkg.log and create a graph to visualize how
3   * often packages are installed, upgraded and removed.
4   * Tested with: gcc 7.2, hiredis 0.13 and redis 4.0.1
5   */
6
7  #include <assert.h>
8  #include <stdio.h>
9  #include <stdlib.h>
10 #include <string.h>
11 #include <hiredis.h>
12
13 #define RCMD(context, ...) {                    \
14   freeReplyObject(reply);                       \
15   reply = redisCommand(context, __VA_ARGS__); \
16   assert(reply != NULL);                        \
17   }
18
19 const char LOG_SEPARATOR = ' ';
20 const char CSV_SEPARATOR = ';';
```

(continues on next page)

(continued from previous page)

```
const char *LOG_FILES[] = { "dpkg.log", };

redisContext *c;

/* This function reads log_file and puts the status into the database.*/
int read_log(const char *log_file, redisContext *c) {
  redisReply *reply=NULL;
  FILE *fp;
  char line[256];
  char *p, *date, *status;

  fp = fopen(log_file, "r");
  while (fgets(line, sizeof(line), fp) != NULL) {
    p = line;
    date = p++;
    while (p[0] != LOG_SEPARATOR) p++;
    p++;
    while (p[0] != LOG_SEPARATOR) p++;
    *(p-3) = '\0';
    status = ++p;
    while (p[0] != LOG_SEPARATOR) p++;
    *p = '\0';
    RCMD(c, "ZINCRBY %s 1 \"%s\"", status, date);
  }
  fclose(fp);
  return(0);
}

/* This function reads the database and writes the status CSV file. */
int write_csv(char *status, redisContext *c) {
  redisReply *reply=NULL;
  FILE *fp;
  int i,n;
  char filename[256];

  strcpy(filename, status);
  strcat(filename, ".csv");
  fp = fopen(filename, "w");
  RCMD(c, "ZRANGE %s 0 -1 WITHSCORES", status);
  n = reply->elements-1;
  for (i=0; i<n; i=i+2) {
    fprintf(fp, "%s %c %s\n", reply->element[i]->str, \
            CSV_SEPARATOR, \
            reply->element[i+1]->str);
  }
  fclose(fp);
  return(0);
}


int main(int argc, char **argv) {
  redisReply *reply=NULL;
  int i, n;
  char *status;

  c = redisConnect("localhost", 6379);
  if (c == NULL || c->err) {
```

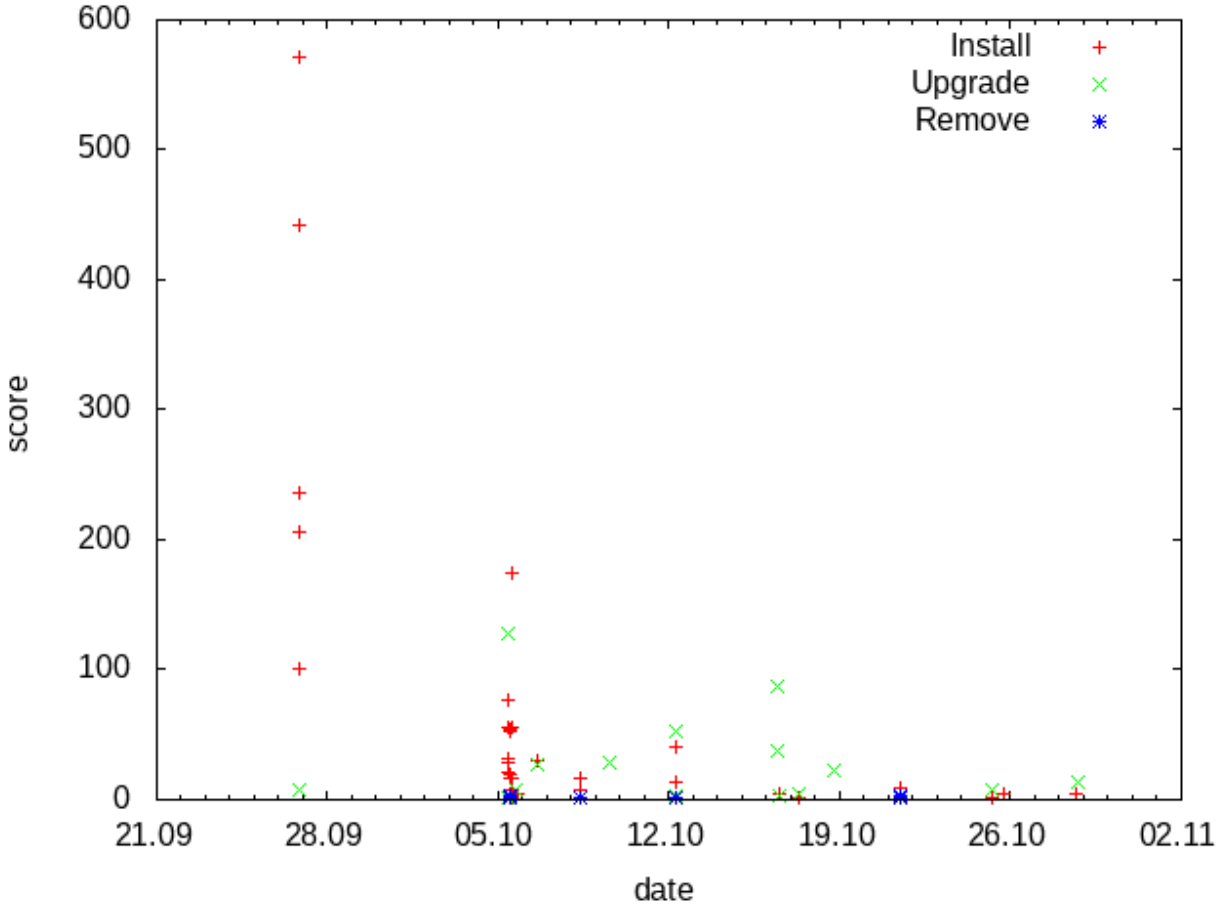
(continues on next page)

(continued from previous page)

