CS 488/588 Cloud & Cluster Data Management
Fall 2020

Project Part 2: Data Modeling and Application Design

Due: Friday, November 13, by midnight (D2L)

The DJS: Dominique Moore, Jane Seigman, Santiago Tobon

Part 2 Deliverables:

1) **Design Report**.
   1. **Choice of Dataset:** *Option 2: Freeway Data (Relational Data)*

   2. **Data Modeling & Application Design.**
      a. *Data Model Summary & Storage:*

We decided to use the traffic data set because MongoDB accepts .csv files as input. We didn't want to work with xml files as was presented in the DBLP data model since none of us are particularly experienced working with those kinds of files. We had initially decided to work with another data set, StackOverflow, that was publically available on the Google Cloud Platform's BigQuery. However when it came down to putting the data in the database, the task proved rather tedious. We realized that the data set was much too large and trying to extract a subset would be a feat. So, by process of elimination we have chosen the freeway dataset.

Since the freeway dataset is relational, we will be restructuring the data to fit our NoSQL model. MongoDB stores data as objects in a document and joins are not well supported, so we realized we needed to have a data model with as few collections as possible. We determined that detectors were the kingpin to most of the queries, so our data model makes each detector an object that holds all of the information of its highway and station. This might seem redundant for highway and station information, but it improves the performance of retrieving detector information. For some of the questions, we will have to compromise on efficiency. This is because we may potentially have to search the entire loopdata collection to find an answer to certain questions.

i. Show how the example data can be represented in the data model of your data store.

See example 1 on the last page.

ii. List any indexes you will create.

We assume that the freeway data set will allow us to create different kinds of indexes to further improve the efficiency of the query questions. Each object will have a

unique object id that will be used as an index, however, those are long and hard to remember so we will likely make **detector_id** an index. We can also create other indexes for other unique data items, such as **speed** for example.

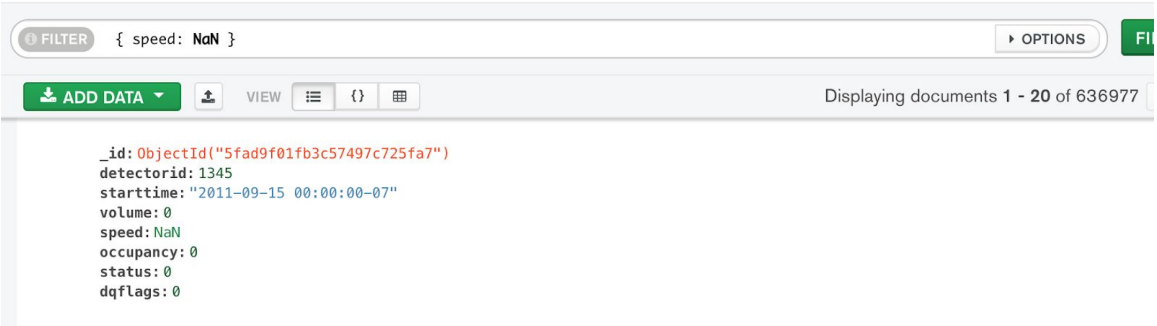
iii. Present one variant you considered during the design phase.
See example 2 on the last page.

iv. Explain why you chose your model. Do you expect it to perform better? Is it easier to use? What motivated you to pick the model you did.

We choose this model because it reduces the need for joins by storing the highway and station information inside each detector object. This better follows Mongo's noSQL data model. We expect this model to perform better than the variant we considered because we are more likely to query based on detectors than highways. What motivated us to use this model was how it would be easier to query the data to answer the questions.

b. Data cleansing and data transfer are important aspects of any project involving data because the incoming raw data might be in any shape or form. Describe the process you will use to restructure the data to fit your model and to clean the data.

