Project Presentation

Mi Yon Kim, Yufei Tao

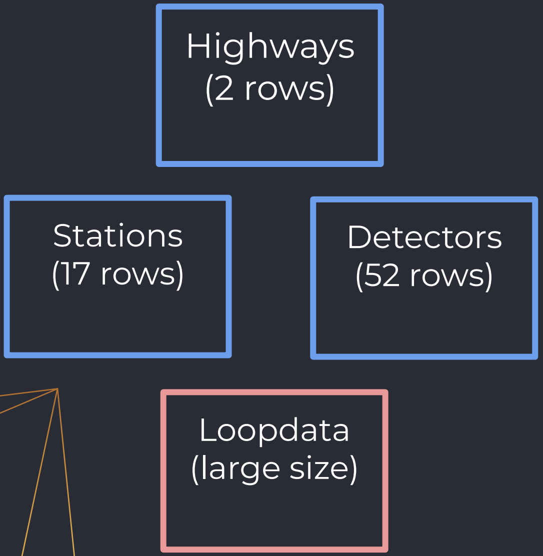
Project Overview

- Dataset: Freeway data

- Tools & Languages:
  - MongoDB Compass
  - MongoDB Database
  - Python(Pymongo)

Data model

- Original model(RDBMS)



Highways (2 rows)

Stations (17 rows)

Detectors (52 rows)

Loopdata (large size)

VS

- Our model(NoSQL)

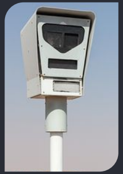
  - **Detectors**

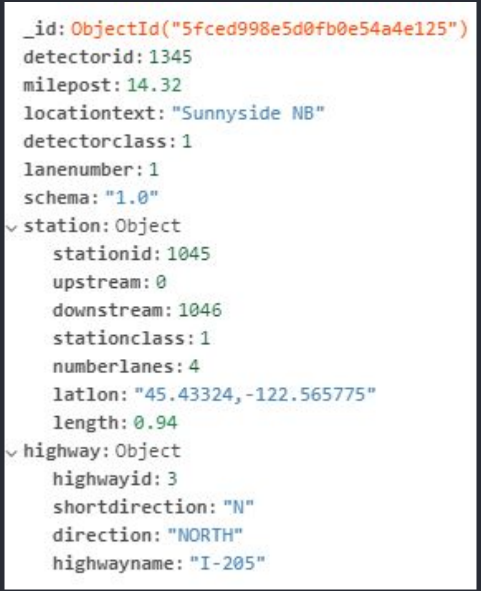
    - Combined 3 tables(Highway, Stations, Detectors)
    - Embedded documents of Highways and Stations.
    - Contains duplicated data however it can be minor because data is small

  - **Loopdata**

    - Added locationtext from the original model.

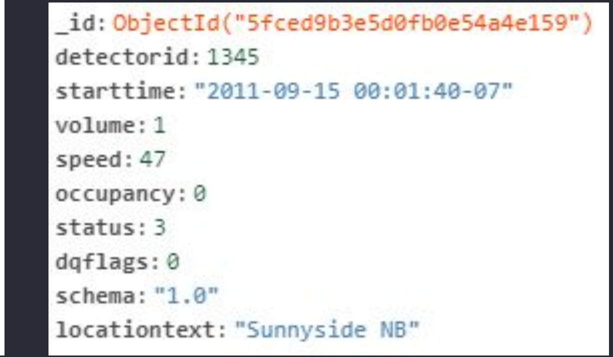
Data model





```
_id: ObjectId("5fced998e5d0fb0e54a4e125")
detectorid: 1345
milepost: 14.32
locationtext: "Sunnyside NB"
detectorclass: 1
lanenumber: 1
schema: "1.0"
station: Object
    stationid: 1045
    upstream: 0
    downstream: 1046
    stationclass: 1
    numberlanes: 4
    latlon: "45.43324,-122.565775"
    length: 0.94
highway: Object
    highwayid: 3
    shortdirection: "N"
    direction: "NORTH"
    highwayname: "I-205"
```

Detectors



```
_id: ObjectId("5fced9b3e5d0fb0e54a4e159")
detectorid: 1345
starttime: "2011-09-15 00:01:40-07"
volume: 1
speed: 47
occupancy: 0
status: 3
dqflags: 0
schema: "1.0"
locationtext: "Sunnyside NB"
```

Loopdata

ETL

- Implemented with python

- Drop loopdata which has the Null or zero value in the speed field

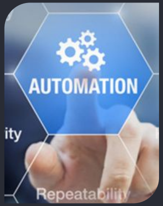
- Issues

  - Some data in Detector’s locationtext and Station’s locationtext didn’t match
  (e.g. Columbia to I-205 NB(Station table) vs I-205 NB at Columbia(Detectors))

  - Some detectorid in Loopdata doesn’t included in Detectors table.
  (e.g. Loopdata with 1350 detectorid vs 1350 doesn’t show up in Detectors table)

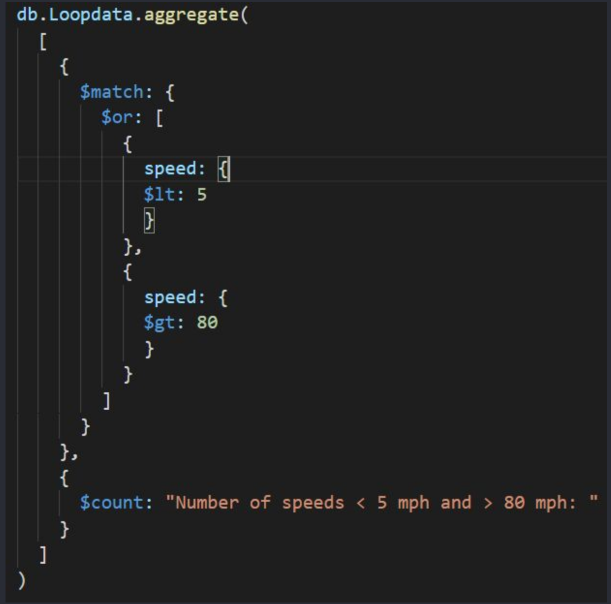
  - Solution 1: Modified the csv file

  - Solution 2: Automate through script

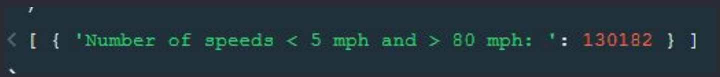


Query Examples

<u>Q1: Count low speeds and high speeds:</u>
Find the number of speeds < 5 mph or > 80 mph in the data set.



