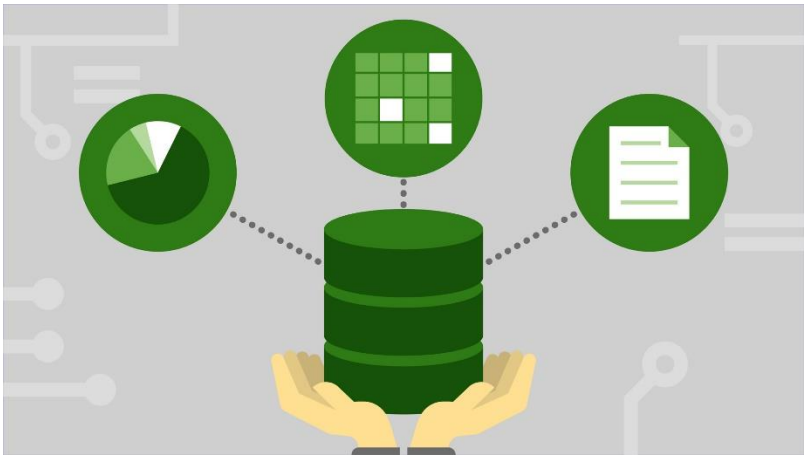
به نام خدا

دانشگاه تهران

پردیس دانشکده‌های فنی

دانشکده برق و کامپیوتر



آزمایشگاه پایگاه‌داده

دستورکار شماره ۸

شماره دانشجویی

۸۱۰۱۹۶۴۴۳

هومان چمنی

خرداد ۱۴۰۰

گزارش فعالیت‌های انجام شده

بخش اول - دستورات مقدماتی:

نتیجه انجام هر بخش در شکل های زیر قابل مشاهده است:

کار با String:

```
houmaan@hoomch:~$ redis-cli
127.0.0.1:6379> set mykey somevalue
OK
127.0.0.1:6379> get mykey
"somevalue"
127.0.0.1:6379>
```

کار با list:

```
houmaan@hoomch: ~
File  Edit  View  Search  Terminal  Help
127.0.0.1:6379> LPUSH mylist a
(integer) 1
127.0.0.1:6379> RPUSH mylist c
(integer) 2
127.0.0.1:6379> LPUSH mylist b
(integer) 3
127.0.0.1:6379> LPUSH mylist d
(integer) 4
127.0.0.1:6379> LRANGE mylist 0 0
1) "d"
127.0.0.1:6379> LRANGE mylist 0 3
1) "d"
2) "b"
3) "a"
4) "c"
127.0.0.1:6379> LPOP mylist
"d"
127.0.0.1:6379> LRANGE mylist 0 3
1) "b"
2) "a"
3) "c"
127.0.0.1:6379> RPOP mylist
"c"
127.0.0.1:6379> LRANGE mylist 0 3
1) "b"
2) "a"
127.0.0.1:6379>
```

کار با set:

```
127.0.0.1:6379> SADD myset "Hello"
(integer) 1
127.0.0.1:6379> SADD myset "World"
(integer) 1
127.0.0.1:6379> SADD myset "World"
(integer) 0
127.0.0.1:6379> SCARD myset
(integer) 2
127.0.0.1:6379> SMEMBERS myset
1) "World"
2) "Hello"
127.0.0.1:6379> SISMEMBER myset "one"
(integer) 0
127.0.0.1:6379> SISMEMBER myset "World"
(integer) 1
127.0.0.1:6379> SREM myset "World"
(integer) 1
127.0.0.1:6379>
```

کار با sorted set:

```
127.0.0.1:6379> ZADD myzset 1 "one"
(integer) 1
127.0.0.1:6379> ZADD myzset 2 "two"
(integer) 1
127.0.0.1:6379> ZADD myzset 3 "three"
(integer) 1
127.0.0.1:6379> ZSCORE myzset "two"
"2"
127.0.0.1:6379> ZRANGE myzset 2 3
1) "three"
127.0.0.1:6379> ZRANGE myzset 2 3
1) "three"
127.0.0.1:6379> ZSCORE myzset "three"
"3"
127.0.0.1:6379> ZRANGE myzset 2 3
1) "three"
127.0.0.1:6379> ZREM myzset "two"
(integer) 1
127.0.0.1:6379>
```

