Report - template

Assignment 2 - MySQL

Group: 48
Students: Anders Normann Hermanrud

Contents

Introduction

Github: https://github.com/AndersNormannHermanrud/TDT4225-TASK2
The task was to create a database that contained multiple users, all their activities and the trackpoints for the activities. Then the task was to create queries to get key information from the dataset. The challenge was to create efficient queries and code, so that the runtime would become acceptable on such a large dataset.

Results

Task 1

Database

Note the on delete cascade, since an activity or trackpoint cannot exist without a parent. The layout of the database implies that a User can delete itself or an activity, and then there is no reason to keep the objects children.

```
queries = []
queries.append("""
CREATE TABLE IF NOT EXISTS User (
                id INT NOT NULL PRIMARY KEY,
                has_labels BOOL NOT NULL)
""")
queries.append("""
CREATE TABLE IF NOT EXISTS Activity (
                id BIGINT NOT NULL PRIMARY KEY,
                user_id INT NOT NULL,
                transportation_mode VARCHAR(30),
                start_date_time VARCHAR(30),
                end_date_time VARCHAR(30),
                FOREIGN KEY (user_id)
                    REFERENCES User(id)
                    ON DELETE CASCADE)
""")
queries.append("""
CREATE TABLE IF NOT EXISTS TrackPoint (
                id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
                activity_id BIGINT NOT NULL,
                lat DOUBLE NOT NULL,
                lon DOUBLE NOT NULL,
                altitude INT NOT NULL,
                date_days DOUBLE NOT NULL,
                date_time DATETIME,
                FOREIGN KEY (activity_id)
                    REFERENCES Activity(id)
                    ON DELETE CASCADE)
""")
```

Insertion

Here I read in the data user by user to not use too much memory at once, then I order the data as a dataframe and uses the pd.to_sql to write data in bulk.
I chose to use pandas for this reason, and pandas speed when modifying data column wise

compared to python loops.
I diverged from pandas in cases where I would append a few lines many times for computational reasons, and instead write bulks of data to a frame and then into the database.

User and a secondary connection to the DB as validation

```
PS C:\Users\ander> docker exec -it gps-mysql mysql -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 30
Server version: 8.1.0 MySQL Community Server - GPL

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use gps
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> SELECT * FROM User LIMIT 10
    -> ;
+----+------------+
| id | has_labels |
+----+------------+
|  0 |          0 |
|  1 |          0 |
|  2 |          0 |
|  3 |          0 |
|  4 |          0 |
|  5 |          0 |
|  6 |          0 |
|  7 |          0 |
|  8 |          0 |
|  9 |          0 |
+----+------------+
10 rows in set (0.00 sec)

mysql>
```

Activity

```
mysql> SELECT * FROM Activity LIMIT 10;
+------------------+---------+--------------------+---------------------+---------------------+
| id               | user_id | transportation_mode | start_date_time     | end_date_time       |
+------------------+---------+--------------------+---------------------+---------------------+
| 20000101231219163 |     163 |                    | 2000-01-01 23:12:19 | 2000-01-01 23:15:22 |
| 20070412093132142 |     142 |                    | 2007-04-12 09:31:31 | 2007-04-12 11:33:40 |
| 20070412101853161 |     161 |                    | 2007-04-12 10:18:53 | 2007-04-12 10:23:15 |
| 20070412102116163 |     163 | bike               | 2007-04-12 10:21:16 | 2007-04-12 14:56:56 |
| 20070412102325161 |     161 |                    | 2007-04-12 10:23:24 | 2007-04-12 10:26:25 |
| 20070412134621097 |      97 |                    | 2007-04-12 13:46:21 | 2007-04-12 14:35:33 |
| 20070413005306163 |     163 |                    | 2007-04-13 00:53:05 | 2007-04-13 07:18:01 |
| 20070413013238142 |     142 |                    | 2007-04-13 01:32:37 | 2007-04-13 07:18:31 |
| 20070413105648161 |     161 |                    | 2007-04-13 10:56:47 | 2007-04-13 15:02:47 |
| 20070413150314161 |     161 |                    | 2007-04-13 15:03:14 | 2007-04-13 15:05:57 |
+------------------+---------+--------------------+---------------------+---------------------+
10 rows in set (0.00 sec)

mysql>
```

TrackPoint

```
mysql> SELECT * FROM TrackPoint LIMIT 10;
+----+-----------------+----------+------------+----------+-----------------+---------------------+
| id | activity_id     | lat      | lon        | altitude | date_days       | date_time           |
+----+-----------------+----------+------------+----------+-----------------+---------------------+
|  1 | 20081023025304000 | 39.984702 | 116.318417 |      492 | 39744.1201851852 | 2008-10-23 02:53:04 |
|  2 | 20081023025304000 | 39.984683 |  116.31845 |      492 | 39744.1202546296 | 2008-10-23 02:53:09 |
|  3 | 20081023025304000 | 39.984686 | 116.318417 |      492 |    39744.1203125 | 2008-10-23 02:53:15 |
|  4 | 20081023025304000 | 39.984688 | 116.318385 |      492 | 39744.1203703704 | 2008-10-23 02:53:20 |
|  5 | 20081023025304000 | 39.984655 | 116.318263 |      492 | 39744.1204282407 | 2008-10-23 02:53:24 |
|  6 | 20081023025304000 | 39.984611 | 116.318026 |      493 | 39744.1204861111 | 2008-10-23 02:53:29 |
|  7 | 20081023025304000 | 39.984608 | 116.317761 |      493 | 39744.1205439815 | 2008-10-23 02:53:35 |
|  8 | 20081023025304000 | 39.984563 | 116.317517 |      496 | 39744.1206018519 | 2008-10-23 02:53:40 |
|  9 | 20081023025304000 | 39.984539 | 116.317294 |      500 | 39744.1206597222 | 2008-10-23 02:53:44 |
| 10 | 20081023025304000 | 39.984606 | 116.317065 |      505 | 39744.1207175926 | 2008-10-23 02:53:50 |
+----+-----------------+----------+------------+----------+-----------------+---------------------+
10 rows in set (0.00 sec)

mysql>
```

