Assignment 3

TDT4225 - Very Large, Distributed Data Volumes

Deadline: Oct 23 at 23:59

(end of weekend due to request from some students)

Introduction

In this assignment you'll be working in groups of (max) 3 students to solve some database tasks using the MongoDB database **(version 6.0.1)** and coding in Python.

Every group has access to a virtual machine at IDI's cluster, running Ubuntu. This machine will have MongoDB already installed and set up for you to use. You may also run MongoDB yourself if your are on group 41 - 100

This assignment will look at an open dataset of trajectories, and your task is to create collections, clean and insert data, and fetch the data to answer some questions. Your Python program will handle this functionality. The program is inspired by the social media workout application Strava, where users can track activities like running, walking, biking etc and post them online with stats about their workout.

Along with this assignment sheet, you have been given several files:

- Dataset - Modified dataset based on Geolife GPS Trajectory dataset (please use this dataset, instead of the one from the website. The dataset from the website contains some files that may give you an error).
- `DbConnector.py` - a Python class that connects to MySQL on the virtual machine.
- `example.py` - a Python class with examples on how to create tables, insert, fetch and delete data in MySQL.
- `requirements.txt` - a file containing some pip packages that should be used in this assignment.
- `labeled_ids.txt` - a file containing the ID of all users who have labeled their data. This is found in the `dataset.zip` file.
- `User-Guide-1.3.pdf` - an official user guide to the Geolife dataset. This is found in the `dataset.zip` file.

This assignment is very similar to assignment 2 where you used MySQL. The dataset is the same, and the tasks are the same. The switch from an SQL-database to a NoSQL-database is the real challenge here.

It is strongly recommended that you read through and understand the whole assignment sheet before you start solving the tasks. There are also some tips at the bottom of this assignment, which could be very handy!

Dataset - Geolife GPS Trajectory dataset

Geolife GPS Trajectory dataset is an open source dataset collected by Microsoft from 2007-2011. This dataset recorded a broad range of users' outdoor movements, including not only life routines like going home and going to work but also some entertainments and sports activities, such as shopping, sightseeing, dining, hiking, and cycling. The user data is mainly from Beijing, China, but also from the US and Europe.

The dataset is provided in the `dataset.zip` and has some differences from the one online, for the purpose of this assignment. It contains 182 users that have tracked 18 669 activities in total. Each activity contains a number of GPS-points, which we call TrackPoints. In total there are over 24 million trackpoints in this dataset.

In the `dataset.zip` you will find the official user guide for the dataset. Be sure to read through this before you start the task. Here is also a short summary of the dataset, understanding this is crucial for completing the assignment:

The folder named Data contains all the 182 users, labeled from "000" to "181". Each user has a Trajectory-folder where each activity is stored as a .plt file. Each .plt file contains several trackpoints.

Line 1…6 in the .plt files are useless in this dataset, and can be ignored. Trackpoints are described in following lines, one for each line:

Field 1: Latitude in decimal degrees.
Field 2: Longitude in decimal degrees.
Field 3: All set to 0 for this dataset, i.e. you don't need this field.
Field 4: Altitude in feet (-777 if not valid).
Field 5: Date - number of days (with fractional part) that have passed since 12/30/1899.
Field 6: Date as a string.
Field 7: Time as a string.
Note that field 5 and field 6&7 represent the same date/time in this dataset. You may use either of them.

Example:
39.906631, 116.385564, 0, 492, 40097.5864583333, 2009-10-11, 14:04:30
39.906554, 116.385625, 0, 492, 40097.5865162037, 2009-10-11, 14:04:35