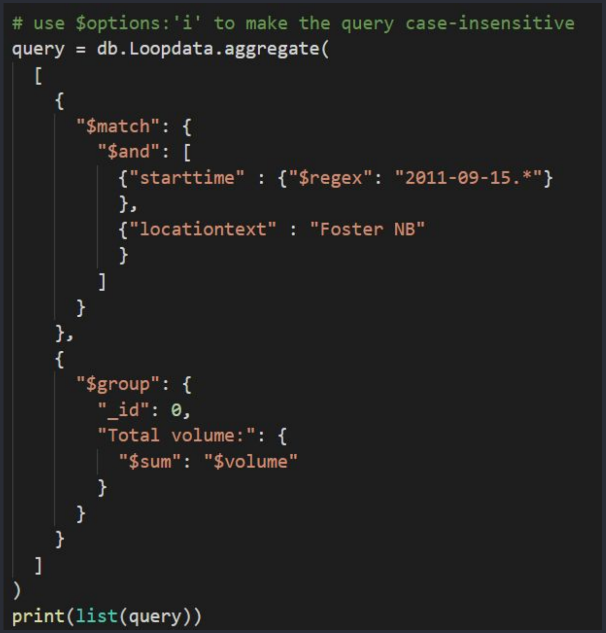
```
db.Loopdata.aggregate(
  [
    {
      $match: {
        $or: [
          {
            speed: {
            $lt: 5
            }
          },
          {
            speed: {
            $gt: 80
            }
          }
        ]
      }
    },
    {
      $count: "Number of speeds < 5 mph and > 80 mph: "
    }
  ]
)
```



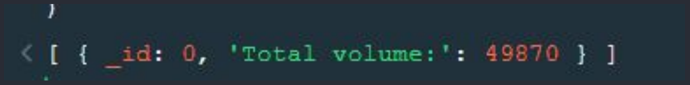
```
< [ { 'Number of speeds < 5 mph and > 80 mph: ': 130182 } ]
```

Query Examples

Q2: Volume: Find the total volume for the station Foster NB on Sept 15, 2011.



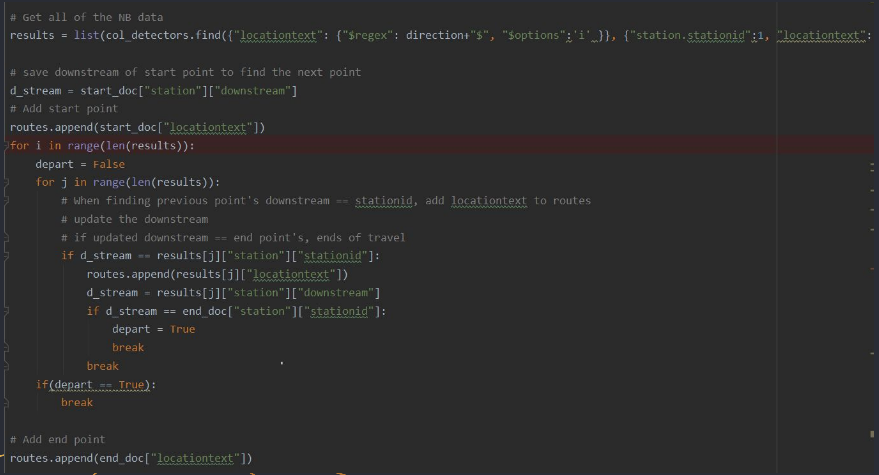
```
# use $options:'i' to make the query case-insensitive
query = db.Loopdata.aggregate(
  [
    {
      "$match": {
        "$and": [
          {"starttime" : {"$regex": "2011-09-15.*"}
          },
          {"locationtext" : "Foster NB"
          }
        ]
      }
    },
    {
      "$group": {
        "_id": 0,
        "Total volume:": {
          "$sum": "$volume"
        }
      }
    }
  ]
)
print(list(query))
```



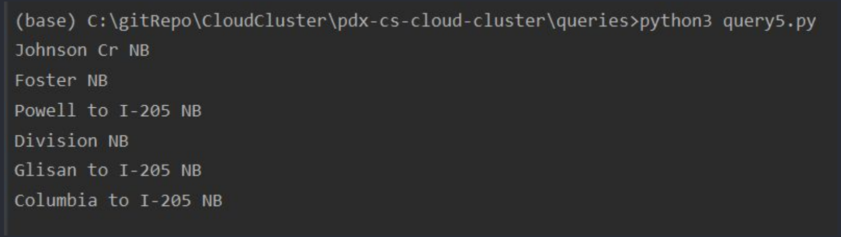
```
  )
< [ { _id: 0, 'Total volume:': 49870 } ]
```

Query Examples

Q5: Route Finding: Find a route from Johnson Creek to Columbia Blvd on I-205 NB using the upstream and downstream fields.