کار با hash :

```
127.0.0.1:6379> HMSET user:1000 username vahiiiid password 1234 age 2
OK
127.0.0.1:6379> HGETALL user:1000
1) "username"
2) "vahiiiid"
3) "password"
4) "1234"
5) "age"
6) "26"
127.0.0.1:6379> HSET user:1000 password 12345
(integer) 0
127.0.0.1:6379> HGET user:1000 age
"26"
127.0.0.1:6379> HEXISTS user:1000 age
(integer) 1
127.0.0.1:6379>
```

بخش دوم - توییت‌ها:

عکس زیر مربوط به کدی است که برای دریافت و ذخیره توییت‌ها استفاده شده است. پس از مشخص کردن کتاب‌خانه‌های معمول مورد استفاده, زمان معطلی ۲۰ ثانیه را در نظر گرفته‌ایم تا ظرف حدود یک ساعت بتوان توییت‌ها را دریافت کرد. تعداد لیمیت‌ها را هم مانند قبلی ۱۰۰۰ گذاشته‌ایم.

```
In [1]: import time
        from elasticsearch import Elasticsearch
        from redis import Redis
        import requests
        import logging
        import sys
        import re
```

```
In [2]: url = 'https://sahamyab.com/guest/twiter/list?v=0.1'
        total = 1000
        api_sleep = 20
```

```
In [3]: red = Redis()
        db_name = 'tweets'
```

گام بعدی که انجام شده تعریف کردن توابع‌ای است که از آنان استفاده خواهیم کرد.

In [4]:
```
def check_duplicate(data_id):
    if (red.sismember(db_name, data_id) == 1):
        return True
    return False

def add_to_db(data_id):
    red.sadd(db_name, data_id)

def inc_to_db(time_tag, input_hashtag):
    red.zincrby(time_tag, amount = 1, value = input_hashtag)

def get_count():
    return red.scard(db_name)

def find_hashtags(text):
    return re.findall(r"#(\w+)", text)

def split_and_parse(input_data):
    dates_container = input_data['sendTimePersian'].split()
    day_index, hour_index = 0, 1
    prefix = 'Hashtags:'
    day_tag = prefix + dates_container[day_index]
    hour_utility = dates_container[hour_index].split(':')[day_index]
    hour_tag = day_tag + ':' + hour_utility
    return (day_tag, hour_tag)

def increment_hashtags(hashtag_list, day_tag, hour_tag):
    for h in hashtag_list:
        inc_to_db(hour_tag, h)
        inc_to_db(day_tag, h)
```

تابع اول برای بررسی اینکه این توییت حال حاضر در دیتابیس وجود دارد یا خیر استفاده می‌شود. بدین صورت که با استفاده از تابع sismember که مربوط به خود کلاینت Redis است که بررسی می‌کند که یک آیدی (که همان آیدی مربوط به توییت است) درون یک set که نام آن را از اول کد با db_name مشخص کرده‌ایم وجود دارد یا خیر.

تابع بعدی هم add_to_db است که با گرفتن آیدی یک توییت آنرا به set ای که درون دیتابیس در نظر گرفته‌ایم اضافه می‌کند.

تابع بعدی تابعی است که برای هشتگ‌ها استفاده می‌شود. هر توییتی که میخواهیم اضافه کنیم پس از اضافه کردن آیدی آن هشتگ‌های آنرا جدا کرده و به ازای هر هشتگ یک count درون دیتابیس نگه‌داری می‌کنیم که نشان‌دهنده تعداد تکرار آن هشتگ است. با استفاده از دستور zincrby می‌توان مشخص کرد که عدد مربوط به یک value را که همان هشتگ است درون یک sorted set و با استفاده از کلید زمانی (که یا ساعت است یا روز) یک واحد اضافه کن. این تابع با همکاری تابع increment_hashtags کار می‌کند.

