8
MSc in Business Analytics
Big Data Systems
Redis & MongoDB Assignment

INSTRUCTIONS

You are going to use REDIS and MongoDB to perform an analysis on data related to classified ads from the used motorcycles market.

1. Install REDIS and MongoDB on your workstations. Version 4 of REDIS for windows is available here: https://github.com/tporadowski/redis/releases If you have an older version, make sure that you upgrade since some of the commands needed for the assignment are not supported by older versions. The installation process is straightforward.
2. Download the **BIKES_DATASET.zip** dataset from https://drive.google.com/open?id=1m4W6anTDphWRnHDwsh-hlexOGrAkMrSq
3. Download the **RECORDED_ACTIONS.zip** dataset from https://drive.google.com/open?id=1wyL8nQKDEu6rdr9BH6CgBwGnPnvRT8cJ
4. Do the tasks listed in the "TASKS" section:

SCENARIO

You are a data analyst at a consulting firm and you have access to a dataset of ~30K classified ads from the used motorcycles market. You also have access to some seller related actions that have been tracked in the previous months. You are asked to create a number of programs/queries for the tasks listed in the "TASKS" section.

ASSIGNMENT NOTES

- You may work on any programming language of your choice. Code samples are provided in R but the choice of language is up to you.
- Working with R is recommended, since the material uploaded on Moodle uses R in order to demonstrate Redis' usage.
- Assignment should be done in groups of two.
- The dataset is in JSON format. It needs cleaning. You don't need to follow the guidelines provided below. You may do the cleaning any way you like.
- In your deliverable, you should include (along with your code) a report justifying the steps you took in order to perform the tasks. The report should be **VERY** brief.
- **ONE** deliverable per team. The names of the members of each team along with their AM should be included in the first page of the report.
- Your code should be fully commented.
- Your deliverable should be a .zip file named as AM1_AM2.zip
- Optional tasks will have no effect on your final grade. However it's **strongly recommended** that you at least try these out in order to understand the actual benefit of the tools/technologies that you are using.

- You don't have to follow the tips provided in the tasks. You can do it any way you prefer. However, they may come in handy.
- The bitmaps.r file contains code samples for working with bitmaps and REDIS through R.
- The mongo.r file contains code samples for working with MongoDB and JSON files through R.

TASKS

Task 1.

In this task you are going to use the **"recorded actions"** dataset in order to generate some analytics with REDIS.

At the end of each month, the classifieds provider sends a personalized e-mail to some of the sellers with a number of suggestions on how they could improve their listings. Some e-mails may have been sent two or three times in the same month due to a technical issue. Not all users open these e-mails. However, we keep track of the e-mails that have been read by their recipients. Apart from that you are also given access to a dataset containing all the user ids along with a flag on whether they performed at least one modification on their listing for each month.

In brief, the datasets are the following:
- emails_sent.csv "Sets of EmailID, UserID, MonthID and EmailOpened"
- modified_listings.csv "Sets of UserID, MonthID, ModifiedListing"

The first dataset contains User IDs that have received an e-mail at least once. The second dataset contains **all** the User IDs of the classifieds provider and a flag that indicates whether the user performed a modification on his/her listing. Both datasets contain entries for the months January, February and March.

You are asked to answer a number of questions using REDIS Bitmaps. A Bitmap is the data structure that immediately pops in your head when the need is to map Boolean information for a huge domain into a compact representation. REDIS, being an in-memory data structure server, provides support for bit manipulation operations. However, there isn't a special data structure for Bitmaps in REDIS. Rather, bit level operations are supported on the basic REDIS structure: Strings. Now, the maximum length for REDIS strings is 512 MB. Thus, the largest domain that REDIS can map as a Bitmap is 2^32 (512 MB = 2^29 bytes = 2^32 bits).

Bitmaps examples:

Let's take the following bitmap as an example. Each bit corresponds to a client. Our company has 8 clients in total. The value of 1 means that the client purchased something from our online store in August:

AugustSales:

0	1	1	0	1	0	0	0

- Clients at the 0,3,5,6,7 positions did not purchase anything.

- Clients at the 1,2,4 positions did at least one transaction in August.