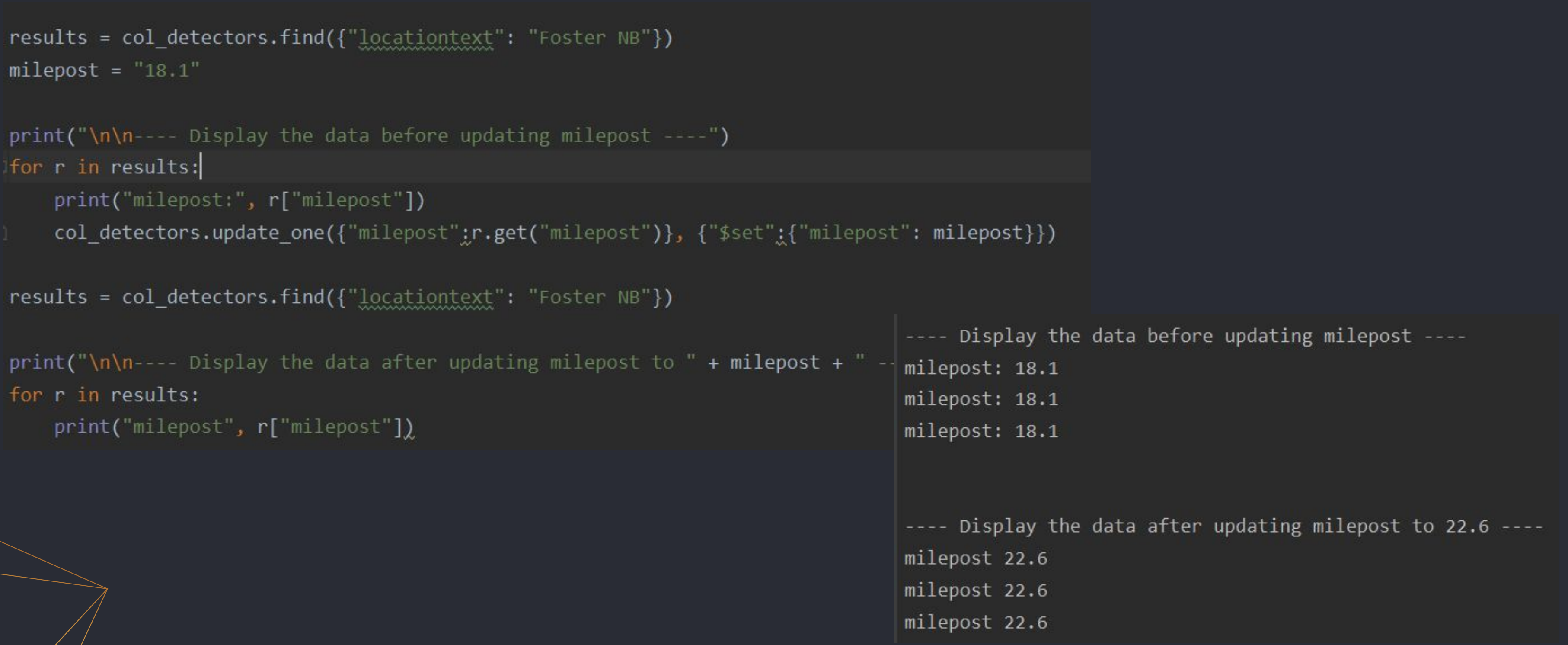
```
# Get all of the NB data
results = list(col_detectors.find({"locationtext": {"$regex": direction+"$", "$options":'i' }}, {"station.stationid":1, "locationtext":

# save downstream of start point to find the next point
d_stream = start_doc["station"]["downstream"]
# Add start point
routes.append(start_doc["locationtext"])
for i in range(len(results)):
    depart = False
    for j in range(len(results)):
        # When finding previous point's downstream == stationid, add locationtext to routes
        # update the downstream
        # if updated downstream == end point's, ends of travel
        if d_stream == results[j]["station"]["stationid"]:
            routes.append(results[j]["locationtext"])
            d_stream = results[j]["station"]["downstream"]
            if d_stream == end_doc["station"]["stationid"]:
                depart = True
                break
            break
    if(depart == True):
        break

# Add end point
routes.append(end_doc["locationtext"])
```



```
(base) C:\gitRepo\CloudCluster\pdx-cs-cloud-cluster\queries>python3 query5.py
Johnson Cr NB
Foster NB
Powell to I-205 NB
Division NB
Glisan to I-205 NB
Columbia to I-205 NB
```

Query Examples

Q6: Update: Change the milepost of the Foster NB station to 22.6.



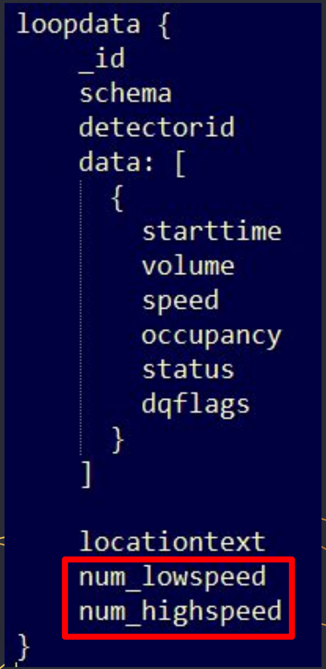
```
results = col_detectors.find({"locationtext": "Foster NB"})
milepost = "18.1"

print("\n\n---- Display the data before updating milepost ----")
for r in results:
    print("milepost:", r["milepost"])
    col_detectors.update_one({"milepost":r.get("milepost")}, {"$set":{"milepost": milepost}})

results = col_detectors.find({"locationtext": "Foster NB"})

print("\n\n---- Display the data after updating milepost to " + milepost + " --
for r in results:
    print("milepost", r["milepost"])
```

```
---- Display the data before updating milepost ----
milepost: 18.1
milepost: 18.1
milepost: 18.1



---- Display the data after updating milepost to 22.6 ----
milepost 22.6
milepost 22.6
milepost 22.6
```

Demo

Critique

- Changed Loopdata model
  - Reason: MongoDB document size limit: 16mb



```
loopdata {
      _id
      schema
      detectorid
      data: [
        {
          starttime
          volume
          speed
          occupancy
          status
          dqflags
        }
      ]

      locationtext
      num_lowspeed
      num_highspeed
}
```

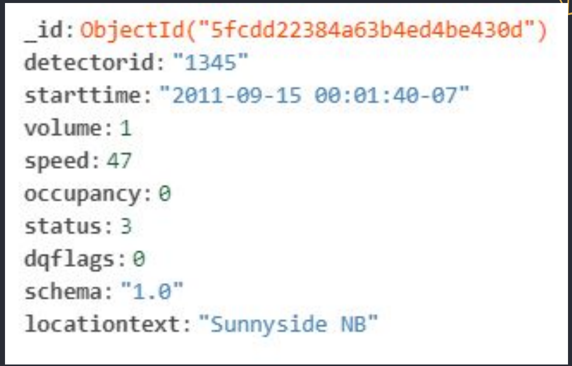
Previous loopdata

Pros

- Improve performance on query #1
- No duplicate data

Cons

- Complicated
- Doesn't fit in MongoDB System



```
_id: ObjectId("5fcdd22384a63b4ed4be430d")
detectorid: "1345"
starttime: "2011-09-15 00:01:40-07"
volume: 1
speed: 47
occupancy: 0
status: 3
dqflags: 0
schema: "1.0"
locationtext: "Sunnyside NB"
```

