TDT4225 - Assignment 3

Group 26
Ole Jonas Liahagen
Martin Johannes Nilsen

22nd October 2021

Introduction

In this assignment we were to insert and process data from the Geolife GPS Trajectory dataset. The data given to us consisted of 182 unique users, each with their own registered activites. These activities each consists of a list of tracking points containing GPS coordinates and timestamps. Activities could either be labeled with a transportation mode or not at all. To solve the assigment, we have used a combination of database queries and Python to get the data desired for answering each task.

Only minimal changes to the Python script were required for inserting data to a mongodb database instead of the MySQL database in the former assignment. We had already created dataclasses, and as classes in Python are stored as dictionaries, we could easily insert these to the db with a few adjustments of the code. After we had built the database structure and inserted data, we split up and solved the assignments separately, with the possibility to ask each other questions and share some thoughts to check that we agreed on the solutions. We used GitHub[1] as a version control system for our code, and used Overleaf to collaboratively write and compile this LaTeX report.

[1] https://github.com/MartinJohannesNilsen/tdt4225-assignment3.git

Results

Part 1

The top 10 documents in each collection:

```
User:
{'_id': 000, 'has_labels': False, 'activities': [...]}
{'_id': 001, 'has_labels': False, 'activities': [...]}
{'_id': 002, 'has_labels': False, 'activities': [...]}
{'_id': 003, 'has_labels': False, 'activities': [...]}
{'_id': 004, 'has_labels': False, 'activities': [...]}
{'_id': 005, 'has_labels': False, 'activities': [...]}
{'_id': 006, 'has_labels': False, 'activities': [...]}
{'_id': 007, 'has_labels': False, 'activities': [...]}
{'_id': 008, 'has_labels': False, 'activities': [...]}
{'_id': 009, 'has_labels': False, 'activities': [...]}
```

The first 10 documents in the user collection. Activities is a list of all activity ids.

```
Activity:
{'_id': 0, 'user_id': '000', 'transportation_mode': None, 'start_date_time': datetime.datetime(2008, 10, 23, 2, 53, 4), 'end_date_time': datetime.datetime(2008, 10, 23, 11, 11, 12)}
{'_id': 1, 'user_id': '000', 'transportation_mode': None, 'start_date_time': datetime.datetime(2008, 10, 24, 2, 9, 59), 'end_date_time': datetime.datetime(2008, 10, 24, 2, 47, 6)}
{'_id': 2, 'user_id': '000', 'transportation_mode': None, 'start_date_time': datetime.datetime(2008, 10, 26, 13, 44, 7), 'end_date_time': datetime.datetime(2008, 10, 26, 15, 4, 7)}
{'_id': 3, 'user_id': '000', 'transportation_mode': None, 'start_date_time': datetime.datetime(2008, 10, 27, 11, 54, 49), 'end_date_time': datetime.datetime(2008, 10, 27, 12, 5, 54)}
{'_id': 4, 'user_id': '000', 'transportation_mode': None, 'start_date_time': datetime.datetime(2008, 10, 28, 0, 38, 26), 'end_date_time': datetime.datetime(2008, 10, 28, 5, 3, 42)}
{'_id': 5, 'user_id': '000', 'transportation_mode': None, 'start_date_time': datetime.datetime(2008, 10, 29, 9, 21, 38), 'end_date_time': datetime.datetime(2008, 10, 29, 9, 30, 28)}
{'_id': 6, 'user_id': '000', 'transportation_mode': None, 'start_date_time': datetime.datetime(2008, 10, 29, 9, 30, 38), 'end_date_time': datetime.datetime(2008, 10, 29, 9, 46, 43)}
{'_id': 7, 'user_id': '000', 'transportation_mode': None, 'start_date_time': datetime.datetime(2008, 11, 3, 10, 13, 36), 'end_date_time': datetime.datetime(2008, 11, 3, 10, 16, 1)}
{'_id': 8, 'user_id': '000', 'transportation_mode': None, 'start_date_time': datetime.datetime(2008, 11, 3, 23, 21, 53), 'end_date_time': datetime.datetime(2008, 11, 4, 3, 31, 8)}
{'_id': 9, 'user_id': '000', 'transportation_mode': None, 'start_date_time': datetime.datetime(2008, 11, 10, 1, 36, 37), 'end_date_time': datetime.datetime(2008, 11, 10, 3, 46, 12)}
```

