Assignment 2

TDT4225 - Very Large, Distributed Data Volumes

Deadline: Oct 8 at 16:00

Introduction

In this assignment you'll be working in groups of 3 students to solve some database tasks using the MySQL database **(version 8.0.26)** and coding in Python.

This exercise will be graded and counts 25 % of your final grade. The grading will be finished within 3 weeks after the delivery deadline.

Most groups have access to a virtual machine at IDI's cluster, running Ubuntu. This machine will have MySQL already installed and set up for you to use.

This assignment will look at an open dataset of trajectories, and your task is to create tables, clean and insert data, and to make queries 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.

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 `tdt4225-assignment2.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 `tdt4225-assignment2.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.

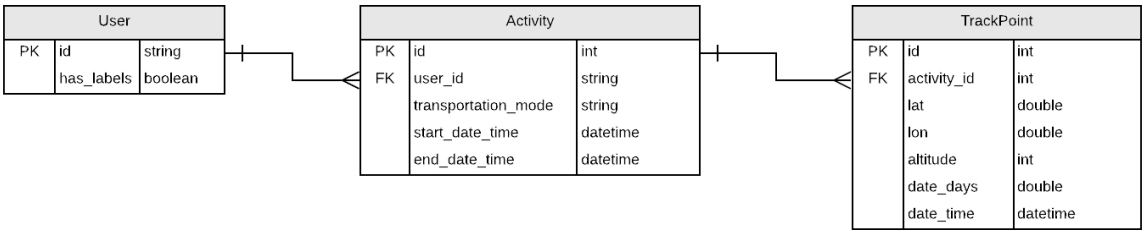
Example:
39.906631, 116.385564, 0, 492, 40097.5864583333, 2009-10-11, 14:04:30
39.906554, 116.385625, 0, 492, 40097.5865162037, 2009-10-11, 14:04:35

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

Tables



User	**Activity**	**TrackPoint**
id - string	id - int	id - int
has_labels - boolean	user_id - string (Foreign Key)	activity_id - int (Foreign Key)
	transportation_mode - string	lat - double
	start_date_time - datetime	lon - double
	end_date_time - datetime	altitude - int
		date_days - double
		date_time - datetime

Datetime should be consistent throughout your tables, 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 MySQL-database user with privileges.

1. First, 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. Now we want to create a MySQL-user that the group will use during this assignment. First, enter `sudo mysql` in your terminal windows (which is now inside the virtual machine) and type in your Feide password. If successful, you are now on the MySQL server and can write MySQL-queries directly in this window: `mysql> "some query"`.

4. Run the following commands in MySQL to create a new user and give the user admin rights.

   ```
   mysql> CREATE USER 'YOUR_USERNAME_HERE'@'%' IDENTIFIED BY 'YOUR_PASSWORD_IN_PLAIN_TEXT_HERE';

   mysql> GRANT ALL PRIVILEGES ON *.* TO 'YOUR_USERNAME_HERE'@'%'WITH GRANT OPTION;
   ```

   Yes, write the username using quotas ''. Here is an example for username=testuser and password=test123:

   ```
   CREATE USER 'testuser'@'%' IDENTIFIED BY 'test123';
   GRANT ALL PRIVILEGES ON *.* TO 'testuser'@'%' WITH GRANT OPTION;
   ```

Now your new user has been granted admin rights. The % sign signifies a wildcard, meaning that your new user will have access to all schemas in MySQL. You will want to flush the privileges so that MySQL will load your user with its new rights. Type `mysql> FLUSH PRIVILEGES;`

To check if the user is created, type `mysql> SELECT User FROM mysql.user;`. If you see your newly created user in the list, you have created the user successfully.

5. Let's create a database. Type `mysql> SHOW DATABASES;`, there should be four default databases, ignore these.
   Create a new database by typing `mysql> CREATE DATABASE db_name;`, where **"db_name"** is your chosen name of the database, example `mysql> CREATE DATABASE test_db;`. Try typing `mysql> SHOW DATABASES;` again and check if the database is there. If it is there, congratulations!