```
78      if (c) {
79        printf("Connection error: %s\n", c->errstr);
80        redisFree(c);
81      } else {
82        printf("Connection error: can't allocate redis context\n");
83      }
84      exit(1);
85    }
86
87    RCMD(c, "SELECT 0");
88    RCMD(c, "FLUSHDB");
89
90    n = sizeof(LOG_FILES)/sizeof(LOG_FILES[0]);
91    for (i = 0; i < n; i++) {
92      read_log(LOG_FILES[i], c);
93    }
94
95    RCMD(c, "SCAN 0 COUNT 10");
96    n = reply->element[1]->elements;
97    for (i=0; i<n; i++) {
98      status = reply->element[1]->element[i]->str;
99      write_csv(status, c);
100   }
101
102   redisFree(c);
103   return(0);
104 }
```

Result: The *status.csv* files can be used to create a graph with *gnuplot*.

[create-graph-01.gnuplot]



2.2 List 10 most used words in a text

Task: Read text from a file and list 10 most frequently used words in it.

Solution: Let's use article about Redis at wikipedia.org as a text.

[create-topchart-text.bash]

```
#!/bin/bash
lynx -dump -nolist https://en.wikipedia.org/wiki/Redis > redis.txt
```

Command *ZINCRBY* (50) increments by 1 the score of *word* in the key *topchart* and *ZRANGE* (57) returns top 10 words with scores.

[create-topchart.c]

```
1  /*
2   * Task: Read text from a file and list 10 most frequently used words
3   * in it.
4   * Tested with: gcc 7.2, hiredis 0.13 and redis 4.0.1
5   */
6
7  #include <assert.h>
8  #include <stdio.h>
```

(continues on next page)

(continued from previous page)

```c
#include <stdlib.h>
#include <string.h>
#include <hiredis.h>

#define RCMD(context, ...) {                    \
  freeReplyObject(reply);                       \
  reply = redisCommand(context, __VA_ARGS__); \
  assert(reply != NULL);                        \
  }

const int SIZEOFLINE = 1023;
const char* DELIMETERS = " ;,.-!?";
const char* filename = "redis.txt";

int main(int argc, char **argv) {
  redisContext *c;
  redisReply *reply=NULL;
  FILE *fp;
  char line[SIZEOFLINE+1];
  char* word;
  int i, n;

  c = redisConnect("localhost", 6379);
  if (c == NULL || c->err) {
    if (c) {
      printf("Connection error: %s\n", c->errstr);
      redisFree(c);
    } else {
      printf("Connection error: can't allocate redis context\n");
    }
    exit(1);
  }

  RCMD(c, "SELECT 0");
  RCMD(c, "FLUSHDB");

  fp = fopen(filename, "r");
  while (fgets(line, SIZEOFLINE, fp) != NULL) {
    word = strtok(line, DELIMETERS);
    while (word != NULL) {
      if (strlen(word) > 1) {
        RCMD(c, "ZINCRBY \"topchart\" 1 \"%s\"", word);
      }
      word = strtok (NULL, DELIMETERS);
    }
  }
  fclose(fp);

  RCMD(c, "ZRANGE \"topchart\" -10 -1 WITHSCORES");
  n = reply->elements-1;
  for (i=0; i<n; i=i+2) {
    printf("%s %s\n", reply->element[i+1]->str, reply->element[i]->str);
  }

  redisFree(c);
  return(0);
}
```

Result:

```
> ./create-topchart
10 "Retrieved"
10 "by"
10 "database"
19 "in"
21 "is"
24 "and"
30 "of"
30 "to"
33 "the"
57 "Redis"
```

CHAPTER 3

Indices and tables

- genindex
- modindex
- search