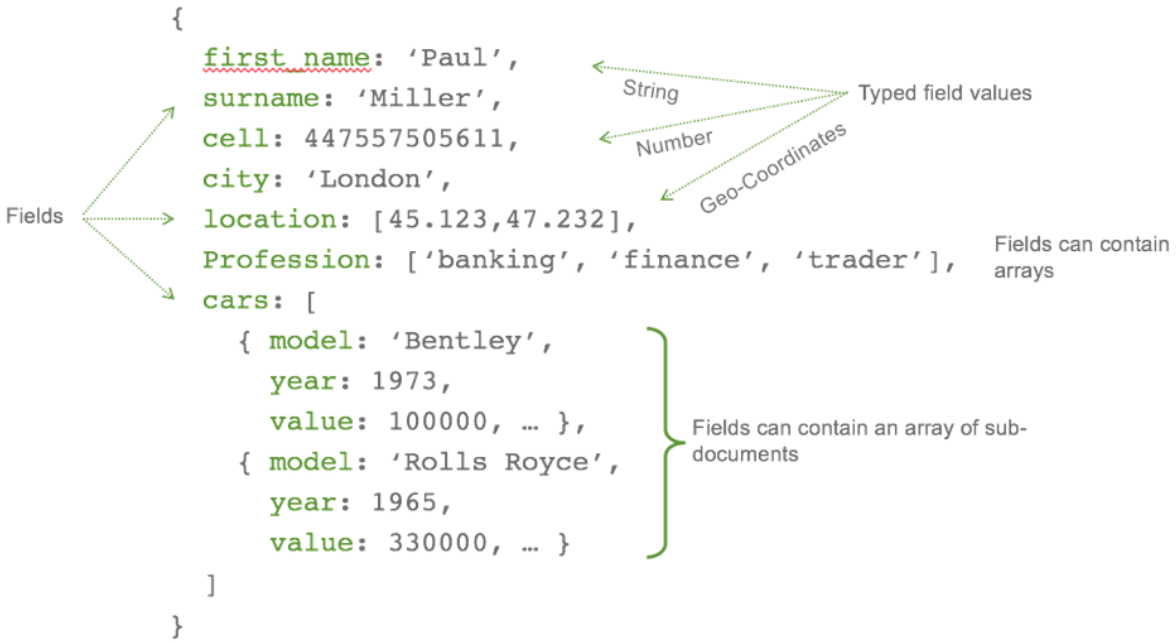
Some users have also labeled their data with transportation modes. Possible transportation modes are: *walk*, *bike*, *bus*, *taxi*, *car*, *subway*, *train*, *airplane*, *boat*, *run and motorcycle*. We have provided you with the ids of the users that have labeled their data in `labeled_ids.txt`, which will be used for the tasks. The labels contain start time and end time (both date and time) along with the transportation mode for each activity.

Example:

Start Time	**End Time**	**Transportation Mode**
2008/04/02 11:24:21	2008/04/02 11:50:45	bus
2008/04/03 01:07:03	2008/04/03 11:31:55	train

MongoDB - NoSQL

A quick note on MongoDB and NoSQL. There is some terminology that is different from the regular relational-database. MongoDB stores data in documents. Documents are JSON documents based on the JSON specification.
An example of a JSON document would be as follows:



Notice that documents are not just key-value pairs but can include arrays and subdocuments. The data itself can be different data types like geospatial, decimal, string etc. This does not have to be pre-defined, neither does two documents in the same collection have to include the same fields.

A **database** in MongoDB is the container for **collections**. A **collection** is a container for **documents**. This grouping is similar to relational databases and is pictured below:

Relational concept	MongoDB equivalent
Database	Database
Tables	Collections
Rows	Documents
Index	Index

Source: Robert Walters, *'Getting Started with Python and MongoDB'*, 2017.
Url: https://www.mongodb.com/blog/post/getting-started-with-python-and-mongodb

Collections

As mentioned earlier, collections in MongoDB are not as strict as in relational databases. How you decide to organize your collections and documents is up to you. The data we are interested in is the same as from Assignment 2 and is displayed in the figure below. Notice that foreign keys are not really a thing in MongoDB, but it is optional how you want to reference things in each document.

Because MongoDB offers a lot of ways to handle this, we encourage you to investigate what solution that might suit the purpose of this assignment best. Here's three great articles that we recommend reading before making a decision:
MongoDB Schema Design - Part 1 (5 min reading time)
MongoDB Schema Design - Part 2 (5 min reading time)
MongoDB Schema Design - Part 3 (5 min reading time)

Remember what you did in Assignment 2 and what approach that would fit the tasks.

User	**Activity**	**TrackPoint**
_id - string has_labels - boolean	_id - int transportation_mode - string start_date_time - datetime end_date_time - datetime	_id - int lat - double lon - double altitude - int date_days - double date_time - datetime

Optional reference to Activity	Optional reference to User and/or TrackPoint	Optional reference to Activity

Notice how the id is written as "_id". In MongoDB, there will always be a "_id" as an ObjectID if the field is not specified (read more here). E.g., creating a document like this:

```
Person = {
        id: 1,
        name: 'Name Namesen'
}
```

would result in a Person-document like this when inserted to the database:

```
Person = {
        _id: ObjectID('Some12byteNumber')
        id: 1,
        name: 'Name Namesen'
}
```

You have to manually override the "_id"-field if you want to set a defined ID (e.g. '014' for Users).

Datetime should be consistent throughout your collections, e.g. in the format YYYY-MM-DD HH:MM:SS.

Setup database

These steps will guide you through how to connect to the virtual machine from your computer and create a MongoDB-database user with privileges.