To exit the MySQL program, simply type `exit` and you are back at the virtual machine. Now you have to type `sudo service mysql restart` in the terminal. Once this is complete, you should be able to use the database from your Python program (or the MySQL terminal).

Setup Python

We will be using a MySQL connector for Python 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 5 of the database setup (test_db from the example)
User = the username provided in step 4, (testuser from the example)
Password = the password provided in step 4 (test123 from the example)
Port = this is optional, and default is 3306

- Passwords should be stored as an environmental variable if your code is public. See here, or search Google to find out how to do it.

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 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 MySQL-Python connector before you start coding.

Part 1

In this task you'll clean and insert the Geolife dataset into your own MySQL 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 "Tables" you'll find our suggestion for how to structure the dataset in your MySQL database. You are free to do it in another way as well, but please discuss in your report why you do things differently.

Write a Python program that does the following:

1. Connects to the MySQL server on your Ubuntu virtual machine.
2. Creates and defines the tables User, Activity and TrackPoint
3. Inserts the data from the Geolife dataset into your MySQL database
   - Here, we require a bit of data cleaning and integration. Study the dataset and the tables closely to understand which data goes where.
   - Insert data into User, Activity and TrackPoint, in that order.
     - E.g. you cannot create an Activity (with the filed user_id) without having a User with that id.

- 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 (you cannot link a trackpoint to an activity that is dropped, anyway).
- Inserting the trackpoints may take a while (could potentially be 10-15 mins)! Instead of inserting one row of trackpoints at a time, find out if there is a way to insert batches of data instead.

Part 2

Some of the tasks can be answered using MySQL-queries only, while some might require both queries and Python code to manipulate the data correctly.

Answer the following questions by writing a Python program using MySQL-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, minimum and maximum number of activities per user.
3. Find the top 10 users with the highest number of activities.
4. Find the number of users that have started the activity in one day and ended the activity the next day.
5. Find activities that are registered multiple times. You should find the query even if you get zero results.
6. Find the number of users which have been close to each other in time and space (Covid-19 tracking). Close is defined as the same minute (60 seconds) and space (100 meters).
7. Find all users that have never taken a taxi.
8. Find all types of transportation modes and count how many distinct users that have used the different transportation modes. Do not count the rows where the transportation mode is null.

9. a) Find the year and month with the most activities.
   b) Which user had the most activities this year and month, and how many recorded hours do they have? Do they have more hours recorded than the user with the second most activities?
10. Find the total distance (in km) walked in 2008, by user with id=112.
11. Find the top 20 users who have gained the most altitude meters.
    - Output should be a table with (id, total meters gained per user).
    - Remember that some altitude-values are invalid
    - Tip: $\sum(tp_n.altitude - tp_{n-1}.altitude), \quad tp_n.altitude > tp_{n-1}.altitude$
12. Find all users who have invalid activities, and the number of invalid activities per user
    - An invalid activity is defined as an activity with consecutive trackpoints where the timestamps deviate with at least 5 minutes.

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 9, at 16:00.

Tips

- Using the MySQL terminal on your Ubuntu machine could be useful for checking simple queries without having to use Python and the connector.
  - E.g. just checking if the data is correct "SELECT * FROM User LIMIT 5"
  - Open the MySQL terminal by typing `sudo mysql` , 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
- Remember to keep track of activity-ids when inserting trackpoints, so that each trackpoint has the correct foreign key to the Activity table
- Remember foreign key cascade rules

- How to use and take advantage of the datetime data type in your queries can be found here.
- Using variables in MySQL may come in handy
- As stated, using Haversine for calculating distance is recommended
- As stated, using Tabulate for printing tables in your Python program is recommended
- Optional - if you want to visualize the data in the dataset, you can get inspired here.