CS 488P/588P

Yufei Tao, Mi Yon Kim

Term Project Part 2 Report

We decided to choose Freeway Dataset and we ended up with the design below.

```
Detector {
        Primary Key(MongoDB index)
        schema
        detectorid
        highway: {
                highwayid
                shortdirection
                direction
                highwayname
        }
        milepost
        locationtext(index)
        detectorclass
        lanenumber
        station: {
                stationid
                upstream
                downstream
                stationclass
                numberlanes
                latlon
                length
        }
}

Loopdata {
        Primary Key(MongoDB index)
        schema
        detectorid(index)
        data: [
         {
                starttime(index)
                volume
                speed
                occupancy
                status
                dqflags
         }
        ]
        stationid
        num_lowspeed
        num_highspeed
}
```

We plan to create an index consisting of the locationtext field from the Detector. We also plan to create an index consisting of the detectorid and a compound index consisting of the detectorid and starttime field from the Loopdata to search effectively. We had a variant we considered during the design phase. We considered the Station Collection instead of Detector collection. E.g.

```
station {
        _id
        schema
        stationid
        highway: {
                highwayid
                shortdirection
                direction
                highwayname
        }
        milepost
        locationtext(index)
        upstream
        downstream
        stationclass
        numberlanes
        latlon
        length
        detector: [
                {
                        detectorid
                        lanenumber
                }
        ]
}
```

We chose the Detector collection instead of Station collection, because Station will have an array of detectors. We assume that it is much simpler to handle the collection without an array. We have Detector and Loopdata collections. We put Station and Highway data inside of the Detector because of the performance, i.e., minimizing the need for joins by denormalizing. In addition, Highway(2 rows), Station(17 rows), and Detector(52 rows) have small sizes of data, therefore this design won't have much redundant data. We separate the Loopdata collection because Loopdata has a large size of data. By separating the Loopdata, we can simplify the

architecture and save the storage by removing the redundant data. Most of frequently asked questions can be retrieved from Detector, Loopdata or joining Detector with Loopdata. In addition, we plan to create an index for the locationtext field from the Detector because most of the questions are searching with a specific location. We also plan to create multiple indices for the detectorid and starttime field from the Loopdata because the questions can be answered with a specific list of detectorids and list of starttime. Therefore, we believe that this design will perform efficiently. We will talk more about this later in the implementation strategies part.

We plan to implement it with python to clean and transfer raw data. We can filter out all of the corrupted or invalid data(We are thinking of the data which has the speed zero from the Loopdata.xls) from the raw data. Next, we will restructure the data by implementing it with python. Last, we will create the json file to fit our model. After we have a final json file, we can import the json file to the storage using the MongoDB Compass.

To answer 6 questions, we plan to create indices. For example, we plan to create an index consisting of a locationtext field from the Detector. We also plan to create a compound index consisting of a detectorid and starttime field from the Loopdata. Additionally, we plan to create an index consisting of a detectorid from the Loopdata. We believe this will make us query efficiently.

For the first question, to find the number of low speeds are less than 5 mph and high speeds that are greater than 80 mph in the data set. Based on our model design, we can just search in the Loopdata compare each data and count the ones that are satisfying the conditions. The query we can use to do this question could be `db.Loopdata.find(`*`speed condition`*`).count()`.

The second question asks to find the volume for the station Foster NB on Sept 15, 2011. To solve this problem, we can first look for locationtext “Foster NB” in Detector Collection, which

will return a list of detectors with different detectorid from the same Detector Collection. Then in Loopdata collection we can use detectorid to find the sum of the volume in the data array where their starttime contains “09/15/2011”. The potential query we can use to solve this problem would be first use `db.Detector.find({“locationtext”: “Foster NB”})` to find the stationid of detectors are in that station. Then we can find all the volume data from all the detectors at that station by using the find command with the date condition , lastly we can sum up all the volume data, that would be the total volume for the station Foster NB for Sept 15, 2011. To sum up all the data, we can use the `db.Loopdata.aggregate({$group}).`

For the third question, we first find out the start time and the last start time within that 5 minutes, which would have 15 documents in that time interval. We will use the `db.Detector.find(“locationtext”: “Foster NB”)` to find the stationid and the station length, then use the stationid in Loopdata collection to find the number of speed and the sum of all speed during that time, take average of the the data. This process can be achieved by using `db.Loopdata.aggregate({$group}).` Then use the formula `Travel time in seconds= ((length)/(avg(speed(mile/hr))))*3600` to find the travel time.