The first 10 documents in the activity collection

```
TrackPoint:
{'_id': 0, 'latitude': '39.984702', 'longitude': '116.318417', 'altitude': '492', 'date_days': '39744.1201851852', 'date_time': datetime.datetime(2008, 10, 23, 2, 53, 4), 'activity_id': 0}
{'_id': 1, 'latitude': '39.984683', 'longitude': '116.31845', 'altitude': '492', 'date_days': '39744.1202546296', 'date_time': datetime.datetime(2008, 10, 23, 2, 53, 10), 'activity_id': 0}
{'_id': 2, 'latitude': '39.984686', 'longitude': '116.318417', 'altitude': '492', 'date_days': '39744.1203125', 'date_time': datetime.datetime(2008, 10, 23, 2, 53, 15), 'activity_id': 0}
{'_id': 3, 'latitude': '39.984688', 'longitude': '116.318385', 'altitude': '492', 'date_days': '39744.1203703704', 'date_time': datetime.datetime(2008, 10, 23, 2, 53, 20), 'activity_id': 0}
{'_id': 4, 'latitude': '39.984655', 'longitude': '116.318263', 'altitude': '492', 'date_days': '39744.1204282407', 'date_time': datetime.datetime(2008, 10, 23, 2, 53, 25), 'activity_id': 0}
{'_id': 5, 'latitude': '39.984611', 'longitude': '116.318026', 'altitude': '493', 'date_days': '39744.1204861111', 'date_time': datetime.datetime(2008, 10, 23, 2, 53, 30), 'activity_id': 0}
{'_id': 6, 'latitude': '39.984608', 'longitude': '116.317761', 'altitude': '493', 'date_days': '39744.1205439815', 'date_time': datetime.datetime(2008, 10, 23, 2, 53, 35), 'activity_id': 0}
{'_id': 7, 'latitude': '39.984563', 'longitude': '116.317517', 'altitude': '496', 'date_days': '39744.1206018519', 'date_time': datetime.datetime(2008, 10, 23, 2, 53, 40), 'activity_id': 0}
{'_id': 8, 'latitude': '39.984539', 'longitude': '116.317294', 'altitude': '500', 'date_days': '39744.1206597222', 'date_time': datetime.datetime(2008, 10, 23, 2, 53, 45), 'activity_id': 0}
{'_id': 9, 'latitude': '39.984606', 'longitude': '116.317065', 'altitude': '505', 'date_days': '39744.1207175926', 'date_time': datetime.datetime(2008, 10, 23, 2, 53, 50), 'activity_id': 0}
```

The first 10 documents in the trackpoint collection

Part 2

Task 1

The amount of users, activities and trackpoints in the dataset after inserted to the database:

```
Task 1
  n_users    n_activities    n_trackpoints
---------  --------------  ---------------
      182           16048          9681756
```

Task 2

The *average*, *minimum* and *maximum* number of activities per user. It is important to notice that some users have tracked activities, but because we exclude all activities with more than 2500 trackpoints, some users end up having zero activities.

```
Task 2
  average    maximum    minimum
---------  ---------  ---------
  88.1758       2102          0
```

Task 3

Top 10 users with the highest number of activities:

```
Task 3
  Id    Count
----  -------
 128     2102
 153     1793
 025      715
 163      704
 062      691
 144      563
 041      399
 085      364
 004      346
 140      345
```

Task 4

The number of users that have started the activity in one day and ended the activity the next day:

```
Task 4
  Users with multi-day activities
---------------------------------
                               98
```

Task 5

Activities that are registered multiple times, i.e. the same start- and end-time, but different ids. The dataset includes zero duplicated activities.

```
Task 5
activities registred multiple times
-----------------------------------
```

Task 6

Users who have been close to the infected person at position (39.97548, 116.33031) the 24th of august 2008, at 15:38.

```
Task 6
Users in contact: ['073']
```