MongoDB does a good job of ingesting the data from the traffic .csv files. During the import, it gives the option to select the variable type to import the data as (i.e string, number, float). We used MongoDB Compass to import the data to our cluster database, which provides a useful filter tool. This filter tool will allow us to clean the data. For example, the freeway_loop data contains over 636977 documents with NULL value for speed, meaning that the detector failed to correctly gather the data from that event. These documents are not useful to us, as they do not provide any useful data. Therefore, in order to cleanse the data of these empty entries we can run the following filter:



The documents left, will be only ones that contain non-null values. Alternate filters might be: **{$and: [{volume:NaN},{speed:NaN}]}** to clear out any documents that have NULL values for both speed and volume or

{$and: [{$or: [{volume:NaN},{volume:0}]},{speed:NaN}]} to clear out even more documents that might not have helpful information. We need to figure out if we need entries that have a volume of 0 as opposed to a volume of NULL. This is the filter we will use to extract out only the loopdata that has non-NULL, non-zero data:
{$and: [{$or: [{volume: {$ne: NaN}},{volume:{$ne: 0}}]},{speed:{$ne: NaN}}]}
In order to delete the documents(entries) that we do not need, we will have to use the Mongo shell **db.collection.deleteMany()** function as the Compass UI does not allow mass deleting. Alternatively, we can filter out the undesirables and save the results to a json or .csv file, then reupload the desirables and delete the old collection. Another more sophisticated method is to write a script to read the data from the .csv file and input only the data that meets certain criteria that satisfies the above filters. We may use a combination of these methods in order to be able to organize the data into our data model. We have determined that we do not need the key:value pairs for status, dqflag, or occupancy in the freeway_loopdata, and our data model reflects that.

c. Outline implementation strategies in pseudo-code for the questions based on the capabilities of your data store and the indexes you plan to define for all the provided questions.

1) *Count low speeds and high speeds:* Find the number of speeds < 5 mph and > 80 mph in the data set.
the count of this query: $or:[{speed:{$lt:5}},{speed:{$gt:80}}]
2) *Volume*: Find the total volume for the station Foster NB for Sept 15, 2011.
Find the detector ids that correspond to “Foster NB” and use the “$match” option in the aggregation to search for the regex: 2011-09-15. Add another stage and use the “$group” option to sum all the volume values.
$match: { starttime: {$regex: '2011-09-15'} }
$group: { _id: null, totalVolume: { $sum: "$volume"} }
{$lookup:{from: "freeway_detectors", localField: "detectorid", foreignField: "detectorid", as: "detector"}}
{ starttime: {$regex: '2011-09-15', $options: 'i'} }

3) *Single-Day Station Travel Times:* Find travel time for station Foster NB for 5-minute intervals for Sept 15, 2011. Report travel time in seconds.

There are 288 five minute intervals in 24 hours.

To calculate travel time in seconds: (length of road) / (sum of the speeds)*3600

Take travel time and divide by 288 gives the average of speeds per 5 minute intervals

Use aggregation to $match for the station Foster NB on the day 2011-09-15. In a new stage window use $group to sum the speeds. In another window, $project the length divided by the speeds then divide that result by 288.

```
{
    "$match": {"locationtext": "Foster NB"}
}
```

4) *Peak Period Travel Times:* Find the average travel time for 7-9AM and 4-6PM on September 22, 2011 for the I-205 NB freeway. Report travel time in minutes.

To calculate travel time: (length of road) / (sum of the speeds)

Use aggregation to $match for I-205 NB on 2011-22-09. In another window use $match to and $or to find data that's oriented between 07:00:00 - 09:00:00 and between 16:00:00 – 18:00:00. Using another aggregation window use $group with $sum operation to add the speeds of all the data, and $count to keep a count of the data items. With another aggregation window use $project to $divide the length of I-205 NB with the calculated sum of speeds, then $divide that value by the count.