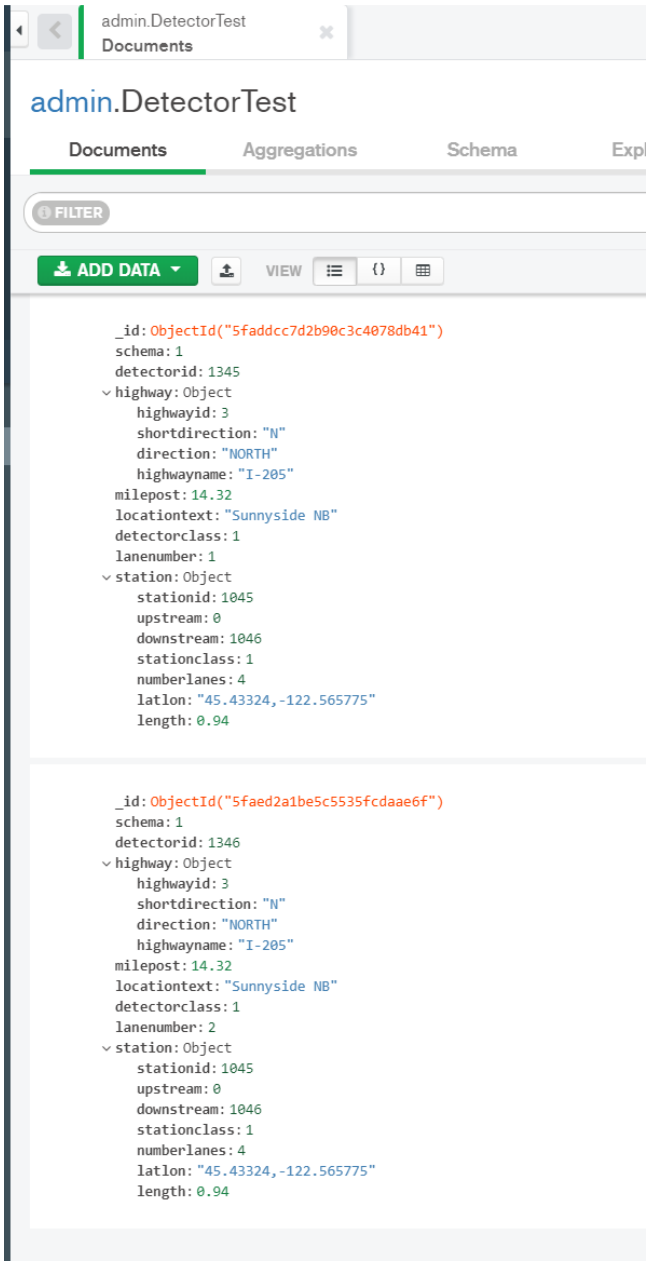
The fourth question asks to find the average travel time for 7-9AM and 4-6PM on September 22, 2011 for the I-205 NB freeway. To do that, we use highway name to find the list of stationids, then use each stationid in the array to find the average of speed inside Loopdata that has same stationid and satisfying time condition. Sum them up and divide by the number of data, that would be the average of the speed. The highway length can be obtained by adding up station’s length_mid from the Detector collection. Last, we can use the formula `Travel time in minutes = (sum(((length)/avg(speed))))*60`. The process to get the result can be fairly complicated in this question. We would have to aggregate many things together to get the result. We can first use `db.Detector.find()` to filter out data, then use

the result to perform joins with Loopdata collection using `$lookup`. At this point we will have a collection of documents that are on I-205 NB freeway. We can use `db.Result.aggregate()` with `$group` and `$match` to finish the rest of the calculations.

The fifth question asks to find the route from Johnson Creek to Columbia Blvd to I-205 NB. We can answer this question only with the Detector collection. First, we will create an empty list for adding locationtexts of the route. We will append the Johnson Creek. Second, we get downstream of Johnson Creek and upstream of Columbia Blvd to I-205. Then loop through the Detector collection to find the stationid, locationtext and downstream which matches with Johnson Creek's downstream(Foster NB). When we find it, we can append the locationtext into the list. Now we will change the downstream data to Foster NB's downstream. We will iterate until the downstream is the same as Columbia Blvd to I-205's stationid. Last, we will append Columbia Blvd to I-205 at the last. We can use `db.Detector.find({"locationtext": location name}, db.Detector.find({"stationid": downstream data})` command to answer this question.

The last question is to update the value at the milepost of the Foster NB station to 22.6. We can first compare the locationtext in Detector Collection to find all data that are at Foster NB. Then change the value of their milpost. To do this, we can use the command `db.Detector.updateMany({$set})`. This command will let us update multiple values on a field.

We inserted a few data items into our system and you can check it from the below.



In our system model, when we need to update the data, we would first need to inspect what potential duplicated fields we have in our two collections. For example, both Detector and Loopdata collection has stationid field. If we want to update the value of this data, we have to

make sure both collections will get updated. The query to do this could be `db.Detector.updateMany({$set})`. For data insertion, we first have to check if the data that's going to be inserted has new value that's not in the existing system. We would have to do some extra work to add a brand new value if there's one. If the values in all fields are already existing in the current system, then we can safely directly insert the data into the system. The command we could use is `db.Detector.insertOne()`. If we are inserting data to the Loopdata collection, we would have to retrieve the stationid from the Detector collection and insert it with the new data into the Loopdata collection.