current loopdata

Pros

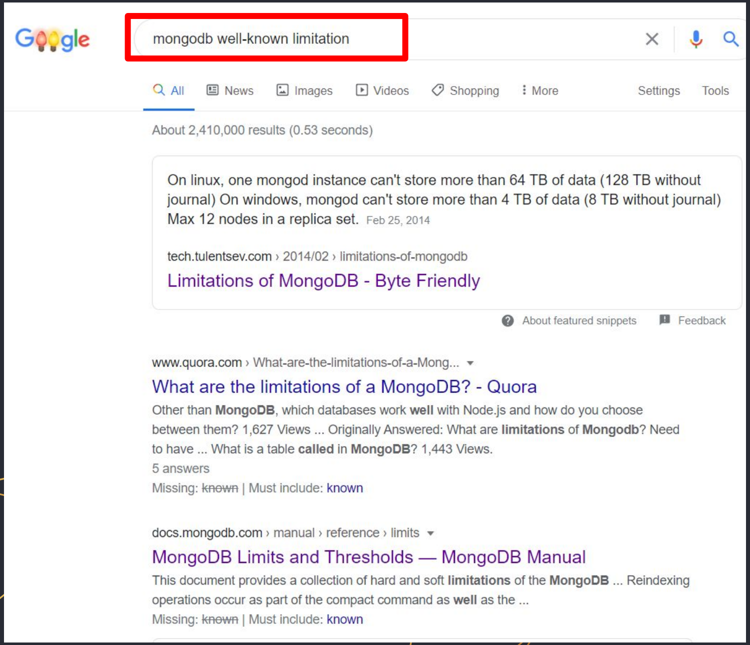
- Simpler
- Simpler to make queries

Cons

- Duplicate data
- Bad performance on query #1 if the data size gets bigger

Critique

- Advice
  - Spend more time on researching new system before you design



mongodb well-known limitation

About 2,410,000 results (0.53 seconds)

On linux, one mongod instance can't store more than 64 TB of data (128 TB without journal) On windows, mongod can't store more than 4 TB of data (8 TB without journal) Max 12 nodes in a replica set. Feb 25, 2014

tech.tulentsev.com › 2014/02 › limitations-of-mongodb

Limitations of MongoDB - Byte Friendly

www.quora.com › What-are-the-limitations-of-a-Mong...

What are the limitations of a MongoDB? - Quora

Other than **MongoDB**, which databases work **well** with Node.js and how do you choose between them? 1,627 Views ... Originally Answered: What are **limitations** of **Mongodb**? Need to have ... What is a table **called** in **MongoDB**? 1,443 Views.

5 answers

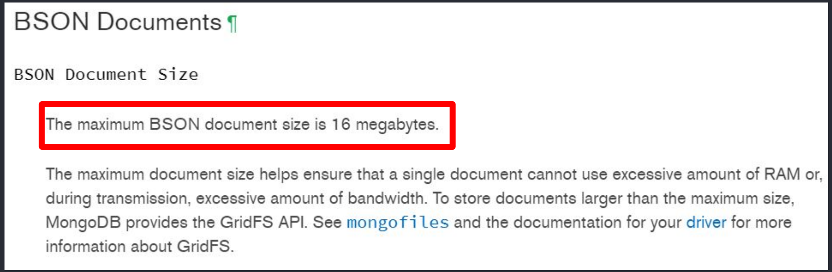
Missing: ~~known~~ | Must include: known

docs.mongodb.com › manual › reference › limits

MongoDB Limits and Thresholds — MongoDB Manual

This document provides a collection of hard and soft **limitations** of the **MongoDB** ... Reindexing operations occur as part of the compact command as **well** as the ...

Missing: ~~known~~ | Must include: known

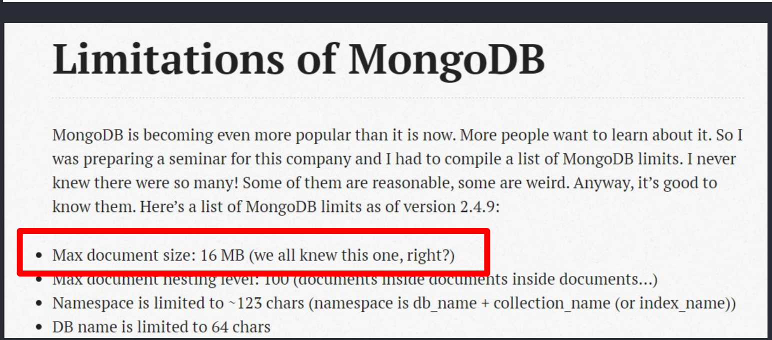


BSON Documents ¶

`BSON Document Size`

The maximum BSON document size is 16 megabytes.

The maximum document size helps ensure that a single document cannot use excessive amount of RAM or, during transmission, excessive amount of bandwidth. To store documents larger than the maximum size, MongoDB provides the GridFS API. See `mongofiles` and the documentation for your driver for more information about GridFS.



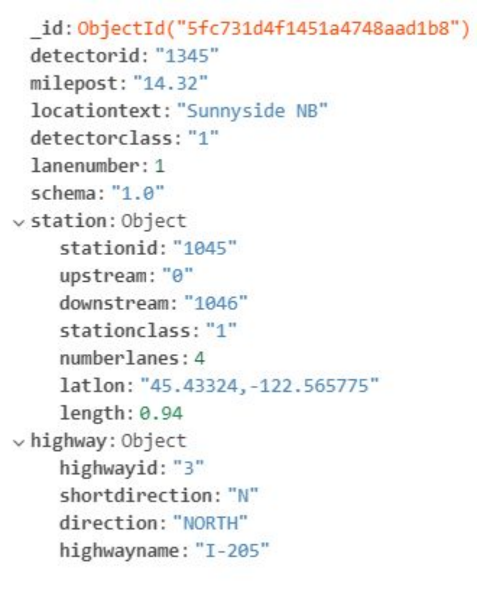
Limitations of MongoDB

MongoDB is becoming even more popular than it is now. More people want to learn about it. So I was preparing a seminar for this company and I had to compile a list of MongoDB limits. I never knew there were so many! Some of them are reasonable, some are weird. Anyway, it's good to know them. Here's a list of MongoDB limits as of version 2.4.9:

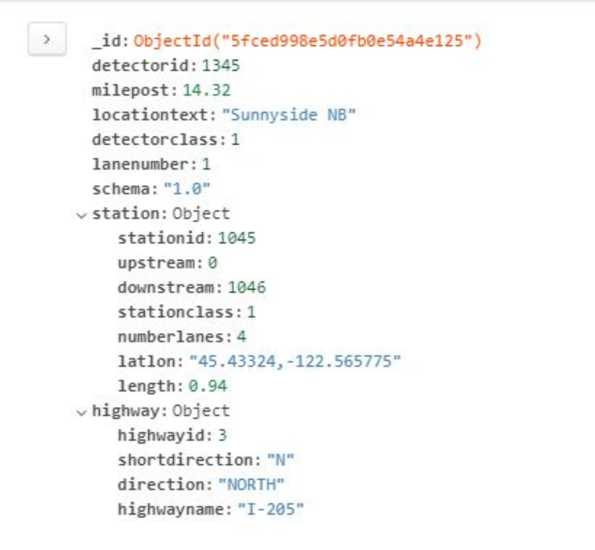
- Max document size: 16 MB (we all knew this one, right?)
- Max document nesting level: 100 (documents inside documents inside documents...)
- Namespace is limited to ~123 chars (namespace is db_name + collection_name (or index_name))
- DB name is limited to 64 chars

