# Assignment 3 - MongoDB

**Group: 64**

**Students: Trym Grande, Thomas Bjerke**

# Introduction

The handout for this exercise consisted of a dataset of trajectories (modified version of Geolife GPS Trajectory dataset), some code to connect to a database using python, and some example code for how to do different queries in MongoDB and python. The task was to clean the dataset, create suitable tables in a MySQL database, insert the data, and then execute different queries through python to get data from the database.

The group consisted of only two members, so communication was easily done in either conference calls or face to face when needed. Git was used throughout the development, but only for version control, as opening issues etc. was not needed in such a small group where both members knew exactly what to work on at all times. Here is a link to the repo: https://gitlab.stud.idi.ntnu.no/trymg/exercise-3-tdt4225

This exercise was done using a local MongoDB database running on Windows.

Special to this assignment compared to the previous, was that the database management system required was MongoDB rather than MySQL, while the dataset stayed the same. This meant that we could use some of the existing code for loading the dataset.

# Results

## Task 1

Top 10 documents from all collections:
Users:

```
> db.users.find().sort({_id:1}).limit(10)
{ "_id" : 1, "has_label" : false }
{ "_id" : 2, "has_label" : false }
{ "_id" : 3, "has_label" : false }
{ "_id" : 4, "has_label" : false }
{ "_id" : 5, "has_label" : false }
{ "_id" : 6, "has_label" : false }
{ "_id" : 7, "has_label" : false }
{ "_id" : 8, "has_label" : false }
{ "_id" : 9, "has_label" : false }
{ "_id" : 10, "has_label" : false }
```

Activities:

```
> db.activities.find().sort({_id:1}).limit(10)
{ "_id" : 1, "user_id" : 1, "transportation_mode" : "NULL", "start_date_time" : ISODate("2008-10
{ "_id" : 2, "user_id" : 1, "transportation_mode" : "NULL", "start_date_time" : ISODate("2008-10
{ "_id" : 3, "user_id" : 1, "transportation_mode" : "NULL", "start_date_time" : ISODate("2008-10
{ "_id" : 4, "user_id" : 1, "transportation_mode" : "NULL", "start_date_time" : ISODate("2008-10
{ "_id" : 5, "user_id" : 1, "transportation_mode" : "NULL", "start_date_time" : ISODate("2008-10
{ "_id" : 6, "user_id" : 1, "transportation_mode" : "NULL", "start_date_time" : ISODate("2008-10
{ "_id" : 7, "user_id" : 1, "transportation_mode" : "NULL", "start_date_time" : ISODate("2008-10
{ "_id" : 8, "user_id" : 1, "transportation_mode" : "NULL", "start_date_time" : ISODate("2008-11
{ "_id" : 9, "user_id" : 1, "transportation_mode" : "NULL", "start_date_time" : ISODate("2008-11
{ "_id" : 10, "user_id" : 1, "transportation_mode" : "NULL", "start_date_time" : ISODate("2008-1
```

Trackpoints:

```
> db.trackpoints.find().sort({_id:1}).limit(10)
{ "_id" : 1, "activity_id" : 1, "latlong" : [ 39.984702, 116.318417 ], "altitude" : 492, "date_time" : ISODate("2
{ "_id" : 2, "activity_id" : 1, "latlong" : [ 39.984683, 116.31845 ], "altitude" : 492, "date_time" : ISODate("20
{ "_id" : 3, "activity_id" : 1, "latlong" : [ 39.984686, 116.318417 ], "altitude" : 492, "date_time" : ISODate("2
{ "_id" : 4, "activity_id" : 1, "latlong" : [ 39.984688, 116.318385 ], "altitude" : 492, "date_time" : ISODate("2
{ "_id" : 5, "activity_id" : 1, "latlong" : [ 39.984655, 116.318263 ], "altitude" : 492, "date_time" : ISODate("2
{ "_id" : 6, "activity_id" : 1, "latlong" : [ 39.984611, 116.318026 ], "altitude" : 493, "date_time" : ISODate("2
{ "_id" : 7, "activity_id" : 1, "latlong" : [ 39.984608, 116.317761 ], "altitude" : 493, "date_time" : ISODate("2
{ "_id" : 8, "activity_id" : 1, "latlong" : [ 39.984563, 116.317517 ], "altitude" : 496, "date_time" : ISODate("2
{ "_id" : 9, "activity_id" : 1, "latlong" : [ 39.984539, 116.317294 ], "altitude" : 500, "date_time" : ISODate("2
{ "_id" : 10, "activity_id" : 1, "latlong" : [ 39.984606, 116.317065 ], "altitude" : 505, "date_time" : ISODate('
```

