Report - template

Assignment 2 - MySQL

Group: 51
Students: Henrik Borge, Torbjørn Grande, Sondre Rogde

Introduction

Briefly explain the task and the problems you have solved. How did you work as a group? If you used Git, a link to the repository would be nice.

The task was to implement a structure for storing data on activities. Each activity was related to a user and multiple trackpoints. The trackpoints contains a timestamp with the coordinates of the user's position along with altitude. This was done keeping in mind that we were storing data for an application similar to Strava (more on this under Discussion).

All team members knew each other well from before, so working as a group posed no problems. Being a team of three people really helped in discussing the technical details of how to store the data and what we had to consider. For most problems in part 2, we implemented a simple solution assuming the data was clean and made sure that solution worked. After implementing this simple solution, we expanded on it to deal with edge cases, invalid data and other things that could invalidate the output. This is discussed in greater detail under Results and Discussion.

We used Github for code collaboration and version control. The repository on Github can be found here: https://github.com/Sondringsen/StoreDistribuerteOvinger. The repository should be publicly available, but please let us know if there is something wrong with the access. The repository also contains a README.md containing all documentation required for running the code.

Results

Add your results from the tasks, both as text and screenshots. Short sentences are sufficient.

Small excerpts from the tables:

First 10 entries of User

id	has_labels
000	0
001	0
002	0
003	0
004	0
005	0
006	0
007	0
008	0
009	0
010	1

First 10 entries of Activity

id	user_id	walk	bike	bus	taxi	car	subway	train	airplane	boat	run	motorcycle	valid_activity	start_date_time	end_date_time
1	000	0	0	0	0	0	0	0	0	0	0	0	1	2009-04-12 07:33:03	2009-04-12 17:24:59
2	000	0	0	0	0	0	0	0	0	0	0	0	1	2009-05-10 02:02:06	2009-05-10 06:20:11
3	000	0	0	0	0	0	0	0	0	0	0	0	1	2008-12-01 11:18:27	2008-12-01 11:35:15
4	000	0	0	0	0	0	0	0	0	0	0	0	1	2009-06-21 10:14:07	2009-06-21 14:47:12
5	000	0	0	0	0	0	0	0	0	0	0	0	1	2008-12-11 12:14:32	2008-12-11 12:24:32
6	000	0	0	0	0	0	0	0	0	0	0	0	1	2009-06-29 01:16:30	2009-06-29 11:13:12
7	000	0	0	0	0	0	0	0	0	0	0	0	1	2009-05-09 18:16:36	2009-05-09 18:30:36
8	000	0	0	0	0	0	0	0	0	0	0	0	1	2009-07-03 00:28:00	2009-07-03 08:37:23
9	000	0	0	0	0	0	0	0	0	0	0	0	1	2009-06-07 07:52:55	2009-06-07 09:46:03
10	000	0	0	0	0	0	0	0	0	0	0	0	1	2009-06-11 22:12:04	2009-06-11 22:34:54

First 10 entries of TrackPoint

id	activity_id	lat	lon	altitude	transportation_mode	date_days	date_time
1	1	40	116.327	105		39915.3	2009-04-12 07:33:03
2	1	40.0002	116.327	80		39915.3	2009-04-12 07:33:09
3	1	40.0001	116.327	99		39915.3	2009-04-12 07:33:14
4	1	40	116.327	109		39915.3	2009-04-12 07:33:19
5	1	40	116.327	111		39915.3	2009-04-12 07:33:24
6	1	40	116.327	114		39915.3	2009-04-12 07:33:29
7	1	39.9999	116.327	120		39915.3	2009-04-12 07:33:34
8	1	39.9997	116.327	125		39915.3	2009-04-12 07:33:39
9	1	39.9997	116.327	126		39915.3	2009-04-12 07:33:44
10	1	39.9995	116.327	127		39915.3	2009-04-12 07:33:49

Question 1:

```
Table User has 182 rows
Table Activity has 24971 rows
Table TrackPoint has 9681756 rows
```

Question 2:

Average activities per user

avg_activity_count
144.341

Question 3:

Users with the most activities

user_id	activity_count
163	3640
153	2294
128	2271
062	1037
085	946
167	854
068	853
025	715
144	569
075	509
126	450
052	404
041	399
084	388
004	346
140	345
010	335
112	307
147	291
017	265

Question 4:

Users who have taken taxi

id
010
020
021
052
056
058
062
065
068
075
078
080
082
084
085
091
098
100
102
104
105
111
114
118
126
128
139
147
153
154
161
163
167
175
179

Question 5:

For this question our implementation worked very well since we only had to sum over all the binary variables in the Activity table.