1. First, in case you are not on the NTNU network, you need to use NTNU's VPN to access the virtual machines. Log onto NTNU-VPN with your Feide user (check out this link if you have not done this before).

2. Open your terminal and enter the following command:
   `ssh your_username@tdt4225-xx.idi.ntnu.no` where **"your_username"** is the Feide-username and **"xx"** is your group number for this project (01, 02, 03... etc).
   (If this message pops up, enter **yes**:
   ```
   The authenticity of host 'tdt4225-xx.idi.ntnu.no
   (xxx.xxx.xxx.xxx)' can't be established.
   ECDSA key fingerprint is …
   Are you sure you want to continue connecting (yes/no)?
   yes)
   ```

You will then be asked to enter your password, use the Feide-password here. If the password is correct, you have now successfully logged in!

3. When using MongoDB, we have to have a server running in one terminal window. First, create the database catalogs.
   % sudo mkdir /data
   % sudo mkdir /data/db
   Then, type `sudo mongod --repair` (only have to be done the first time you start your server) in your terminal window and type your Feide password. Once that is completed type `sudo mongod --bind_ip_all`. If successful, you have now a running MongoDB-server (the bind_ip_all flag opens for connections from any IP-address, but don't worry, we'll use authentication). This server must be running whenever you are working with the task. Only one of the group members has to run the server at a time.

4. Open a new terminal window and ssh into the virtual machine. Do not close the server-window you have running. To open a client-window for MongoDB type `sudo mongosh`. Now, you have your MongoDB-server running in one terminal window, and a MongoDB-client running in the other. If you check your server, the client-login should've been logged there.

5. Now we want to create an admin MongoDB-user. The MongoDB-client accepts shell commands written in JavaScript.
   Start by creating an admin-user by typing the following commands:
   `> use admin`, this chooses the admin database
   ```
   > db.createUser({
       user: 'your_username',
       pwd: 'your_password',
       roles: ['root']
   })
   ```
   , this creates a user with the root-role. Change your_username and your_password with desired names. The command can be written in one line, it is spread across several lines here for readability. Here is an example where your_username=adminuser and pwd=admin123:

   ```
   > db.createUser({
       user: 'adminuser',
       pwd: 'admin123',
       roles: ['root']
   })
   ```
   Now type `> show users;` and find your newly created admin user.

6. Now let's create the user and the database that the group will use during this assignment. To create a database you simply type `> use my_db;` where my_db is your chosen name. If the database does not exist, it will create it for you. Now do the same command as in Step 5, but with a read/write access-role:
```
> db.createUser({
      user: 'your_username',
      pwd: 'your_password',
      roles: ['readWrite']
})
```
Now, your user will have read and write access to the my_db database that you created. Type `> show users;` and find your newly created user, and check that the role is correct and attached to the correct database (my_db not admin). You can also type , but the database will not show until you have created collections and inserted data into it.

7. You can use this client window to create collections etc with the shell commands, but from now on the Python program will be our MongoDB-client. Remember to keep the MongoDB-server running.

Setup Python

We will be using a MongoDB connector to Python (PyMongo) in this assignment. If you for some reason would like to complete the assignment in another language, you are allowed to do so. ***(NB! We will not provide support or documentation for other languages, so we strongly encourage you to use Python.)***

With the files provided in this assignment, you will find a `requirements.txt`, `DbConnector.py` and `example.py`, among others.

1. To set up the required pip-packages for this assignment, simply run `pip install -r requirements.txt` while you are in the directory with the requirements-file. This will install the connector, along with tabulate (used to print pretty tables in python) and haversine (used to calculate distance between two coordinates).

2. Now, open `DbConnector.py` and look at the code. There will be a TODO there to set up the database correctly with the settings from the database setup. Update the values accordingly:
Host = tdt4225-xx.idi.ntnu.no, where xx is your group number

Database = the name you provided for your database in step 6 of the database setup (my_db from the example)
User = the username provided in step 6, (not the admin user)
Password = the password provided in step 6

3. Open `example.py`, the file has a main method that creates a connection to the database from DbConnector, creates a table named "Person", inserts some names in the database, displays the data, and then drops the table.
   Try to run the code. If everything is set up correctly, you will establish a connection to the database-server and insert/delete data. You can use this file for inspiration when solving the tasks in this assignment.

Tasks

The tasks are divided into three parts: Part 1 will focus on cleaning and inserting the data into defined tables. Part 2 will focus on writing queries to the database to gain knowledge of the dataset. Part 3 will focus on writing a report where you discuss your answers.
We recommend looking at the documentation for the MongoDB-Python connector before you start coding.