Let's add another bitmap to the example. It contains the September sales of the same company for the exact same clients:

SeptemberSales:

1	1	0	0	1	1	0	0

- Clients at the 2,3,6,7 positions did not purchase anything.
- Clients at the 0,1,4,5 positions did at least one transaction in September.

In order to create a Bitmap in REDIS you may use the SETBIT command. The syntax of SETBIT is: **>>** SETBIT key offset value

In order to create the SeptemberSales Bitmap we should enter the following commands:

>> SETBIT SeptemberSales 0 1
>> SETBIT SeptemberSales 1 1
>> SETBIT SeptemberSales 4 1
>> SETBIT SeptemberSales 5 1

Having these Bitmaps at hand, makes it very easy for us to calculate things like:
- Which clients ordered at least once for two months in a row?
- Which clients have not placed any orders within these two months?

This can be achieved with the use of bit-wise logical operations.

For example, in order to find out the clients that ordered at least once every month, we could perform an "AND" bitwise operation:

AugustSales	0	1	1	0	1	0	0	0
SeptemberSales	1	1	0	0	1	1	0	0
AugustSales **AND** SeptemberSales	0	1	0	0	1	0	0	0

In REDIS the following bitwise operations are supported:

AND	A bitwise AND performs the logical AND operation on each pair of the corresponding bits. If both bits are 1, the bit in the resulting binary representation is 1 (1 & 1 = 1); otherwise, the result is 0 (1 & 0 = 0 and 0 & 0 = 0). For example: 0101 AND 0011 = 0001

OR	A bitwise OR performs the logical inclusive OR operation on each pair of corresponding bits. The result is 0 if both bits are 0; otherwise the result is 1. For example: 0101 OR 0011 = 0111
XOR	A bitwise XOR performs the logical exclusive OR operation on each pair of corresponding bits. The result is 1 if only the first bit is 1 or only the second bit is 1, but will be 0 if both are 0 or both are 1. For example: 0101 XOR 0011 = 0110
NOT	The bitwise NOT, performs logical negation on each bit. Bits that are 0 become 1, and those that are 1 become 0. For example: NOT 0111 => 1000

These operations are performed with the BITOP command. The results of each command are written in a new key.

The syntax is as follows:

	Syntax	**Example**
AND	BITOP AND destkey srckey1 srckey2	BITOP AND results AugustSales SeptemberSales
OR	BITOP OR destkey srckey1 srckey2	BITOP OR results AugustSales SeptemberSales
XOR	BITOP XOR destkey srckey1 srckey2	BITOP XOR results AugustSales SeptemberSales
NOT	BITOP NOT destkey srckey	BITOP NOT results AugustSales

In all these examples, the results will be written in the key "results".

In order to count the number of "1"s in a key, we may use the BITCOUNT command. So, in order to count the number of clients that ordered in August, we would do: **>>** BITCOUNT AugustSales

Now that you are familiar with all the theory and tools that you need to work with Bitmaps in REDIS, let's proceed with your assignment.

General Note: Some users may have received more than one e-mail in the same month. If a client opened at least one of the e-mails that she/he received in the same month then we will classify this client as having opened this month's newsletter.

Provide answers for the following questions:

1.1 How many users modified their listing on January?
Tip: Create a BITMAP called "ModificationsJanuary" and use "SETBIT -> 1" for each user that modified their listing. Use BITCOUNT to calculate the answer.

1.2 How many users did NOT modify their listing on January?
Tip: Use "BITOP NOT" to perform inversion on the "ModificationsJanuary" BITMAP and use BITCOUNT to calculate the answer. Combine the results with the answer of 1.1. Do these numbers match the total of your users? Even if they don't, an explanation of why this happens will give you the full grade. Keep in mind that all BITOP operations happen at byte-level increments.

1.3 How many users received at least one e-mail per month (at least one e-mail in January and at least one e-mail in February and at least one e-mail in March)?
Tip: Create three BITMAPS "EmailsJanuary", "EmailsFebruary" and "EmailsMarch". Fill these with "SETBIT" and use "BITOP ***AND***" followed by "BITCOUNT" in order to calculate the answer.

1.4 How many users received an e-mail on January and March but NOT on February?
Tip: Perform "BITOP AND" on "EmailsJanuary" and "EmailsMarch". Perform an inversion of "EmailsFebruary" and use "BITOP AND" as well.