تابع بعدی هم find_hashtags بوده که دقیقا مشابه تابعی است که در آزمایش الستیک استفاده شد و توضیح خاصی ندارد.

تابع آخر نیز برای جدا کردن ساعت و روز از روی زمان ارسال توییت بکار می‌رود که دقیقا مانند همان پروتکلی که در شرح پروژه داده شد یک سری کلید برای استفاده درون sorted set که در بالا مطرح شد تولید کنیم. (هر توییت دو مقدار ساعت و روز دارد)

گام آخر نیز ران کردن کد می‌باشد که فرق چندانی با ران کردن کد الستیک نداشته و شکل آن به صورت زیر است:

```
n [5]: res = 0
while res < total:
    response = requests.request('GET', url, headers={'User-Agent': 'Chrome/61'})
    if response.status_code == requests.codes.ok:
        data = response.json()['items']
        for d in data:
            if (check_duplicate(d['id']) == False):
                d['hashtags'] = find_hashtags(d['content'])
                day_tag, hour_tag = split_and_parse(d)
                add_to_db(d['id'])
                increment_hashtags(d['hashtags'], day_tag, hour_tag)
                res = get_count()
    else:
        print("Response code error: " + str(response.status_code))
    print('Fetched and inserted {} tweets so far'.format(res))
    print('Waiting for {} seconds...'.format(api_sleep))
    time.sleep(api_sleep)
```

```
Waiting for 20 seconds...
Fetched and inserted 955 tweets so far
Waiting for 20 seconds...
Fetched and inserted 965 tweets so far
Waiting for 20 seconds...
Fetched and inserted 974 tweets so far
Waiting for 20 seconds...
Fetched and inserted 974 tweets so far
Waiting for 20 seconds...
Fetched and inserted 984 tweets so far
Waiting for 20 seconds...
Fetched and inserted 991 tweets so far
Waiting for 20 seconds...
Fetched and inserted 991 tweets so far
Waiting for 20 seconds...
Fetched and inserted 999 tweets so far
Waiting for 20 seconds...
Fetched and inserted 1006 tweets so far
Waiting for 20 seconds...
```

تمامی کدهایی که در این سه عکس تا الان مشاهده شد در فایل twitter_redis یکجا قابل دسترس است.

برای نشان دادن برترین‌های هر روز و هر ساعت نیز از کد شکل زیر استفاده شده که هر قسمت را به طور خلاصه توضیح می‌دهیم.

```
In [1]: import numpy as np
        import jdatetime
```

```
In [2]: akbar = jdatetime.datetime.now()
```

```
In [3]: from redis import Redis
        red = Redis()
```

```
In [4]: def print_output(input_lst, tag, mode):
            if (mode == 'h'):
                print("Hour:", tag, "-", tag+1)
            else:
                print(tag)
            for d in reversed(input_lst):
                print('-------', str((d[0]), encoding='utf8'),
                      '-------', d[1])
```

```
In [5]: print("Please provide desired hour:")
        hour = int(input())
```

```
Please provide desired hour:
9
```

بخش اول که در شکل بالا قابل مشاهده است مربوط به اضافه کردن کتابخانه و تعریف تابع برای خروجی دادن است. همچنین برای سادگی ما ساعتی از روز که می‌خواهیم توییت‌ها را داشته باشیم از کاربر دریافت می‌کنیم که فرقی در کارکرد کلی کد ندارد.

In [6]:
```
prefix = 'Hashtags:'
access_code = "Hashtags:{year}/0{month}/{day}".format(year=akbar.year,
                                                      month = akbar.month,
                                                      day = akbar.day)

if (np.floor(hour / 10) == 0):
    hour_suffix = ":0" + str(hour)
else:
    hour_suffix = ":" + str(hour)
hour_ranked = red.zrange(access_code + hour_suffix, -15,
                         -1, withscores=True)
print_output(hour_ranked, hour, "h")
```

```
Hour: 9 - 10
------- 9.0 ------- لکما
------- 8.0 ------- فجر
------- 8.0 ------- شاخص_بورس
------- 8.0 ------- سمگا
------- 8.0 ------- خودرو
------- 6.0 ------- شپنا
------- 4.0 ------- قصفها
------- 4.0 ------- رنیک
------- 4.0 ------- برکت
------- 3.0 ------- پالایش
------- 3.0 ------- فزرین
------- 3.0 ------- غنوش
------- 3.0 ------- شستا
------- 3.0 ------- خساپا
------- 3.0 ------- آینده
```