Lessons learned

1. Declare each attributes with the proper data type.



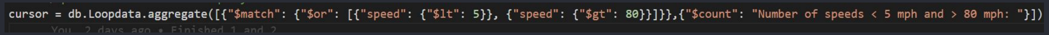
```
 _id: ObjectId("5fc731d4f1451a4748aad1b8")
 detectorid: "1345"
 milepost: "14.32"
 locationtext: "Sunnyside NB"
 detectorclass: "1"
 lanenumber: 1
 schema: "1.0"
v station: Object
   stationid: "1045"
   upstream: "0"
   downstream: "1046"
   stationclass: "1"
   numberlanes: 4
   latlon: "45.43324,-122.565775"
   length: 0.94
v highway: Object
   highwayid: "3"
   shortdirection: "N"
   direction: "NORTH"
   highwayname: "I-205"
```



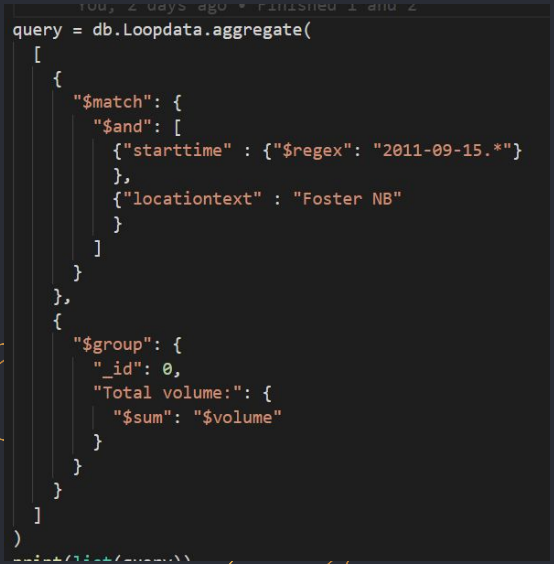
```
 _id: ObjectId("5fced998e5d0fb0e54a4e125")
 detectorid: 1345
 milepost: 14.32
 locationtext: "Sunnyside NB"
 detectorclass: 1
 lanenumber: 1
 schema: "1.0"
v station: Object
   stationid: 1045
   upstream: 0
   downstream: 1046
   stationclass: 1
   numberlanes: 4
   latlon: "45.43324,-122.565775"
   length: 0.94
v highway: Object
   highwayid: 3
   shortdirection: "N"
   direction: "NORTH"
   highwayname: "I-205"
```

Lessons learned

2. Writing query purely using MongoDB could make things look messy. But they are easy to understand.

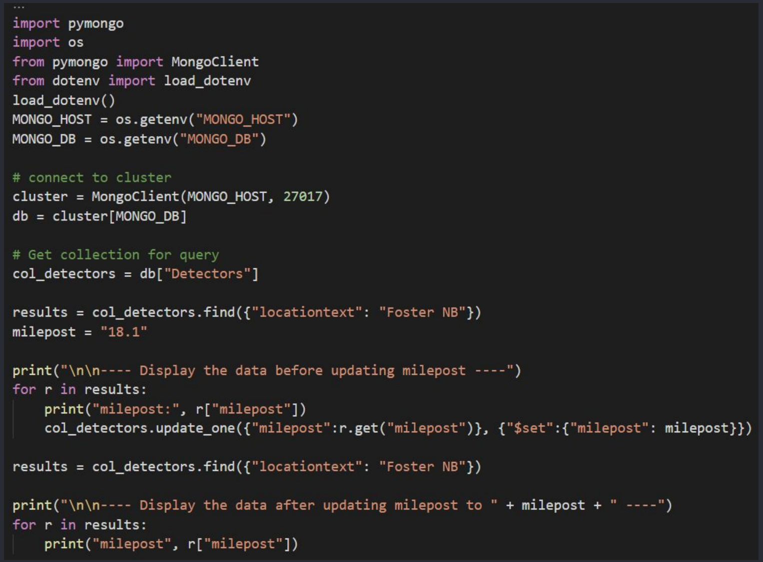


```
cursor = db.Loopdata.aggregate([{"$match": {"$or": [{"speed": {"$lt": 5}}, {"speed": {"$gt": 80}}]}},{"$count": "Number of speeds < 5 mph and > 80 mph: "}])
      You, 2 days ago • Finished 1 and 2
```



```
    You, 2 days ago • Finished 1 and 2
query = db.Loopdata.aggregate(
  [
    {
      "$match": {
        "$and": [
          {"starttime" : {"$regex": "2011-09-15.*"}
          },
          {"locationtext" : "Foster NB"
          }
        ]
      }
    },
    {
      "$group": {
        "_id": 0,
        "Total volume:": {
          "$sum": "$volume"
        }
      }
    }
  ]
)
print(list(query))
```

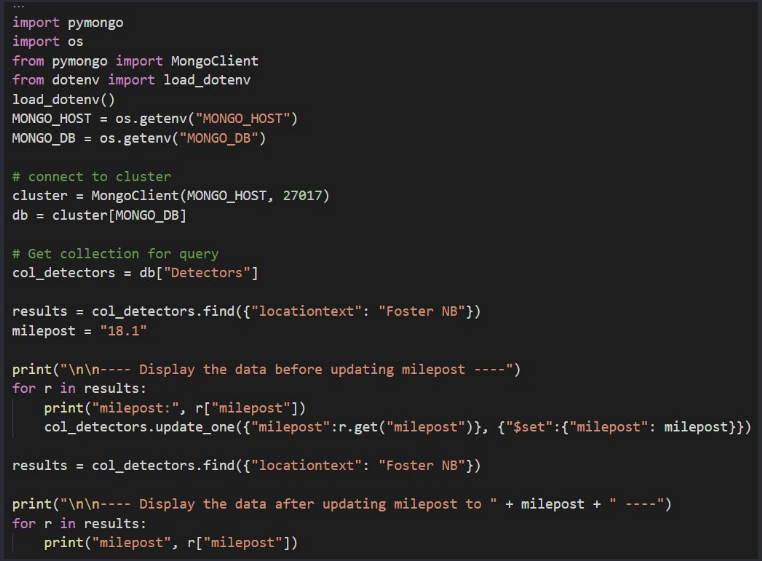
VS



```
...
import pymongo
import os
from pymongo import MongoClient
from dotenv import load_dotenv
load_dotenv()
MONGO_HOST = os.getenv("MONGO_HOST")
MONGO_DB = os.getenv("MONGO_DB")

# connect to cluster
cluster = MongoClient(MONGO_HOST, 27017)
db = cluster[MONGO_DB]

# Get collection for query
col_detectors = db["Detectors"]

results = col_detectors.find({"locationtext": "Foster NB"})
milepost = "18.1"

print("\n\n---- Display the data before updating milepost ----")
for r in results:
    print("milepost:", r["milepost"])
    col_detectors.update_one({"milepost":r.get("milepost")}, {"$set":{"milepost": milepost}})

results = col_detectors.find({"locationtext": "Foster NB"})

print("\n\n---- Display the data after updating milepost to " + milepost + " ----")
for r in results:
    print("milepost", r["milepost"])
```

