Report

Assignment 3 - MongoDB

Group: 35
Students: Albert Lesniewski, Thomas Tran, Md Anwarul Hasan

Introduction

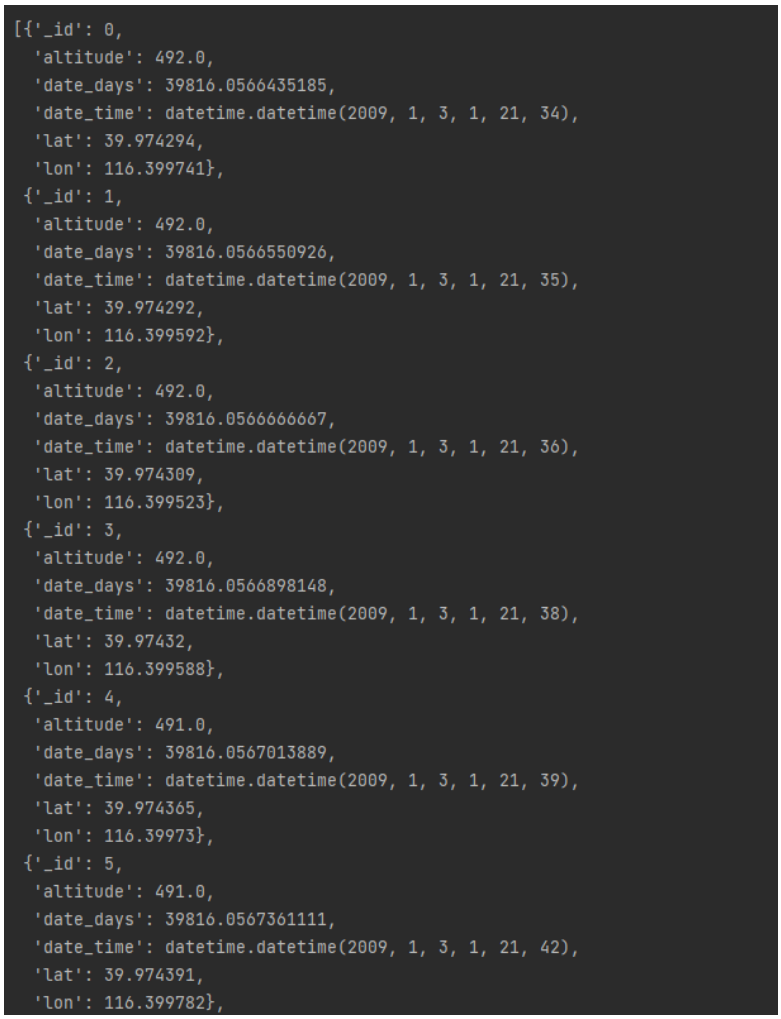
In this assignment we were tasked to process and import data into Mongodb database and query said database with different queries. Because the tasks are very similar to assignment 2 tasks, we reused most of assignment 2 code in this assignment. The major difference was that instead of dealing with string queries, we had to build dictionaries with fields, this required some changes to our assignment 2 code. Writing the queries was by far the most challenging task, more challenging than in assignment 2, since none of us had any experience with writing queries for mongodb or any NoSQL database for that matter. The group worked well together, we worked in a pair programming style for the data processing and insertion part, then for the queries, we split the different queries evenly between ourselves and everyone did their part.