# Task 2

Queries terminal output:

```
PS C:\Users\trymg\Documents\master\datamengder\exercise-3-tdt4225> python .\src\main.py
You are connected to the database: admin
-----------------------------------------------

Task 2.1:
182
16048
9681756

task 2.2:
{'Average': 92.76300578034682, 'Minimum': 1, 'Maximum': 2102}

task 2.3:
{'COUNT(*)': 2102, 'users': {'_id': 129}}
{'COUNT(*)': 1793, 'users': {'_id': 154}}
{'COUNT(*)': 715, 'users': {'_id': 26}}
{'COUNT(*)': 704, 'users': {'_id': 164}}
{'COUNT(*)': 691, 'users': {'_id': 63}}
{'COUNT(*)': 563, 'users': {'_id': 145}}
{'COUNT(*)': 399, 'users': {'_id': 42}}
{'COUNT(*)': 364, 'users': {'_id': 86}}
{'COUNT(*)': 346, 'users': {'_id': 5}}
{'COUNT(*)': 345, 'users': {'_id': 141}}

task 2.4:
98

task 2.5:

task 2.6: (too long runtime, limit is needed)
Limit 50000:
100%|████████████████████████████████████████████████████████████████████████████████████████████████████████████████████████████████████████████████
users in contact: 0

task 2.6: (too long runtime, limit is needed)
No limit, canceled after 5 hours:
  0%|                        | 1073/9318110 [5:06:03<44293:16:27, 17.11s/it]

task 2.7:
{'_id': 1, 'has_label': False}
{'_id': 2, 'has_label': False}
{'_id': 3, 'has_label': False}
{'_id': 4, 'has_label': False}
{'_id': 5, 'has_label': False}
{'_id': 6, 'has_label': False}
{'_id': 7, 'has_label': False}
{'_id': 8, 'has_label': False}
{'_id': 9, 'has_label': False}
{'_id': 10, 'has_label': False}
{'_id': 12, 'has_label': False}
{'_id': 13, 'has_label': False}
```

```
{'_id': 14, 'has_label': False}
{'_id': 15, 'has_label': False}
{'_id': 16, 'has_label': False}
{'_id': 17, 'has_label': False}
{'_id': 18, 'has_label': False}
{'_id': 19, 'has_label': False}
{'_id': 20, 'has_label': False}
{'_id': 21, 'has_label': True}
{'_id': 22, 'has_label': True}
{'_id': 23, 'has_label': False}
{'_id': 24, 'has_label': False}
{'_id': 25, 'has_label': False}
{'_id': 26, 'has_label': False}
{'_id': 27, 'has_label': False}
{'_id': 28, 'has_label': False}
{'_id': 29, 'has_label': False}
{'_id': 30, 'has_label': False}
{'_id': 31, 'has_label': False}
{'_id': 32, 'has_label': False}
{'_id': 33, 'has_label': False}
{'_id': 34, 'has_label': False}
{'_id': 35, 'has_label': False}
{'_id': 36, 'has_label': False}
{'_id': 37, 'has_label': False}
{'_id': 38, 'has_label': False}
{'_id': 39, 'has_label': False}
{'_id': 40, 'has_label': False}
{'_id': 41, 'has_label': False}
{'_id': 42, 'has_label': False}
{'_id': 43, 'has_label': False}
{'_id': 44, 'has_label': False}
{'_id': 45, 'has_label': False}
{'_id': 46, 'has_label': False}
{'_id': 47, 'has_label': False}
{'_id': 48, 'has_label': False}
{'_id': 49, 'has_label': False}
{'_id': 50, 'has_label': False}
{'_id': 51, 'has_label': False}
{'_id': 52, 'has_label': False}
{'_id': 53, 'has_label': True}
{'_id': 54, 'has_label': True}
{'_id': 55, 'has_label': False}
{'_id': 56, 'has_label': False}
{'_id': 57, 'has_label': True}
{'_id': 58, 'has_label': False}
{'_id': 60, 'has_label': True}
{'_id': 61, 'has_label': True}
{'_id': 62, 'has_label': False}
{'_id': 64, 'has_label': False}
{'_id': 65, 'has_label': True}
{'_id': 66, 'has_label': True}
```

```
{'_id': 67, 'has_label': False}
{'_id': 68, 'has_label': True}
{'_id': 69, 'has_label': True}
{'_id': 70, 'has_label': True}
{'_id': 71, 'has_label': False}
{'_id': 72, 'has_label': False}
{'_id': 73, 'has_label': False}
{'_id': 74, 'has_label': True}
{'_id': 75, 'has_label': False}
{'_id': 76, 'has_label': True}
{'_id': 77, 'has_label': True}
{'_id': 78, 'has_label': False}
{'_id': 80, 'has_label': False}
{'_id': 82, 'has_label': True}
{'_id': 83, 'has_label': True}
{'_id': 84, 'has_label': False}
{'_id': 85, 'has_label': True}