Part 1

In this task you'll clean and insert the Geolife dataset into your own MongoDB database, to be able to solve the questions in Part 2. In the section "Geolife GPS Trajectory dataset" above, you'll find all the information you need about the dataset you are handed, and in the section "Collection" you'll find some nice articles for how to structure the dataset in your MongoDB database. Please discuss in your report how you structure your collections and documents.

Write a Python program that does the following:

1. Connects to the MongoDB server on your Ubuntu virtual machine.
2. Creates and defines the collections User, Activity and TrackPoint
3. Inserts the data from the Geolife dataset into your MongoDB database
   - Here, we require a bit of data cleaning and integration. Study the dataset closely to understand which data goes where.
   - Iterating through the dataset in the directories can be done using os.walk method, but you are free to use other methods if you find them better for your solution.
   - When matching transportation_mode from the `labels.txt` files to the activities, we only consider exact matches on starttime and end time, i.e. if you find a match in start time, but not in end time, or vice

versa, it should not be included. Additionally, there are some labeled activities in `labels.txt` that will not have a match amongst the activities.
- **Important!** Sometimes it is wise to limit the dataset we're working with. 24 million TrackPoints is a lot. Therefore we only want you to insert activities that have **fewer than or exactly 2500 trackpoints** in them, i.e. when inserting activities into the database, check that the size of the .plt-files do not exceed 2500 lines (excluding the headers, of course). When you insert TrackPoints, the same rule applies.
- Inserting the trackpoints may take a while! Instead of inserting one document of trackpoints at a time, find out if there is a way to insert bulks of data instead.

Part 2

Some of the tasks can be answered using MongoDB-queries only, while some might require both queries and Python code to manipulate the data correctly. Use pprint to pretty-print the data output from MongoDB.

Answer the following questions by writing a Python program using MongoDB-queries (like in `example.py`):

1. How many users, activities and trackpoints are there in the dataset (after it is inserted into the database).
2. Find the average number of activities per user.
3. Find the top 20 users with the highest number of activities.
4. Find all users who have taken a taxi.
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.
6. a) Find the year with the most activities.
   b) Is this also the year with most recorded hours?
7. Find the total distance (in km) walked in 2008, by user with id=112.
8. Find the top 20 users who have gained the most altitude meters.
   1. Output should be a field with (id, total meters gained per user).
   2. Remember that some altitude-values are invalid
   3. Tip: $\sum (tp_n.altitude - tp_{n-1}.altitude),\ tp_n.altitude > tp_{n-1}.altitude$
9. Find all users who have invalid activities, and the number of invalid activities per user
   1. An invalid activity is defined as an activity with consecutive trackpoints where the timestamps deviate with at least 5 minutes.

10. Find the users who have tracked an activity in the Forbidden City of Beijing.
    1. In this question you can consider the Forbidden City to have coordinates that correspond to: *lat 39.916, lon 116.397*.
11. Find all users who have registered transportation_mode and their most used transportation_mode.
    1. The answer should be on format (user_id, most_used_transportation_mode) sorted on user_id.
    2. Some users may have the same number of activities tagged with e.g. walk and car. In this case it is up to you to decide which transportation mode to include in your answer (choose one).
    3. Do not count the rows where the mode is null.

Part 3

Write a short report (see `report-template.docx` in the `tdt4225-assignment2.zip`) where you will display and discuss your results from the tasks. Include screenshots from both part 1 (showing top 10 rows from all of your tables is sufficient) and part 2 (all the results to each task).

Hand in both the report (as PDF) and your code to BlackBoard within Oct 23, at 23:59.

Tips

- Using the MongoDB terminal on your Ubuntu machine could be useful for checking simple queries without having to use Python and the connector.
  - Be sure to have a server running, `sudo mongod --bind_ip_all`
  - Open the MongoDB client terminal by typing `sudo mongosh`, log in with Feide-password and type `use name_of_db` to do this.
- Using dictionaries/hashmaps are faster for lookups than lists/arrays in your Python code
- When extracting transportation_mode from the `labels.txt` files, remember that you only have to consider the user-ids found in the `labeled_ids.txt`, as they are the only users where transportation_mode may not be null.
- Start_time and end_time for activities can be found by looking at the date and time for the first and last trackpoint in each .plt-file
- Use some time to consider the different choices for how to store your data.
  - Embedded documents, one-way referencing or to -way referencing.
- Remember to insert data with the right type. Insert integer as int, date as some dateformat, etc..
  - MongoDB support e.g ISODate
- As stated, using Haversine for calculating distance is recommended

- Comparison between SQL and MongoDB.
- Comparison between SQL-functions and MongoBD-functions.
- Aggregation and the pipeline stages for this are worth checking out.
- Optional - if you want to visualize the data in the dataset, you can get inspired here.