Results

Part 1

```
[{'_id': '135', 'has_labels': False},
 {'_id': '132', 'has_labels': False},
 {'_id': '104', 'has_labels': True},
 {'_id': '103', 'has_labels': False},
 {'_id': '168', 'has_labels': False},
 {'_id': '157', 'has_labels': False},
 {'_id': '150', 'has_labels': False},
 {'_id': '159', 'has_labels': False},
 {'_id': '166', 'has_labels': False},
 {'_id': '161', 'has_labels': True}]
```

```
[{'_id': 1,
  'end_date_time': datetime.datetime(2009, 1, 3, 5, 40, 31),
  'start_date_time': datetime.datetime(2009, 1, 3, 1, 21, 34),
  'transportation_mode': None},
 {'_id': 2,
  'end_date_time': datetime.datetime(2009, 1, 2, 4, 41, 5),
  'start_date_time': datetime.datetime(2009, 1, 2, 4, 31, 27),
  'transportation_mode': None},
 {'_id': 3,
  'end_date_time': datetime.datetime(2009, 1, 27, 4, 50, 32),
  'start_date_time': datetime.datetime(2009, 1, 27, 3, 0, 4),
  'transportation_mode': None},
 {'_id': 4,
  'end_date_time': datetime.datetime(2009, 1, 10, 4, 42, 47),
  'start_date_time': datetime.datetime(2009, 1, 10, 1, 19, 47),
  'transportation_mode': None},
 {'_id': 5,
  'end_date_time': datetime.datetime(2009, 1, 14, 12, 30, 53),
  'start_date_time': datetime.datetime(2009, 1, 14, 12, 17, 57),
  'transportation_mode': None},
 {'_id': 6,
  'end_date_time': datetime.datetime(2009, 1, 12, 2, 14, 1),
  'start_date_time': datetime.datetime(2009, 1, 12, 1, 41, 22),
  'transportation_mode': None},
 {'_id': 7,
  'end_date_time': datetime.datetime(2008, 12, 24, 15, 26, 45),
  'start_date_time': datetime.datetime(2008, 12, 24, 14, 42, 7),
  'transportation_mode': None},
 {'_id': 8,
  'end_date_time': datetime.datetime(2008, 12, 28, 12, 19, 32),
  'start_date_time': datetime.datetime(2008, 12, 28, 10, 36, 5),
  'transportation_mode': None},
 {'_id': 9,
  'end_date_time': datetime.datetime(2010, 2, 15, 12, 22, 33),
  'start_date_time': datetime.datetime(2010, 2, 15, 10, 56, 35),
  'transportation_mode': None},
 {'_id': 10,
  'end_date_time': datetime.datetime(2010, 5, 1, 0, 35, 31),
  'start_date_time': datetime.datetime(2010, 4, 30, 23, 38, 1),
  'transportation_mode': None}]
```



```
[{'_id': 0,
  'altitude': 492.0,
  'date_days': 39816.0566435185,
  'date_time': datetime.datetime(2009, 1, 3, 1, 21, 34),
  'lat': 39.974294,
  'lon': 116.399741},
 {'_id': 1,
  'altitude': 492.0,
  'date_days': 39816.0566550926,
  'date_time': datetime.datetime(2009, 1, 3, 1, 21, 35),
  'lat': 39.974292,
  'lon': 116.399592},
 {'_id': 2,
  'altitude': 492.0,
  'date_days': 39816.0566666667,
  'date_time': datetime.datetime(2009, 1, 3, 1, 21, 36),
  'lat': 39.974309,
  'lon': 116.399523},
 {'_id': 3,
  'altitude': 492.0,
  'date_days': 39816.0566898148,
  'date_time': datetime.datetime(2009, 1, 3, 1, 21, 38),
  'lat': 39.97432,
  'lon': 116.399588},
 {'_id': 4,
  'altitude': 491.0,
  'date_days': 39816.0567013889,
  'date_time': datetime.datetime(2009, 1, 3, 1, 21, 39),
  'lat': 39.974365,
  'lon': 116.39973},
 {'_id': 5,
  'altitude': 491.0,
  'date_days': 39816.0567361111,
  'date_time': datetime.datetime(2009, 1, 3, 1, 21, 42),
  'lat': 39.974391,
  'lon': 116.399782},
```

As seen in the image above, we successfully imported data into the three tables. user, activity and track_point. The import time is around 5-8 min on our computers. The user collection contains three fields: _id, has_label and activity_ref. Activity_ref is a list of activity ids that references the activities that were registered by each individual user.

The activity collection contains six fields: _id, transportation_mode, start_date_time, end_date_time, user_ref, track_point_ref.
User_ref field is an user id that refers to the user that the activity was registered by, track_point_ref is a list of track point ids that references the track points that belong to each activity.

The track_point collection contains six fields: _id, lat, lon, altitude, date_days, date_time.

Part 2

Task 1:

How many users, activities and trackpoints are there in the dataset

Result:

```
'Number of users: 182'
'Number of activities: 16048'
'Number of track points: 9681756'
```

As shown in image above, we successfully count the number of inserted users, activities and track points. We used the count_documents function to count the documents in each collection.

Task 2:

Find the average number of activities per user

Result:

```
'Average number of activities per user: 88.17582417582418'
```

As shown in image above, we computed the average number of activities per user by simply dividing the count of all activities by the count of all users.

Task 3:

Find the top 20 users with the highest number of activities.