1.5 How many users received an e-mail on January that they did not open but they updated their listing anyway?
Tip: Create a new BITMAP "EmailsOpenedJanuary".

1.6 How many users received an e-mail on January that they did not open but they updated their listing anyway on January OR they received an e-mail on February that they did not open but they updated their listing anyway on February OR they received an e-mail on March that they did not open but they updated their listing anyway on March?
Tip: Create two new BITMAPs "EmailsOpenedFebruary" and "EmailsOpenedMarch". Do the same thing you did on 1.5 and calculate the answer using "BITOP OR".

1.7 Does it make any sense to keep sending e-mails with recommendations to sellers? Does this strategy really work? How would you describe this in terms a business person would understand?
Tip: You may use the findings of the previous questions or calculate anything else you want in order to justify your answer.

1.8 (Optional Task) Do the previous subtasks again by using any type of relational or non-relational database. Compare the complexity of the solutions. Then benchmark the query execution time for the dataset that you have. At last, boost the number of entries to 1 billion rows (create your own dummy entries). Perform the benchmark again.

Task 2.

In this task you are going to use the **"bikes"** dataset in order to generate some analytics with MongoDB.

2.1 Add your data to MongoDB.
Tip 1: You are free to structure your data whatever way you see fit. Before deciding on that, read the other tasks below. These will help you in order to decide on the data cleaning actions that you need to perform. You are allowed to perform any data cleaning actions you like. Please document all the actions that you performed (**briefly**) along with the reasoning behind any of your actions. The dataset is not clean. You might need to remove entries (or edit them) in order to maintain a clean database.
Tip 2: You will need to read all the files from R, do some cleaning and then add the data to MongoDB. When dealing with files split in that many folders, there are two options on how you read these files. The first (simpler) option is to write some code that will recursively read each folder, discover the files and bring them to memory. If you choose this route, every time you execute your code, the files have to be re-discovered. This takes time. Another option is to generate a list with all the paths of the files and use this file as an index whenever you need to do some kind of manipulation (read/write) to these files. This

approach will be much faster. In order to build that file, open the folder through a terminal and run the following command:

- ***Windows Powershell:*** dir -Recurse -Name -File > files_list.txt
- ***Unix Terminal:*** find * | grep json > files_list.txt
- ***Windows CMD:*** dir /a-D /S /B > files_list.txt

Now, the only thing you have to do through your code is read the "files_list.txt" line by line and load the file that is in the path contained in each row. No time will be spent in order to discover the files in case you want to re-run your code. In this assignment, you have a total of ~30K files. In a real-life scenario this number could have been several millions. In this case, the second option would most probably be your only option.

Tip 3: You will need to work on your data prior to writing to the database. Code samples of working with MongoDB through R are available on the "mongo.r" file. Use this as a reference along with the documentation of the package used.

2.2 How many bikes are there for sale?

2.3 What is the average price of a motorcycle (give a number)? What is the number of listings that were used in order to calculate this average (give a number as well)? Is the number of listings used the same as the answer in 2.2? Why?

2.4 What is the maximum and minimum price of a motorcycle currently available in the market?

Tip: The numbers should make sense.

2.5 How many listings have a price that is identified as negotiable?

Tip: Search for the word "Negotiable" in the ad.

2.6 (Optional) For each Brand, what percentage of its listings is listed as negotiable?

2.7 (Optional) What is the motorcycle brand with the highest average price?

2.8 (Optional) What are the TOP 10 models with the highest average age? (Round age by one decimal number)

Tip: Calculate age based on registration date. You don't need to take into account the months (only years). Group by model, calculate AVG Age and then Sort. Keep the TOP 10. In case of draws, treat it the same way you would in a real life scenario.

2.9 (Optional) How many bikes have "ABS" as an extra?

2.10 (Optional) What is the average Mileage of bikes that have "ABS" AND "Led lights" as an extra?

2.11 (Optional) What are the TOP 3 colors per bike category?

2.12 (Optional) Identify a set of ads that you consider "Best Deals".

Tip: Describe "why" in a manner that a business person would understand. Justify your decision with actual data. Even though it's not really needed, you are free to use external data sources.