Task 7

In this task we were requested to find all users that have never taken a taxi. The table includes a column named *labeled* to show if the concerning user is labelled or not. If the user is not labelled, they will obviously never have a labelled taxi ride registerred. The result set contains 172 rows, which indicates that only 10 users have taken a taxi and labelled the ride (182-172). Out of the 172 who have never taken a taxi, 69 users have labelled their data.

```
Task 7
  Users that have never taken a taxi  labeled
------------------------------------  ---------
                                 020  True
                                 021  True
                                 052  True
                                 053  True
                                 056  True
                                 059  True
                                 060  True
                                 064  True
                                 065  True
                                 067  True
                                 068  True
                                 069  True
                                 073  True
                                 075  True
                                 076  True
                                 081  True
                                 082  True
                                 084  True
                                 086  True
                                 087  True
                                 088  True
                                 089  True
                                 091  True
                                 092  True
                                 096  True
                                 097  True
                                 100  True
                                 101  True
                                 102  True
                                 104  True
                                 105  True
                                 106  True
                                 107  True
                                 108  True
                                 110  True
                                 112  True
                                 114  True
                                 115  True
                                 116  True
                                 117  True
                                 118  True
                                 124  True
                                 125  True
                                 126  True
                                 129  True
                                 136  True
                                 138  True
                                 139  True
                                 141  True
                                 144  True
                                 147  True
                                 153  True
                                 154  True
                                 161  True
                                 167  True
                                 170  True
                                 174  True
                                 175  True
                                 179  True
```

```
000   False
001   False
002   False
003   False
004   False
005   False
006   False
007   False
008   False
009   False
011   False
012   False
013   False
014   False
015   False
016   False
017   False
018   False
019   False
022   False
023   False
024   False
025   False
026   False
027   False
028   False
029   False
030   False
031   False
032   False
033   False
034   False
035   False
036   False
037   False
038   False
039   False
040   False
041   False
042   False
043   False
044   False
045   False
046   False
047   False
048   False
049   False
050   False
051   False
054   False
055   False
057   False
061   False
063   False
066   False
070   False
071   False
072   False
074   False
077   False
079   False
083   False
090   False
093   False
094   False
095   False
```

```
099   False
103   False
109   False
113   False
119   False
120   False
121   False
122   False
123   False
127   False
130   False
131   False
132   False
133   False
134   False
135   False
137   False
140   False
142   False
143   False
145   False
146   False
148   False
149   False
150   False
151   False
152   False
155   False
156   False
157   False
158   False
159   False
160   False
162   False
164   False
165   False
166   False
168   False
169   False
171   False
172   False
173   False
176   False
177   False
178   False
180   False
181   False
```

Task 8

All types of transportation modes and how many distinct users that have used them:

```
Task 8
transportation_mode      Count
---------------------  -------
airplane                     1
bike                        19
boat                         1
bus                         12
car                          8
run                          1
subway                       4
taxi                        10
train                        2
walk                        31
```

Task 9

a) The year and month with the most activities:

```
Task 9a
  Year with most activities    Month with most activities in this year
---------------------------  -----------------------------------------
                       2008                                         11
```

b) The two users with the most activities in the month and year found from 9a (11-2008), and how many recorded hours they have. The user with the second most activities has more registered hours than the user with the most registered activities.

```
Task 9b
  uid    n_activities    hours
-----  --------------  -------
  062             130  47.3136
  128              75  68.2211
```

Task 10

The total distance (in km) walked in 2008, by user with id=112. This distance seems small, but we think it is correct after going over our code and queries. Looking over the trackpoints associated with user 112, it seems they have not moved much around by foot this year. This might be due to our method of labeling the data, so that certain walks were not accepted by our filtering. However, this is only speculation.

```
Task 10
User_id: 112
Distance in 2008: 1.3497848643311852
```

Task 11

The top 20 users who have gained the most altitude meters:

```
Task 11
  uid     m gained
-----  -----------
  128  650951.9973
  153  554960.6231
    4  332036.3184
   41  240768.8657
    3  233663.6424
   85  217643.3849
  163  205274.3705
   62  181693.2917
  144  179441.5245
   30  175679.7096
   39  146703.5928
   84  131161.2312
    0  121504.8624
    2  115198.2456
  167  112974.1546
   25  109158.5726
   37   99234.5894
  140   94846.2994
  126   83025.8358
   17   62581.3531
```

Task 12

In the figures below you will find the users who have invalid activities, and the number of invalid activities per user. An invalid activity is when the time difference between two trackpoints is 5 minutes or more.

Task 12

uid	Invalid activities
0	101
1	45
2	98
3	179
4	219
5	45
6	17
7	30
8	16
9	31
10	50
11	32
12	43
13	29
14	118
15	46
16	20
17	129
18	27
19	31
20	20
21	7
22	55
23	11
24	27
25	263
26	18
27	2
28	36
29	25
30	112
31	3
32	12
33	2
34	88

35	23
36	34
37	100
38	58
39	147
40	17
41	201
42	55
43	21
44	32
45	7
46	13
47	6
48	1
49	0
50	8
51	36
52	44
53	7
54	2
55	15
56	7
57	16
58	13
59	5
60	1
61	12
62	249
63	8
64	7
65	26
66	6
67	33
68	139
69	6
70	5
71	29
72	2
73	18
74	19
75	6
76	8
77	3
78	19
79	2
80	6
81	16
82	27
83	15
84	99
85	184
86	5
87	3
88	11
89	40
90	3
91	63
92	101
93	4
94	16
95	4
96	35
97	14
98	5
99	11
100	3
101	46
102	13
103	24
104	97
105	9
106	3
107	1
108	5