Task 2

TASK 1

Users: 182, Activities: 16048, Trackpoints: 9681756

TASK 2

Average 55963.9075, Max 1010325, Min 17

TASK 3

User 128 with 2102 activities

User 153 with 1793 activities
User 25 with 715 activities
User 163 with 704 activities
User 62 with 691 activities
User 144 with 563 activities
User 41 with 399 activities
User 85 with 364 activities
User 4 with 346 activities
User 140 with 345 activities
User 167 with 320 activities
User 68 with 280 activities
User 17 with 265 activities
User 3 with 261 activities
User 14 with 236 activities

TASK 4

User (175,)
User (112,)
User (62,)
User (128,)
User (85,)

TASK 5

User 128 with 8 transportation modes
User 62 with 6 transportation modes
User 85 with 4 transportation modes
User 139 with 3 transportation modes
User 115 with 3 transportation modes
User 102 with 3 transportation modes
User 65 with 3 transportation modes
User 112 with 3 transportation modes
User 20 with 3 transportation modes
User 126 with 3 transportation modes

TASK 7

User 20, activity bike, time 610
User 115, activity car, time 60
User 115, activity car, time 83
User 115, activity car, time 68

User 115, activity car, time 84

TASK 9

Top 15 users that gained the most altitude (ordered left to right, top to bottom):

128, 2135455),	153, 1820766,	4, 1089358,	41, 789890,
3, 766613,	85, 714049,	163, 673439,	62, 596103,
144, 588771,	30, 576377,	39, 481311,	84, 430319,
0, 398638,	2, 377947,	167, 370647	

TASK 10

	user_id	transportation_mode	travel_date	distance
0	128		22/02/2009	36649.07
1	128	airplane	22/02/2009	1112.247
2	128	bike	28/06/2008	70.20009
3	62	bus	25/09/2008	1535.96
4	128	car	05/03/2009	19215.13
5	128	subway	23/11/2008	1995.616
6	128	taxi	05/03/2009	19181.91
7	128	walk	07/11/2008	1989.629

TASK 11

index	user_id	inv_act_count
0	0	101
1	1	45
2	2	98
3	3	179
4	4	219
.. ...	...	
166	175	4
167	176	8
168	179	28
169	180	2
170	181	14

171 Rows, some users have no invalid activities.

TASK 12

Index	Id	Trns_mode

0	10	taxi
1	20	bike
3	56	bike
4	60	walk
9	62	walk
10	64	bike
11	65	bike
13	67	walk
14	73	walk
15	78	walk
16	81	bike
17	82	walk
18	84	walk
19	85	bus
22	86	walk
23	87	walk
24	89	car
25	98	taxi
26	102	bike
28	107	walk
29	108	walk
30	111	taxi
32	112	walk
33	115	car
35	117	walk
36	126	bike
41	128	car
45	139	bike
47	153	walk
48	163	bike
50	167	bike
51	175	bus

Discussion

The first difference between the assignment and my solution is that I chose to have the User.id as an int and not string, since each user is defined by a unique string that always can be parsed to an int, and an int is more compact.

I also chose to use sqlalchemy as a database connector, since it can be combined with pandas to write and read data in bulk directly to dataframes. The disadvantage here is

that I had to use time learning a new framework, but the advantage is more simple code that can to more advanced data manipulation in an efficient readable manner.

From this assignment I further learned how to bulk up operations into pieces that can fit in memory, since my personal computer has little of it when running the docker image.
I also got a lot of training in writing mysql, and learned multiple useful commands that can move the data manipulations from python back to sql, reducing the amount of data fetched in the output and reducing runtime.

One of my pain points was making tasks 8 and 10 work, since I quickly got code that calculated correctly, but had a runtime that made them unsuitable to tackle the entire dataset. Here I spent a lot of time.

I also had problems with task 8, I got a code that works on small datasets, but uses 1-3 hours on the full dataset. I sadly did not have this much time left when I got the solution working. I first used a hard coded approach that would have used days to run but changed it after letting it run overnight with no results. This approach is included as an comment in the submission.

I used a Euclidian distance estimate in task 10, that adds an approximated length based on latitude and longitude so it would run magnitudes faster than with just haversine. The loss is a tiny bit of accuracy.

Feedback

This task worked great for me, since I have quite a good pc, but if a student that does not have a great pc and does not get a group with a virtual machine gets the task, they can face really bad runtime and memory errors.
I think setting up the mysql docker image is a nice repetition, and is a nice practical skill to have