Number of times each transportation mode has been used

walk	bike	bus	taxi	car	subway	train	airplane	boat	run	motorcycle
5700	1850	2726	1126	934	764	293	16	5	6	2

Question 6a:

```
Number of activities per year
  year    activity_count
------  ----------------
  2008             11210
  2009              7739
  2007              2021
  2011              1873
  2010              1511
  2012               616
  2000                 1
```

Question 6b:

```
Time spent on activities per year
  year    time_spent
------  ------------
  2008     15128.8
  2009     13576
  2007      4055.53
  2010      1718.03
  2011      1565.31
  2012       719.884
  2000         0.0511
```

Question 7:

For this question only 10km moves between two trackpoints were allowed. This was to avoid any faulty data where two consecutive trackpoints was too far from each other. For this specific user, it did not matter, however, it could matter for other users. Most of this task is done using pandas.

```
Total kilometers traveled by user 112 in 2008:
202.89423691177663
```

Question 8:

For this question only altitudes between -300 and 50,000 feet were allowed. All altitudes of -777 altitudes were dropped. Also, if the altitude changed with more than 300 feet it was discarded. For this question, the only terms in the sum are the terms where there was a positive difference in altitude between two trackpoints, i.e., where the user ascends. Most of this task is done using pandas.

```
Users who have gained the most meters
user_id
128    429163.2768
153    424721.7312
004    248855.4840
163    174994.2144
003    167510.1552
085    159698.4360
030    131727.8544
144    114739.2168
084    106898.5416
039    106203.5976
167     99268.7880
041     97915.1712
002     87255.7056
000     85150.4520
126     68920.7664
062     64691.0568
025     62745.8232
140     49622.0496
013     44283.7824
028     43945.4544
dtype: float64
```

Question 9:

We found that most users have invalid activities. This task was also done using some pandas.

```
 Invalid activities per user (only includes users who has invalid activities)
    user_id  invalid_activity_count
0       000                     101
1       001                      45
2       002                      98
3       003                     179
4       004                     219
..      ...                     ...
168     176                       8
169     178                       0
170     179                      28
171     180                       2
172     181                      14

 [173 rows x 2 columns]
```

Question 10:

For this question we allowed the latitude to be between 39.915 and 39.917 and the longitude to be between 116.396 and 116.398.

```
Users who have registered activities in the Forbidden City
  user_id
---------
      004
      018
      019
      131
```

Question 11:

```
Users most used transportation mode
id
010    walk
020    bike
021    walk
052     bus
053    walk
       ...
167    walk
170    walk
174     car
175     bus
179    walk
Length: 69, dtype: object
```

Discussion

Discuss your solutions. Did you do anything differently than how it was explained in the assignment sheet, in that case why and how did that work? Were there any pain points or problems? What did you learn from this assignment?

What we did differently

We did some things a bit differently than what was described in the given description. They were only minor changes implemented to make the solution more realistic for an app similar to Strava. In Strava you can have multiple transportation modes per activity. You can for instance run and cycle in one activity in Strava. To better meet this requirement the table storing activities contained an column for each transportation mode stored as a BIT(). A different solution is to store a comma-separated list, but this is seen as an antipattern in relational databases. In addition, we stored the transportation mode for a given trackpoint as a string (VARCHAR). Trackpoints can only have one transportation mode since it is just a point in time and

not an interval. Please see the figures below for a detailed description of the tables and datatypes.

Column	Datatype
id	VARCHAR(4)
has_labels	BIT(1)

Column	Datatype
id	SMALLINT()
user_id	VARCHAR(4)
walk	BIT(1)
bike	BIT(1)
bus	BIT(1)
taxi	BIT(1)
car	BIT(1)
subway	BIT(1)
train	BIT(1)
airplane	BIT(1)
boat	BIT(1)
run	BIT(1)
motorcycle	BIT(1)
valid_activity	BIT(1)
start_date_time	DATETIME
end_date_time	DATETIME

Column	Datatype
id	INT
activity_id	SMALLINT
lat	DOUBLE(8,6)
lon	DOUBLE(9,6)
altitude	MEDIUMINT
transportation_mode	VARCHAR(30)
date_days	DOUBLE(15,10)
date_time	DATETIME

Handling messy data

To make the functionality most similar to Strava we let any activity be defined by the .plt files with start time and end time equal to the min and max timestamp of that plt.file respectively. If an activity was labeled, but did not correspond to any TrackPoints, we added it as an activity, but flagged it as invalid. We did this because

you can add activities in Strava without having any GPS tracking (i.e. no trackpoints). Before inserting the data, any trackpoints with highly unlikely values, for instance, the altitude was restricted to be between -300 and 30,000 feet were removed. The coordinates were also verified to be between -90 and 90 and -180 and 180 for latitude and longitude respectively. There is also some consecutive trackpoints which does not make sense, for instance, where the distance between them is unrealistically high. This is not handled when inserting the data but is handled during the queries where the distance between trackpoint-coordinates must be within a certain range. The same goes for altitude.

Pain points

Most of the exercise was completed without any great obstacles. The only pain point experienced was during part 2 with some of the more complicated queries. Those queries often required extracting large chunks of data and manipulating it in pandas. Since extracting and manipulating such large quantities of data is computationally intensive and time consuming it required careful attention to detail when coding to avoid spending to long debugging.

What we learned

You learn a lot of things doing projects like these. The most important one and relating to this course would be how to actually implement and maintain a database. Since we did not have access to a virtual machine from the start, we maintained a database locally hosted using Docker. Experience and knowledge about these things are hard to gain from a book and is best learned through doing projects. These skills are also essential for future employers. Additionally, we gained more experience with working with data in python and using pandas which is also highly appreciated by employers and very applicable to all sorts of problems be it academic or professional.

Feedback

Optional - give us feedback on the task if you have any. The assignment is new this semester and we would love to improve if there were any problems.

This was a very interesting exercise with the appropriate workload and difficulty.