Lessons learned

3. Solve complicated questions with combination of python and queries.



```
import pymongo
import os
from pymongo import MongoClient
from dotenv import load_dotenv
load_dotenv()
MONGO_HOST = os.getenv("MONGO_HOST")
MONGO_DB = os.getenv("MONGO_DB")

# connect to cluster
cluster = MongoClient(MONGO_HOST, 27017)
db = cluster[MONGO_DB]

# Get collection for query
col_detectors = db["Detectors"]

results = col_detectors.find({"locationtext": "Foster NB"})
milepost = "18.1"

print("\n\n---- Display the data before updating milepost ----")
for r in results:
    print("milepost:", r["milepost"])
    col_detectors.update_one({"milepost":r.get("milepost")}, {"$set":{"milepost": milepost}})

results = col_detectors.find({"locationtext": "Foster NB"})

print("\n\n---- Display the data after updating milepost to " + milepost + " ----")
for r in results:
    print("milepost", r["milepost"])
```

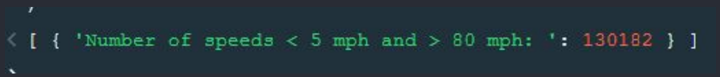
Q & A

Appendix

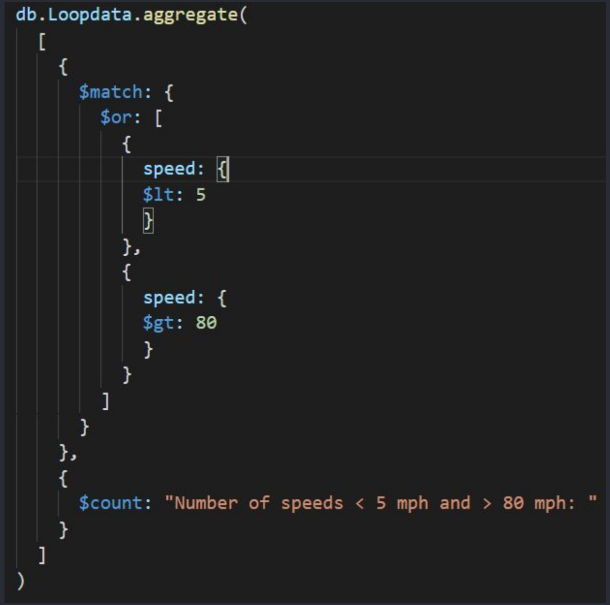
Project github link: youn0125/pdx-cs-cloud-cluster (github.com)

Appendix

Q1: Count low speeds and high speeds:
Find the number of speeds < 5 mph and > 80 mph in the data set.



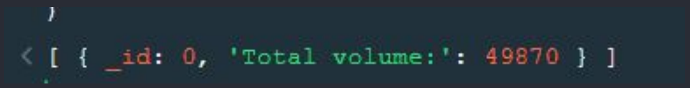
```
< [ { 'Number of speeds < 5 mph and > 80 mph: ': 130182 } ]
```



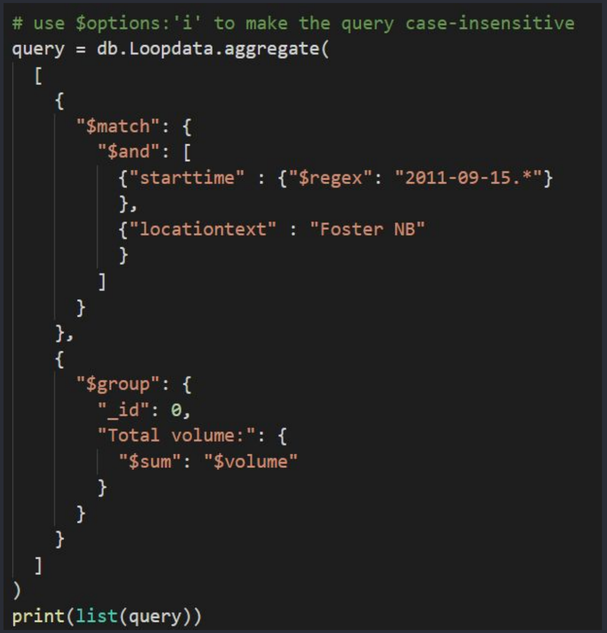
```
db.Loopdata.aggregate(
  [
    {
      $match: {
        $or: [
          {
            speed: {
            $lt: 5
            }
          },
          {
            speed: {
            $gt: 80
            }
          }
        ]
      }
    },
    {
      $count: "Number of speeds < 5 mph and > 80 mph: "
    }
  ]
)
```

Appendix

Q2: Volume: Find the total volume for the station Foster NB for Sept 15, 2011.



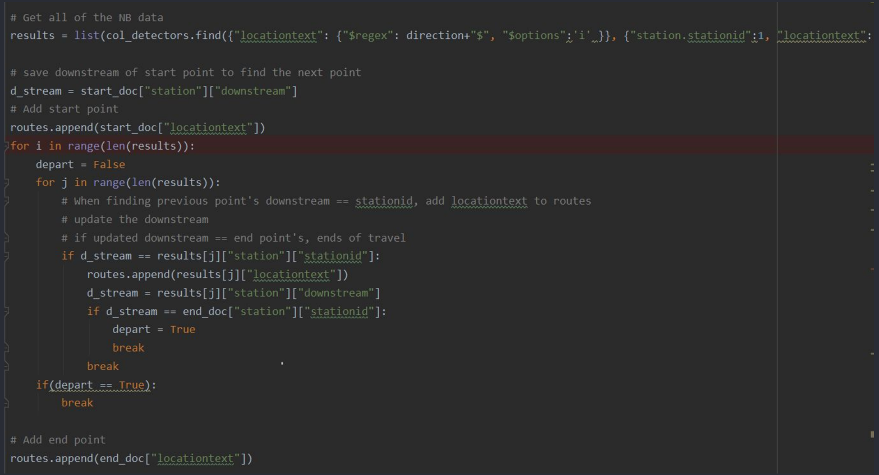
```
< [ { _id: 0, 'Total volume:': 49870 } ]
```