Result:

```
result = list(user_collection.aggregate([
    {'$unwind': "$activity_ref"},
    {'$group': {'_id': "$_id", 'activity_ref': {'$push':"$activity_ref"},'size': {'$sum':1}}},
    {'$sort': {'size': -1}},
    {'$limit': 20},
    {'$project': {'_id': 1}}
]))
```

```
'Top 20 users with most activities:'
[{'_id': '128'},
 {'_id': '153'},
 {'_id': '025'},
 {'_id': '163'},
 {'_id': '062'},
 {'_id': '144'},
 {'_id': '041'},
 {'_id': '085'},
 {'_id': '004'},
 {'_id': '140'},
 {'_id': '167'},
 {'_id': '068'},
 {'_id': '017'},
 {'_id': '003'},
 {'_id': '014'},
 {'_id': '126'},
 {'_id': '030'},
 {'_id': '112'},
 {'_id': '011'},
 {'_id': '039'}]
```

As shown in the image above, we were able to query mongodb for the top 20 users with the most activities. We used mongodb aggregation framework to do this.

Task 4:

Find all users who have taken a taxi

Result:

```
result = user_collection.aggregate([
    {'$lookup': {'from': 'activity', 'localField': 'activity_ref', 'foreignField': '_id', 'as': 'u_a_join'}},
    {'$unwind': '$u_a_join'},
    {'$match': {'u_a_join.transportation_mode': 'taxi'}},
    {'$project': {'_id': 1}},
    {'$group': {'_id': '$_id'}}
])
```

```
'All users that have used a taxi:'
[{'_id': '085'},
 {'_id': '078'},
 {'_id': '098'},
 {'_id': '111'},
 {'_id': '128'},
 {'_id': '163'},
 {'_id': '062'},
 {'_id': '080'},
 {'_id': '010'},
 {'_id': '058'}]
```

As shown in the image above, we managed to query mongodb directly to get all the users that have registered to have used a taxi as a transportation mode. We used the aggregation framework from mongodb. We used a lookup method in the pipeline which works similar to a left outer join, we used this functionality to join the user and activity collections.

Task 5:

Find all types of transportation modes and count how many activities that are tagged with these transportation mode labels. Do not count the rows where the mode is null.

Result:

```
[{'_id': 'walk', 'count': 480},
 {'_id': 'car', 'count': 419},
 {'_id': 'bike', 'count': 263},
 {'_id': 'bus', 'count': 199},
 {'_id': 'subway', 'count': 133},
 {'_id': 'taxi', 'count': 37},
 {'_id': 'airplane', 'count': 3},
 {'_id': 'train', 'count': 2},
 {'_id': 'run', 'count': 1},
 {'_id': 'boat', 'count': 1}]
```

Task 6:

a) Find the year with the most activities.

Result:

```
def task6a(self):
    activity_collection = self.db["activity"]
    result = list(activity_collection.aggregate([
        {'$project': {
            'year': {'$year': '$start_date_time'}
            }
        },
        {'$group': {
            '_id': '$year',
            'count': {'$sum': 1}
            }
        },
        {'$sort': {'count': -1}},
        {'$limit': 1},
    ]))
    pprint(result)
```

```
[{'_id': 2008, 'count': 5895}]
```

According to the requirement, we counted the activity of each year, arranged it in descending order and finally printed the top row to find the outcome in MongoDB.

b) Is this also the year with the most recorded hours?

Result:

```
def task6b(self):
    activity_collection = self.db["activity"]
    result = list(activity_collection.aggregate([
        {'$project': {
            'year': {'$year': '$start_date_time'},
            'start_time': '$start_date_time',
            'end_time': '$end_date_time'
            }
        },
        {'$group': {'_id': '$year',
                    'total_hour': {
                        '$sum': {
                            '$dateDiff': {
                                'startDate': '$start_time',
                                'endDate': '$end_time',
                                'unit': 'hour'
                                }
                            }
                        }
                    }
                }, {
        '$sort': {'total_hour': -1}},
    ]))
    pprint(result)
```

```
[{'_id': 2009, 'total_hour': 11636},
 {'_id': 2008, 'total_hour': 9105},
 {'_id': 2007, 'total_hour': 2324},
 {'_id': 2010, 'total_hour': 1432},
 {'_id': 2011, 'total_hour': 1130},
 {'_id': 2012, 'total_hour': 721},
 {'_id': 2000, 'total_hour': 0}]
```

According to the requirement, we first projected the required fields from activity and then summed the activity in unit of hours. After sorting the data according to hours we found our required outcome and observed that 2008 (the year with most activities) is not the year with the most recorded hours. The year with the most recorded hours is 2009.