{'_id': 87, 'has_label': True}
{'_id': 88, 'has_label': True}
{'_id': 89, 'has_label': True}
{'_id': 90, 'has_label': True}
{'_id': 91, 'has_label': False}
{'_id': 92, 'has_label': True}
{'_id': 93, 'has_label': True}
{'_id': 94, 'has_label': False}
{'_id': 95, 'has_label': False}
{'_id': 96, 'has_label': False}
{'_id': 97, 'has_label': True}
{'_id': 98, 'has_label': True}
{'_id': 100, 'has_label': False}
{'_id': 101, 'has_label': True}
{'_id': 102, 'has_label': True}
{'_id': 103, 'has_label': True}
{'_id': 104, 'has_label': False}
{'_id': 105, 'has_label': True}
{'_id': 106, 'has_label': True}
{'_id': 107, 'has_label': True}
{'_id': 108, 'has_label': True}
{'_id': 109, 'has_label': True}
{'_id': 110, 'has_label': False}
{'_id': 111, 'has_label': True}
{'_id': 113, 'has_label': True}
{'_id': 114, 'has_label': False}
{'_id': 115, 'has_label': True}
{'_id': 116, 'has_label': True}
{'_id': 117, 'has_label': True}
{'_id': 118, 'has_label': True}
{'_id': 119, 'has_label': True}
{'_id': 120, 'has_label': False}
{'_id': 121, 'has_label': False}
{'_id': 122, 'has_label': False}
```

```
{'_id': 123, 'has_label': False}
{'_id': 124, 'has_label': False}
{'_id': 125, 'has_label': True}
{'_id': 126, 'has_label': True}
{'_id': 127, 'has_label': True}
{'_id': 128, 'has_label': False}
{'_id': 130, 'has_label': True}
{'_id': 131, 'has_label': False}
{'_id': 132, 'has_label': False}
{'_id': 133, 'has_label': False}
{'_id': 134, 'has_label': False}
{'_id': 135, 'has_label': False}
{'_id': 136, 'has_label': False}
{'_id': 137, 'has_label': True}
{'_id': 138, 'has_label': False}
{'_id': 139, 'has_label': True}
{'_id': 140, 'has_label': True}
{'_id': 141, 'has_label': False}
{'_id': 142, 'has_label': True}
{'_id': 143, 'has_label': False}
{'_id': 144, 'has_label': False}
{'_id': 145, 'has_label': True}
{'_id': 146, 'has_label': False}
{'_id': 147, 'has_label': False}
{'_id': 148, 'has_label': True}
{'_id': 149, 'has_label': False}
{'_id': 150, 'has_label': False}
{'_id': 151, 'has_label': False}
{'_id': 152, 'has_label': False}
{'_id': 153, 'has_label': False}
{'_id': 154, 'has_label': True}
{'_id': 155, 'has_label': True}
{'_id': 156, 'has_label': False}
{'_id': 157, 'has_label': False}
{'_id': 158, 'has_label': False}
{'_id': 159, 'has_label': False}
{'_id': 160, 'has_label': False}
{'_id': 161, 'has_label': False}
{'_id': 162, 'has_label': True}
{'_id': 163, 'has_label': False}
{'_id': 165, 'has_label': False}
{'_id': 166, 'has_label': False}
{'_id': 167, 'has_label': False}
{'_id': 168, 'has_label': True}
{'_id': 169, 'has_label': False}
{'_id': 170, 'has_label': False}
{'_id': 171, 'has_label': True}
{'_id': 172, 'has_label': False}
{'_id': 173, 'has_label': False}
{'_id': 174, 'has_label': False}
{'_id': 175, 'has_label': True}
```

```
{'_id': 176, 'has_label': True}
{'_id': 177, 'has_label': False}
{'_id': 178, 'has_label': False}
{'_id': 179, 'has_label': False}
{'_id': 180, 'has_label': True}
{'_id': 181, 'has_label': False}
{'_id': 182, 'has_label': False}

task 2.8:
airplane 1
bike 19
boat 1
bus 13
car 8
run 1
subway 4
taxi 10
train 2
walk 34

task 2.9a:
{'year': 2008, 'month': 11, 'COUNT(*)': 1006}

task 2.9b - Which user had the most activities 2008-11?
{'activities': {'user_id': 63}, 'COUNT(user_id)': 130}

task 2.9b2 - How many recorded hours does user 63 have in 2008-11?
2835

task 2.9b3 - Which user had the second most activities 2008-11?
{'activities': {'user_id': 129}, 'COUNT(user_id)': 75}

task 2.9b4 - Does the user with the most activities 2008-11 have more recorded hours than the us
4092
2835 < 4092 - The user with the most activities does not have more recorded hours than the user

------------------------------------------------
Connection to admin-db is closed
```

