Lab 3: MongoDB queries

Due date: January 21, 11:59pm.

Lab Assignment

Assignment Preparation

This is an individual lab. I expect every person to complete it without consulting others.

This is a short lab to give you some familiarity with MongoDB's `db.<collection>.find` command. To complete this lab, you will need to create a simple Javascript program, which, largely will consists of your queries and the loops over the cursor printing the results.

Data

In this lab you will write a number of MongoDB queries extracting data from some collections that you will set up. Below, we briefly describe the data collections you are expected to create.

Your lab will use two data collections: one constructed out of the `BeFuddled` JSON objects, and one - constructed out of the `ThghtShre` JSON objects.

For this lab **we are not providing you the data on which your queries are to be run**. Instead, you will build queries that return correct answers on **any** instance of a collection of a given type. You will test your queries by building specific collections out of the objects constructed by your JSON generators, running the queries on them, and ensuring that the queries produce correct answers. For this lab, there is no need to generate large collections of JSON objects. You can successfully test everything using collections of size 50-100 objects. Some queries can be successfully tested on even smaller collections.

General notes. Create a script (use any language you want), that takes as input a file containing a JSON document array, and produces as output a Javascript program that inserts the document into a database collection with a specified name (for both the database and the collection).

BeFuddled data. Generate one or more test JSON collections of BeFuddled JSON objects. Use collection named `fudd` to store BeFuddled objects in your MongoDB database. Use the script above to convert each JSON array produced by your generator into a Javascript program that creates/inserts the documents from the array into MongoDB. Use as many or as few collections as you need to debug your queries.

ThghtShre data. Generate one or more test JSON collections of ThghtShre JSON objects. Use collection named `thot` to store ThghtShre objects in your MongoDB database. Use the script above to convert each JSON array produced by your generator into a Javascript program that creates/inserts the documents from the array into MongoDB. Use as many or as few collections as you need to debug your queries.

Queries

The main objective of this lab is for each of you to get comfortable using MongoDB's `find()` command. To that extent, you will write 20 MongoDB queries: 10 for each of the dataset.

Query preparation and submission. For each dataset, you will submit your queries in two separate ways:

1. **Text file.** Create text files `beFuddled.mongo` and `thghtShre.mongo` and include all your queries there. Each query must be on its own lines, prefaced with a Javascript comment line specifying the query number and with at least one empty line between queries. The header of the file must contain one or more Javascript comment lines identifying with your name and other information about the file. The expected format is something like this:

```
// CSC 369. Lab 3.
// Alex Dekhtyar
// BeFuddled dataset

// Query 1
db.fudd.find(...)

// Query 2
db.fudd.find(...)

...

//end of queries
```

2. **Javascript program.** Create a Javascript program that connects to the MongoDB server, runs, in turn, each of the queries, and prints out the results. The program shall connect to the database bearing your name, and use the prescribed collection name. For each query, the program shall print its number, the query itself, followed by the results obtained from running the query on the collection. Name your programs `beFuddled.js` and `thghtShre.js`.

BeFuddled queries

Write MongoDB `db.<collection>.find()` queries that produce answers to each of the questions below. Each question must be answered with exactly one `find()` command.

1. Report, ordered from the earliest to the latest, all actions for game with id 1.
2. Report the first three regular moves for game 1, sorted in ascending order by the move id.
3. Report all regular moves that used location $(x = 11, y = 12)$. Sort the output in descending order by the total number of points in the game after the move.
4. Report all special moves that resulted in negative point gain.
5. Find the names of all users who completed at least one game. Report the name of each user *no more than the number of times they played the game*. Report only the user name (no other attributes, and no `"_id"` attribute).
6. Report the three largest point scores observed. Report only the scores, no other information is to be retrieved.
7. Report all special moves of type `"Clear"` and (separately) all regular moves that resulted in 10 points or more added to the score. Sort output by the action type, and for the regular moves - in descending order of the added score.
8. Report all moves that were made in the middle portion of the board $(x \in [8, 12], y \in [8, 12])$ that yielded either a negative score, or a positive score higher than 10.
9. Report the total number of started games.
10. For each regular move that resulted in adding five or six points to the score, report only its location, game number, and move number. Sort output in ascending order by game number and by move number.

ThghtShre Queries

Write MongoDB `db.<collection>.find()` queries that produce answers to each of the questions below. Each question must be answered with exactly one `find()` command.

1. Report all messages written by user `u100`.
2. Report messages with the sixth-to-twelveth highest message id.
3. List all user ids of users who were recepients of at least one message (each user can be listed as many times as the number of messages they received). Sort output in alphabetical order by user id.
4. Count the number of protected messages.
5. Output the text (and only the text) for all messages written in response to another message.
6. Report all public messages sent to self. Sort output in descending order byt message id.
7. Report the text (and only the text) of all public messages sent to self that were written in response to another message.
8. Report the recepient and the text of the four messages with the largest numbers.
9. Report user ids for all private messages sent to self.
10. Find all public messages addressed to all that were written *not* in response to another message. Report only the text of each message, and order the messages in ascending order by message id.

Submission

Submit the following artefacts:

- Your script for creating Javascript data insertion programs.
- Data insertion Javascript programs for one data collection for each dataset (`BeFuddled` and `ThghtShre`). (The size of the dataset need not exceed 50-100 objects).
- `BeFuddled.mongo` and `ThghtShre.mongo` text files with queries.
- `BeFuddled.js` and `ThghtShre.js` Javascript programs with queries.
- `README` file.

All submitted files must contain your name on them.

Submit all your code in a single archive (zip or tar.gz).

Use `handin` to submit as follows:

Section 01:

```
$ handin dekhtyar lab3-01 <FILES>
```

Section 03:

```
$ handin dekhtyar lab3-03 <FILES>
```

Good Luck!