In [7]:
```
day_ranked = red.zrange(access_code, -15,
                        -1, withscores=True)
print_output(day_ranked, access_code, "d")
```

```
Hashtags:1400/03/25
------- 71.0 ------- شاخص_بورس
------- 30.0 ------- شپنا
------- 27.0 ------- پالایش
------- 24.0 ------- خودرو
------- 20.0 ------- برکت
------- 19.0 ------- سمگا
------- 18.0 ------- لکما
------- 16.0 ------- فزرین
------- 15.0 ------- سصفها
------- 13.0 ------- خساپا
------- 12.0 ------- چکاپا
------- 12.0 ------- قصفها
------- 12.0 ------- فجر
------- 10.0 ------- واعتبار
------- 10.0 ------- شستا
```

شکل بالا برای ساعت و روز می‌باشد. لازم به ذکر است که برای درست کردن access_code کار درست این است که به ازای ماه‌ها و روزهای مختلف ساخته شود چون برخی ماه‌ها و برخی روزها تک رقمی هستند ولی به دلیل ساده‌سازی و بدون کم شدن از کلیت صرفا یک حالت در نظر گرفته شد. برای در نظر گرفتن حالت‌های دیگر صرفا دوتا شرط باید بگذاریم و مشکل حل می‌شود. سپس ساعت دریافت شده را به کد می‌چسبانیم و با استفاده از zrange خود ردیس آیتم‌ها را به صورت رنک شده با امتیاز مربوط به هرکدام دریافت می‌کنیم. روال خروجی دادن موارد مربوط به روز هم مانند ساعت می‌باشد ولی فقط از access_code استفاده می‌کنیم.

بخش سوم - سوالات:

سوال اول: دوتا گزینه داریم که می‌توان انجام داد. یا از EXPIRE استفاده کنیم که باید به آن حتما تعداد ثانیه‌ها را به صورت Time to live بدهیم که شاید خیلی کاربردی نباشد چون هر پیامی در زمان متفاوتی ذخیره شده و همه باید در پایان روز حذف شوند ولی برخی جاها بد نیست. دستور دیگر که بهتر است EXPIREAT می‌باشد که مانند قبلی بوده ولی اون AT آخرش مشخص می‌کند که دقیقا چه زمانی پاک شود. اگر زمانی که می‌گذاریم قبل‌تر از زمان کنونی باشد همان لحظه پاک می‌شود. زمان را هم به صورت Unix timestamp باید در اختیار آن قرار دهیم.

سوال دوم: ده دقیقه را ابتدا باید به ثانیه تبدیل کنیم که می‌شود ۶۰۰ ثانیه. سپس از دستور save استفاده کرده و بعد از عدد ۶۰۰ هم مشخص می‌کنیم که در صورتی که تغییر در چند کلید دیدیم این عمل را انجام بده که مثلا می‌توان ۱ یا تعداد بیشتری را در نظر گرفت.

سوال سوم: در صورتی که روی دیسک داده‌ها را ذخیره‌سازی کنیم مشکلی پیش نخواهد آمد. در صورتی که مثلا فقط آیدی را در ردیس نگه‌داریم و اصل داده را یک جای دیگر باید حواسمان به آن جای دیگر هم باشد ولی در این کیس ما که کل داده‌ها را در کنار آیدی مستقیم در ردیس ذخیره می‌کنیم اگر همان precaution مربوط به بردن به دیسک را رعایت کنیم مشکلی بوجود نخواهد آمد.

مشکلات و توضیحات تکمیلی

مشکلی برخورد نکردم.

آنچه آموختم

کار کردن اولیه با دیتابس ردیس