Task 7:

```
result = activity_collection.aggregate([
    {'$project': {'_id': 1, 'user_ref': 1, 'transportation_mode': 1,
                  'start_year': {'$year': '$start_date_time'},
                  'track_point_ref': 1}},
    {'$match': {'user_ref': '112', 'transportation_mode': 'walk', 'start_year': 2008}},
    {'$lookup': {'from': 'track_point', 'localField': 'track_point_ref', 'foreignField': '_id', 'as': 'tp'}},
    {'$unwind': '$tp'},
    {'$project': {'_id': 1, 'lon': '$tp.lon', 'lat': '$tp.lat'}}
])
```

```
Total distance walked by user 112 in 2008 is: 114.92625793252165km
```

As shown in the image above, we computed the total distance walked by user 112 in 2008 by first querying mongodb database for all the trackpoints for all the activities for user 112 which start date was registered in 2008, then we computed the haversine distance between the trackpoints for each activity using python code.

Task 8:

Find the top 20 users who have gained the most altitude meters.

Result:

```
[{'total_meters_gained_per_user': 152456732.7288, 'user_id': '153'},
 {'total_meters_gained_per_user': 130042696.58160001, 'user_id': '128'},
 {'total_meters_gained_per_user': 71684959.1952, 'user_id': '025'},
 {'total_meters_gained_per_user': 65785983.744, 'user_id': '041'},
 {'total_meters_gained_per_user': 64447674.6096, 'user_id': '068'},
 {'total_meters_gained_per_user': 55505273.5968, 'user_id': '004'},
 {'total_meters_gained_per_user': 46120403.8728, 'user_id': '163'},
 {'total_meters_gained_per_user': 45679311.3336, 'user_id': '085'},
 {'total_meters_gained_per_user': 45374087.052, 'user_id': '014'},
 {'total_meters_gained_per_user': 45095304.475200005, 'user_id': '003'},
 {'total_meters_gained_per_user': 44999029.4328, 'user_id': '017'},
 {'total_meters_gained_per_user': 44544422.3664, 'user_id': '013'},
 {'total_meters_gained_per_user': 41689438.7952, 'user_id': '030'},
 {'total_meters_gained_per_user': 34991901.3648, 'user_id': '084'},
 {'total_meters_gained_per_user': 34953733.3944, 'user_id': '062'},
 {'total_meters_gained_per_user': 34143914.8464, 'user_id': '167'},
 {'total_meters_gained_per_user': 33633270.6048, 'user_id': '168'},
 {'total_meters_gained_per_user': 31142187.144, 'user_id': '144'},
 {'total_meters_gained_per_user': 29952683.503200002, 'user_id': '126'},
 {'total_meters_gained_per_user': 29536106.3328, 'user_id': '002'}]
```

Task 9:

Find all users who have invalid activities, and the number of invalid activities per user
Result:

The images shows a dictionary where the key is the user id and value is the number of invalid activities.

```
{'000': 101,
 '001': 45,
 '002': 98,
 '003': 179,
 '004': 219,
 '005': 45,
 '006': 17,
 '007': 30,
 '008': 16,
 '009': 31,
 '010': 50,
 '011': 32,
 '012': 43,
 '013': 29,
 '014': 118,
 '015': 46,
 '016': 20,
 '017': 129,
 '018': 27,
 '019': 31,
 '020': 20,
 '021': 7,
 '022': 55,
 '023': 11,
 '024': 27,
 '025': 263,
 '026': 18,
 '027': 2,
 '028': 36,
 '029': 25,
 '030': 112,
 '031': 3,
 '032': 12,
 '033': 2,
 '034': 88,
 '035': 23,
 '036': 34,
 '037': 100,
 '038': 58,
 '039': 147,
 '040': 17,
 '041': 201,
 '042': 55,
 '043': 21,
 '044': 32,
 '045': 7,
 '046': 13,
 '047': 6,
 '048': 1,
 '050': 8,
 '051': 36,
 '052': 44,
 '053': 7,
 '054': 2,
 '055': 15,
 '056': 7,
```

```
'055': 15,      '113': 1,
'056': 7,       '114': 3,
'057': 16,      '115': 58,
'058': 13,      '117': 3,
'059': 5,       '118': 3,
'060': 1,       '119': 22,
'061': 12,      '121': 4,
'062': 249,     '122': 6,
'063': 8,       '123': 3,
'064': 7,       '124': 4,
'065': 26,      '125': 25,
'066': 6,       '126': 105,
'067': 33,      '127': 4,
'068': 139,     '128': 720,
'069': 6,       '129': 6,
'070': 5,       '130': 8,
'071': 29,      '131': 10,
'072': 2,       '132': 3,
'073': 18,      '133': 4,
'074': 19,      '134': 31,
'075': 6,       '135': 5,
'076': 8,       '136': 6,
'077': 3,       '138': 10,
'078': 19,      '139': 12,
'079': 2,       '140': 86,
'080': 6,       '141': 1,
'081': 16,      '142': 52,
'082': 27,      '144': 157,
'083': 15,      '145': 5,
'084': 99,      '146': 7,
'085': 184,     '147': 30,
'086': 5,       '150': 16,
'087': 3,       '151': 1,
'088': 11,      '152': 2,
'089': 40,      '153': 557,
'090': 3,       '154': 14,
'091': 63,      '155': 30,
'092': 101,     '157': 9,
'093': 4,       '158': 9,
'094': 16,      '159': 5,
'095': 4,       '161': 7,
'096': 35,      '162': 9,
'097': 14,      '163': 233,
'098': 5,       '164': 6,
'099': 11,      '165': 2,
'100': 3,       '166': 2,
'101': 46,      '167': 134,
'102': 13,      '168': 19,
'103': 24,      '169': 9,
'104': 97,      '170': 2,
'105': 9,       '171': 3,
'106': 3,       '172': 9,
'107': 1,       '173': 5,
'108': 5,       '174': 54,
'109': 3,       '175': 4,
'110': 17,      '176': 8,
'111': 26,      '179': 28,
'112': 67,      '180': 2,
                '181': 14}
```

Task 10:

```
result = activity_collection.aggregate([
    {'$lookup': {'from': 'track_point', 'localField': 'track_point_ref', 'foreignField': '_id', 'as': 'tp'}},
    {'$unwind': '$tp'},
    {'$project': {'lat': '$tp.lat', 'lon': '$tp.lon', 'uid': '$user_ref'}},
    {'$match': {'lat': {'$gt': 39.915, '$lt': 39.917}, 'lon': {'$gt': 116.396, '$lt': 116.398}}},
    {'$group': {'_id': '$uid'}}
])
```

```
Number of users that registered activity in the Forbidden City: 4
```

As shown in the image above we managed to find 4 users that have registered an activity in the Forbidden city (where we used the 0.001 as threshold for the coordinates).

Task 11:

Find all users who have registered transportation_mode and their most used transportation_mode.

Result:

```
[{'most_used_transportation_mode': 'taxi', 'user_id': '010'},
 {'most_used_transportation_mode': 'bike', 'user_id': '020'},
 {'most_used_transportation_mode': 'walk', 'user_id': '021'},
 {'most_used_transportation_mode': 'bus', 'user_id': '052'},
 {'most_used_transportation_mode': 'bike', 'user_id': '056'},
 {'most_used_transportation_mode': 'taxi', 'user_id': '058'},
 {'most_used_transportation_mode': 'walk', 'user_id': '060'},
 {'most_used_transportation_mode': 'walk', 'user_id': '062'},
 {'most_used_transportation_mode': 'bike', 'user_id': '064'},
 {'most_used_transportation_mode': 'bike', 'user_id': '065'},
 {'most_used_transportation_mode': 'walk', 'user_id': '067'},
 {'most_used_transportation_mode': 'bike', 'user_id': '069'},
 {'most_used_transportation_mode': 'walk', 'user_id': '073'},
 {'most_used_transportation_mode': 'walk', 'user_id': '075'},
 {'most_used_transportation_mode': 'car', 'user_id': '076'},
 {'most_used_transportation_mode': 'walk', 'user_id': '078'},
 {'most_used_transportation_mode': 'taxi', 'user_id': '080'},
 {'most_used_transportation_mode': 'bike', 'user_id': '081'},
 {'most_used_transportation_mode': 'walk', 'user_id': '082'},
 {'most_used_transportation_mode': 'walk', 'user_id': '084'},
 {'most_used_transportation_mode': 'walk', 'user_id': '085'},
 {'most_used_transportation_mode': 'car', 'user_id': '086'},
 {'most_used_transportation_mode': 'walk', 'user_id': '087'},
 {'most_used_transportation_mode': 'car', 'user_id': '089'},
 {'most_used_transportation_mode': 'walk', 'user_id': '091'},
 {'most_used_transportation_mode': 'walk', 'user_id': '092'},
 {'most_used_transportation_mode': 'bike', 'user_id': '097'},
 {'most_used_transportation_mode': 'taxi', 'user_id': '098'},
 {'most_used_transportation_mode': 'car', 'user_id': '101'},
 {'most_used_transportation_mode': 'bike', 'user_id': '102'},
 {'most_used_transportation_mode': 'walk', 'user_id': '107'},
 {'most_used_transportation_mode': 'walk', 'user_id': '108'},
 {'most_used_transportation_mode': 'taxi', 'user_id': '111'},
 {'most_used_transportation_mode': 'walk', 'user_id': '112'},
 {'most_used_transportation_mode': 'car', 'user_id': '115'},
 {'most_used_transportation_mode': 'walk', 'user_id': '117'},
 {'most_used_transportation_mode': 'bike', 'user_id': '125'},
 {'most_used_transportation_mode': 'bike', 'user_id': '126'},
 {'most_used_transportation_mode': 'car', 'user_id': '128'},
 {'most_used_transportation_mode': 'walk', 'user_id': '136'},
 {'most_used_transportation_mode': 'bike', 'user_id': '138'},
 {'most_used_transportation_mode': 'bike', 'user_id': '139'},
 {'most_used_transportation_mode': 'walk', 'user_id': '144'},
 {'most_used_transportation_mode': 'walk', 'user_id': '153'},
 {'most_used_transportation_mode': 'walk', 'user_id': '161'},
 {'most_used_transportation_mode': 'bike', 'user_id': '163'},
 {'most_used_transportation_mode': 'bike', 'user_id': '167'},
 {'most_used_transportation_mode': 'bus', 'user_id': '175'}]
```