# Discussion

Modelling:
Users to activities: Max relationship number is 2000, meaning one-to-many relationship. Array of ObjectID(activity) inside each user is appropriate here (child referencing). For activities to trackpoints, the relationship is "one-to-squillions". Here, one would use parent referencing by keeping an ObjectID(activity) inside each trackpoint. This is similar to relational database systems.

However, for this exercise, we decided to use parent referencing for both cases because of the following reasons:

- Simpler to implement as first time MongoDB users.
- Simpler to query, and less to keep track of.
- Can use old queries as it mimicks relational database systems.

Latitude and longitude were combined into a "latlong" array containing both values as the array length is always only 2.

For task 2, we were a little confused as to wether or not we had modelled the database correctly for our queries to work properly. This was due to not having any prior experience using MongoDB or any other document oriented database management systems. We eventually figured out how to query correctly, and moved along.

Task 2.6
This task was done locally in python as we did not find a viable solution using MongoDB queries. In this solution, the runtime was too long to be computed, and therefore a limit of 50 000 documents was set. The implementation was done simply using substring on the dates to remove the "seconds" part, and mathing values, effectively being the same as rounding down to the nearest minute. This means that each match would be within the same minute, but not all would be included. For the distance part, haversine was used normally in python. The result was 0 people in contact after 50 000 documents cross searched. We also searched for a consecutive 5 hours without limit, but only 10K out of the 10M trackpoints got checked. This also resulted in 0 results.

Task 2.10
The query returns no rows, indicating that something is either wrong with the query itself, or something went wrong during the insertion of the data. If some rows had been returned, we would have used Haversine in python to check the distance between each lon, lat-pair, and then added the distances together for each activity to find the total distance walked in km in 2008 by user with id = 112.

Task 2.11 and 2.12 are unfortunately missing due to lack of time.

We learned a lot about the practicalities of using MongoDB as opposed to MySQL in this exercise. Both in terms of syntax, modelling, and querying.

# MySQL vs MongoDB

MongoDB/NoSQL databases:

- Pros

    - Advantageous if the items in the collection don't all have the same structure for some reason.
    - Suits document-like structures.
    - Can scale vertically and horizontally.
    - No need for relations, only key-value-stores.
- Cons
    - Bad support for joins, forcing awkward manual joins in application code.
    - Bad support for many-to-many relationships.

SQL/Relational databases

- Pros
    - Good support for joins.
    - Good support for many-to-many relationships.
    - Gives room for optimizations and parallel execution e.g. using indexes.
- Cons
    - Can require ORM (Object Relational Mapping) for mismatch between DB rows and application objects.
    - Have to maintain relations.
    - Can't scale horizontally because relations will be split up.

For this exercise, we preferred using MySQL as the data objects have relations between each other. This especially made querying a lot easier using MySQL.

# Feedback

This was a good, educational exercise. The only improvement we would suggest is to make less ambiguous task descriptions, so that it is very clear what the task asks for (e.g. task 2.7).