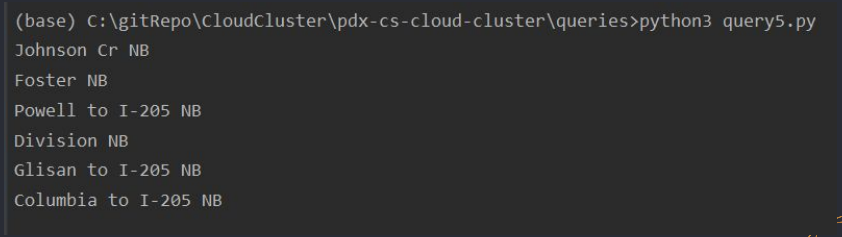
```
# use $options:'i' to make the query case-insensitive
query = db.Loopdata.aggregate(
  [
    {
      "$match": {
        "$and": [
          {"starttime" : {"$regex": "2011-09-15.*"}
          },
          {"locationtext" : "Foster NB"
          }
        ]
      }
    },
    {
      "$group": {
        "_id": 0,
        "Total volume:": {
          "$sum": "$volume"
        }
      }
    }
  ]
)
print(list(query))
```

Appendix

Q5. Route Finding: Find a route from Johnson Creek to Columbia Blvd on I-205 NB using the upstream and downstream fields.



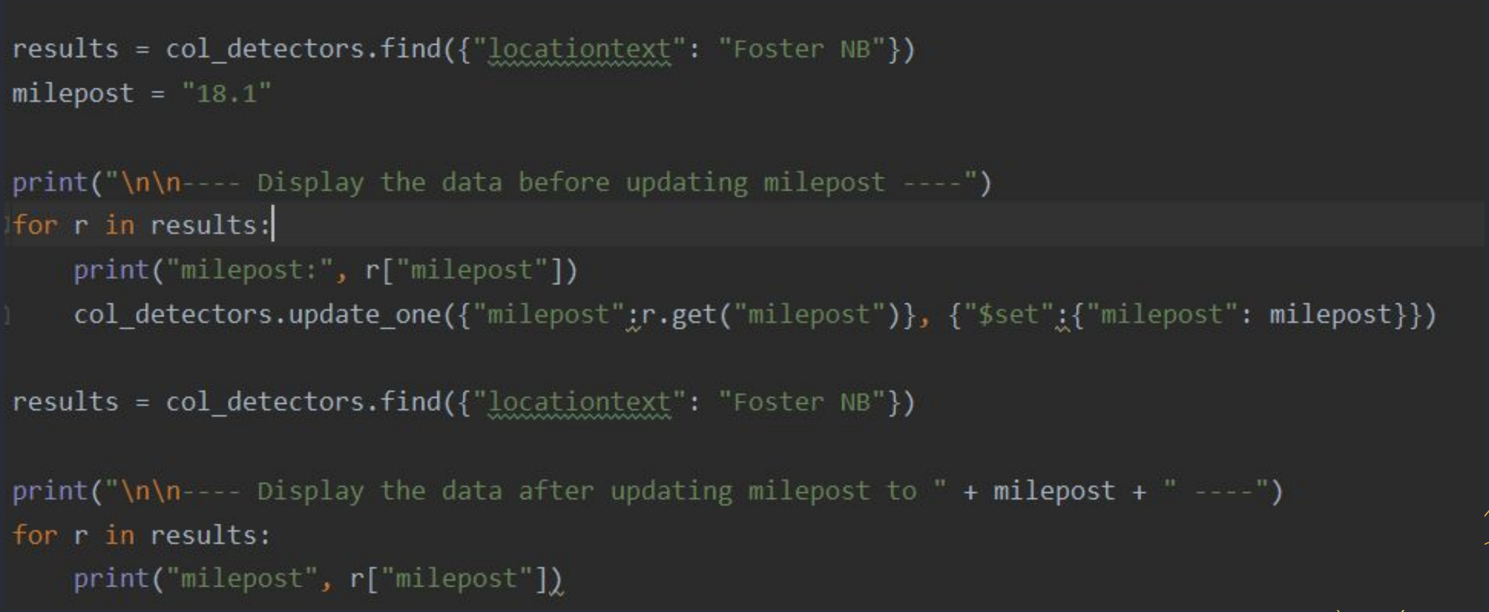
```python
# Get all of the NB data
results = list(col_detectors.find({"locationtext": {"$regex": direction+"$", "$options":'i' }}, {"station.stationid":1, "locationtext":

# save downstream of start point to find the next point
d_stream = start_doc["station"]["downstream"]
# Add start point
routes.append(start_doc["locationtext"])
for i in range(len(results)):
    depart = False
    for j in range(len(results)):
        # When finding previous point's downstream == stationid, add locationtext to routes
        # update the downstream
        # if updated downstream == end point's, ends of travel
        if d_stream == results[j]["station"]["stationid"]:
            routes.append(results[j]["locationtext"])
            d_stream = results[j]["station"]["downstream"]
            if d_stream == end_doc["station"]["stationid"]:
                depart = True
                break
            break
    if(depart == True):
        break

# Add end point
routes.append(end_doc["locationtext"])
```



```
(base) C:\gitRepo\CloudCluster\pdx-cs-cloud-cluster\queries>python3 query5.py
Johnson Cr NB
Foster NB
Powell to I-205 NB
Division NB
Glisan to I-205 NB
Columbia to I-205 NB
```