Discussion

We decided to have two-way-referencing for the relation between user and activity, and activity and track point. Calculated in task 2, the average activities per user is around 88. Embedded documents of activities in user would be a poor choice because it would lead to a big nested list of documents. We therefore chose to have a list of activity ids instead to reference the associated activities.

Activity has references to user and track point. The user reference is just the id to the user. It would be possible to embed the document instead, because an activity can only be associated with one user. However, in most use cases, we would only need the user id when performing queries against the activity collection. Therefore, the user id alone would suffice. Due to the same reasons as described with the activity reference from user, the reference to track point from activity is a list of the track point ids related to an activity. As described in the assignment, we only included the activities and track points that had less than or equal to 2500 track points. We concluded that because the number of track points per activity wouldn't exceed 2500, we could reference the track points by id.

Track point has a reference to its related activity id. One track point is only associated with one activity, so we could have embedded the activity document in track point. However, looking at the number of track points, embedding the activity documents would lead to lots of duplicates. But they could have saved us time by eliminating the need of accessing the activities through the activity collection. However, there are lots of queries where we would need the activities by themselves, which would be difficult if they were embedded . So embedding them into the track point collection would take up unnecessary space.

Some minor changes were made from the presented schema in the assignment. We changed the datatype of the altitude in track point from int to float, because some of the altitudes from the dataset had decimal values.

Through this assignment, we learned the basics of document oriented databases, and saw practical differences between relational and document oriented databases, like how the schema design and queries are done. We also learned how to do queries in MongoDB with the help of the aggregation pipeline which was similar to lambda expressions in Java.

MySQL vs MongoDB

MySQL is a relation SQL database that uses a row oriented format with a strict schema, MongoDB on the other hand is a NoSQL database that stores data as documents in collection, those collection don't enforce any strict schema on the documents, as they are very much similar to JSON documents. The document model provides more data locality, but also requires that all the data should be stored in the document, as performing joins on the different documents is more difficult because there are no constraints on the schema of the documents, so it is possible that a document may refer to another document that don't exists, or does not contain the fields we would expect, it is the user that must verify that the formats are correct.

In general we preferred to work with MySQL, it is quite possible that we have a strong bias towards MySQL as we had much more experience with MySQL and relational SQL databases from before, which made it much easier and intuitive to solve the tasks compared to MongoDB.

As mentioned briefly above, MongoDB provides no schema enforcement, which makes the collections (aka. tables) much more flexible and easier to change, but it also means that there is no constraint enforcement on what fields exist in a document and what the structure of the document even is. As a group we found it easier to work with the programmatic style of mongodb than having to build long strings to insert data into mysql.

Feedback

Decent assignment, that involved different parts from data preprocessing, schema creation and writing queries. It was interesting to get some hands-on experience in NoSQL databases since we did not have any from before.