109	3
110	17
111	26
112	67
113	1
114	3
115	58
116	0
117	3
118	3
119	22
120	0
121	4
122	6
123	3
124	4
125	25
126	105
127	4
128	720
129	6
130	8
131	10
132	3
133	4
134	31
135	5
136	6
137	0
138	10
139	12
140	86
141	1
142	52
143	0
144	157
145	5
146	7
147	30
148	0
149	0
150	16
151	1
152	2
153	557
154	14
155	30
156	0
157	9
158	9
159	5
160	0
161	7
162	9
163	233
164	6
165	2
166	2
167	134
168	19
169	9
170	2
171	3
172	9
173	5
174	54
175	4
176	8
177	0
178	0
179	28
180	2
181	14

Discussion

To start off the assignment, we discussed how to structure our data to make our queries the most effective. The decision landed on every user having a list of activity ids as a one to many relation. Each trackpoint also has a reference to the activity that it belongs to, but not the other way around. The activities do not have a reference to each of their trackpoints. One could argue that including such a list could be beneficial due to our limit of 2500 trackpoints per activity, however, we did not see the need to add such a relation. It would just increase complexity and inserts needed. This leads to our database model being quite similar to the one for the MySQL assignment, with the exception of the arrays of activity ids in the user documents. Our choice of modeling the database this way may be influenced by our previous experience with relational databases and little experience with document databases. There are possibilities of denormalisation here, such as adding data from the trackpoints to the activities. There might even be a more compelling case for doing this rather than putting activity ids in the user documents. If one were to look at the user documents as a complete user object, one would not need activities to define a user, however, it could be argued that an activity needs a list of coordinates to track where the activity found place. There really is no right or wrong here, and it is mostly up to preference. Our preference leaned towards the approach we have implemented, and that is how our design came to be.

The design we chose made the tasks quite similar to the ones from the MySQL assignment, with some differences here and there. Seeing as joins are considered slow in document databases, these were mostly avoided in favor of doing application level joins on retrieved documents instead.

It is also worth mentioning that a teaching assistant fortunate enough to read both of our group assignments with a very good memory would notice that our answers to tasks 11 and 12 differ from the ones given in assignment 2. This is due to us finding and correcting some errors that were present in our previous delivery. Our old answer for task 11 did not perform a conversion from feet to meters, resulting in a very large amount of altitude meters gained for several users. This is now resolved. Task 12 was missing a way to skip past an activity's remaining trackpoints if said activity already had been classified as invalid. This resulted in an invalid activity being able to register as several invalid activities instead of just one. This check is now added, and the results should be correct now.

All in all, working with a document database was less of a hassle than expected, albeit a little tricky to begin with. But after some trial and error, we feel like we got the knack of it, and we are pretty certain we deliver a new and more correct solution this time around!

MySQL vs MongoDB

MongoDB, having a document-collection database format, differ from MySQL in multiple ways. The first obvious difference is the use of collections and documents, instead of tables and rows. This is more reminiscent of how objects are stored in code and can benefit comprehension of the data stored in the database. This also makes it easy to insert data from code to the database, since the JSON format is the standard way to store data in MongoDB. In python we used dictionaries to send data to the database, which is then easily translated to a document in the JSON format. Doing this in MySQL without middleware is a bit more cumbersome, requiring the user to insert data into all the concerning tables. If there is a large object to be inserted, this could span many tables if the database is normalised. The same problem arises when retrieving objects from a relational model. A user would have to assemble the object from each table before returning it for use in code.

The lack of an enforced schema is another thing differentiating MongoDB from MySQL. The flexibility may be a significant advantage for some scenarios, meanwhile in others the guarantee of a given format may be wanted or even required. It is worth mentioning, that while not mandatory, MongoDB does have the opportunity to enforce schemas with special flags in insertions and creations of collections.

After working with both types of systems, we do not feel there is a "better" or "worse" approach to the database model overall. Each system has it's up- and downsides. MongoDB is great for easy access to more complex objects, but slower for lookups in lots of tables. This could however, be overcome by denormalising data. Unfortunately this might lead to more complexity in making the database itself and maintaining it. MySQL keeps this simple and with a properly normalised database, there should be no reason to update fields in any other table than the one you are using at any given moment. Joins in a relational database are also often much faster due to built-in and optimised support for such operations.

As stated earlier - the two models are two different approaches at database systems, that work well in their own respective use-cases. For this assignment, we did not feel there was a "better" version among the two.