Appendix

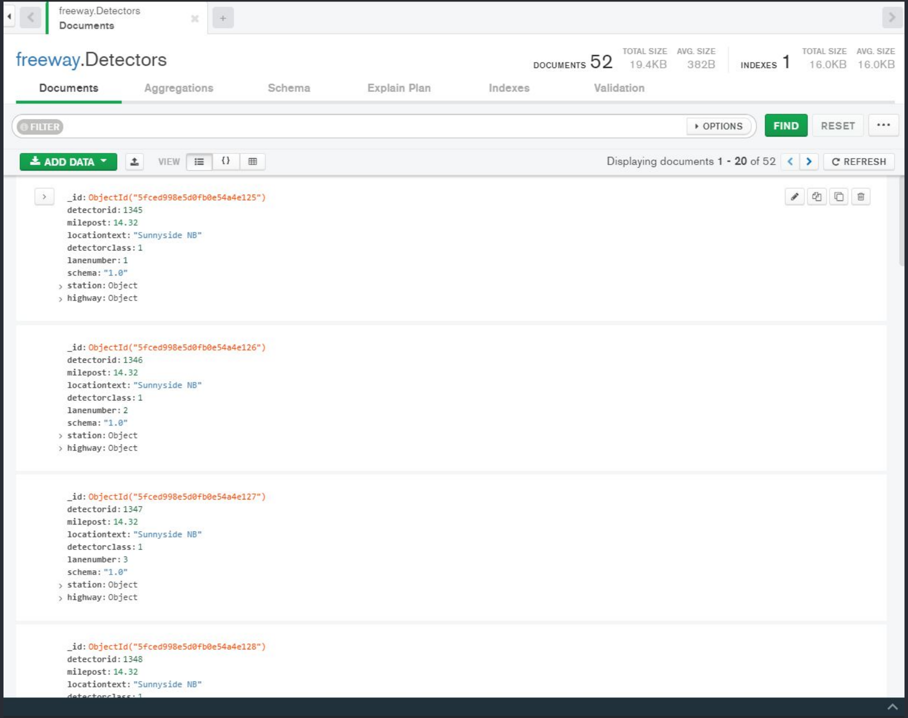
<u>Q6: Update</u>: Change the milepost of the Foster NB station to 22.6.



```
results = col_detectors.find({"locationtext": "Foster NB"})
milepost = "18.1"

print("\n\n---- Display the data before updating milepost ----")
for r in results:
    print("milepost:", r["milepost"])
    col_detectors.update_one({"milepost":r.get("milepost")}, {"$set":{"milepost": milepost}})

results = col_detectors.find({"locationtext": "Foster NB"})

print("\n\n---- Display the data after updating milepost to " + milepost + " ----")
for r in results:
    print("milepost", r["milepost"])
```

Appendix

Data size:



freeway.Detectors

DOCUMENTS 52 TOTAL SIZE 19.4KB AVG. SIZE 382B

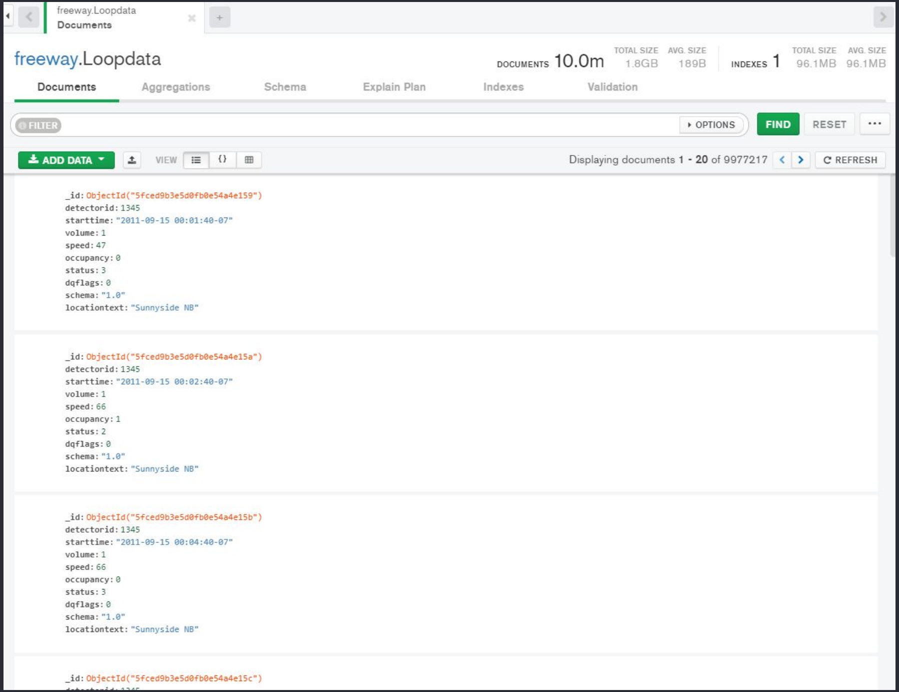
INDEXES 1 TOTAL SIZE 16.0KB AVG. SIZE 16.0KB

Displaying documents 1 - 20 of 52

```
_id: ObjectId("5fced998e5d0fb0e54a4e125")
detectorid: 1345
milepost: 14.32
locationtext: "Sunnyside NB"
detectorclass: 1
lanenumber: 1
schema: "1.0"
> station: Object
> highway: Object

_id: ObjectId("5fced998e5d0fb0e54a4e126")
detectorid: 1346
milepost: 14.32
locationtext: "Sunnyside NB"
detectorclass: 1
lanenumber: 2
schema: "1.0"
> station: Object
> highway: Object

_id: ObjectId("5fced998e5d0fb0e54a4e127")
detectorid: 1347
milepost: 14.32
locationtext: "Sunnyside NB"
detectorclass: 1
lanenumber: 3
schema: "1.0"
> station: Object
> highway: Object

_id: ObjectId("5fced998e5d0fb0e54a4e128")
detectorid: 1348
milepost: 14.32
locationtext: "Sunnyside NB"
detectorclass: 1
```

Appendix

Data size:



freeway.Loopdata

DOCUMENTS 10.0m TOTAL SIZE 1.8GB AVG. SIZE 189B

INDEXES 1 TOTAL SIZE 96.1MB AVG. SIZE 96.1MB

Displaying documents 1 - 20 of 9977217

```
_id: ObjectId("5fced9b3e5d0fb0e54a4e159")
detectorid: 1345
starttime: "2011-09-15 00:01:40-07"
volume: 1
speed: 47
occupancy: 0
status: 3
dqflags: 0
schema: "1.0"
locationtext: "Sunnyside NB"

_id: ObjectId("5fced9b3e5d0fb0e54a4e15a")
detectorid: 1345
starttime: "2011-09-15 00:02:40-07"
volume: 1
speed: 66
occupancy: 1
status: 2
dqflags: 0
schema: "1.0"
locationtext: "Sunnyside NB"

_id: ObjectId("5fced9b3e5d0fb0e54a4e15b")
detectorid: 1345
starttime: "2011-09-15 00:04:40-07"
volume: 1
speed: 66
occupancy: 0
status: 3
dqflags: 0
schema: "1.0"
locationtext: "Sunnyside NB"

_id: ObjectId("5fced9b3e5d0fb0e54a4e15c")
```