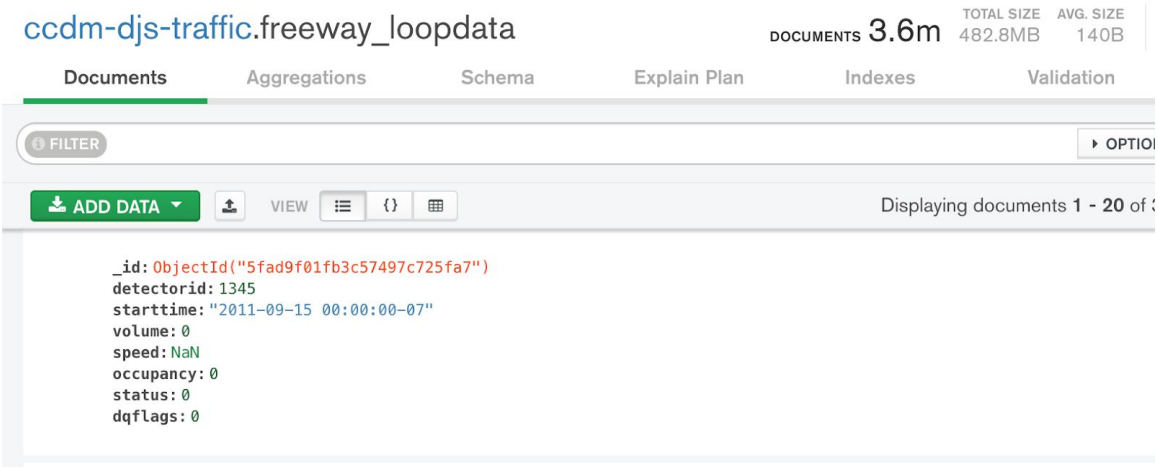
5) *Route Finding:* Find a route from Johnson Creek to Columbia Blvd on I-205 NB using the upstream and downstream fields

$match the upstream and downstreams from station to station from locationtext:Johnson Creek to locationtext: Columbia Blvd on highway I-205 NB.

6) *Update:* Change the milepost of the Foster NB station to 22.6.

Using MongoDB compass we can navigate to where the Foster NB station object is stored, and click on the pencil icon to edit. Click on the milepost field and change the value to 22.6 then click the update button to save the changes to the database.

d. Include proof that you have loaded at least a few data items into your system.



e. Discuss how your design could handle updates/inserts to the data.

The JSON mode in MongoDB Compass UI allows multiple documents to be inserted at a time as an array. Individual documents can be created via the UI as well. Mongo shell provides functions **db.collections.updateOne()**, **db.collections.updateMany()**, **db.collections.insertOne()**, and **db.collections.insertMany()**. However, ingesting streamed freeway loop data would require a more sophisticated and automated method of insertion and updates. This could be accomplished by writing a program that monitors raw data input and inserts the data as a document to the correct collection. The Python MongoDB library pymongo provides functions such as **pymongo.collection.Collection.updateOne()** that can be used to change the milepost of the Foster NB station to 22.6 and **pymongo.collection.Collection.insertMany()** to input cleaned and organized data into the appropriate collection.

2) **Meeting with the Prof to discuss your design.** Nov 12th @ 16:45

Data Model Example Document 1 (The one we are using):

Detector Collection:

{
_id: ObjectId(“5fad9e5bfb3c57497c725fa5”)
detectorid:1345
highway: {
highwayid:3
shortdirection:"N"
direction:"NORTH"

highwayname:"I-205"
}
milepost:14.32
locationtext:"Sunnyside NB"
detectorclass:1
lanenumber:1
station: {
stationid:1045
upstream:0
downstream:1046
stationclass:1
numberlanes:4
latlon:"45.43324,-122.565775"
length:0.94
}

}

Freeway LoopData Collection:

{
_id: ObjectId("5fad9f01fb3c57497c725faf")
detectorid:1345
starttime:"2011-09-15 00:02:40-07"
volume:1
speed:66
}

Data Model Example Document 2 (Variant that we decided against):

Highway Collection:

{
_id: ObjectId("5fad9e5bfb3c57497c725fa5")
highwayid:3
shortdirection:"N"
direction:"NORTH"
highwayname:"I-205"
stations: {
stationid:1045
milepost:14.32
upstream:0
downstream:1046
stationclass:1
numberlanes:4
latlon:"45.43324,-122.565775"
length:0.94
detectors: {
detectorid:1345
locationtext:"Sunnyside NB"
detectorclass:1
lanenumber:1
}
}

}

Freeway LoopData Collection:

{
_id: ObjectId(“5fad9f01fb3c57497c725faf”)
detectorid:1345
starttime:"2011-09-15 00:02:40-07"
volume:1
speed:66
}