**Report**

Assignment 3 - MongoDB

**Group**: 34

**Students**: Pablo Díaz Viñambres, Guillermo García-Nalda Noval

**INTRODUCTION**

The purpose of this practice is to work with a dataset of trajectories, users, activities and trackpoints. We had to create tables, clean and insert data, and make queries to answer some questions using python programming and MongoDB functions. We simulated some features of Strava, a website where users can track activities like running, walking, biking, etc. and post them online with stats about their workout.

We had two tasks to do. In task 1 we had to focus on cleaning and inserting the data into defined tables, and in task 2 we had to focus on writing queries to the database to gain knowledge of the dataset.

In relation to the team, we worked mainly on one computer, but we also worked with GitHub, which allowed us to upload code repositories and code simultaneously.

Since this assignment was very similar to assignment 2, most of the application-side code was recycled, so most of the work was put into elaborating the MongoDB queries.

**RESULTS**

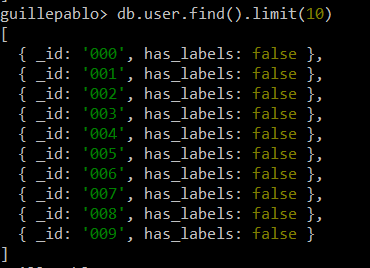
**Task 1**

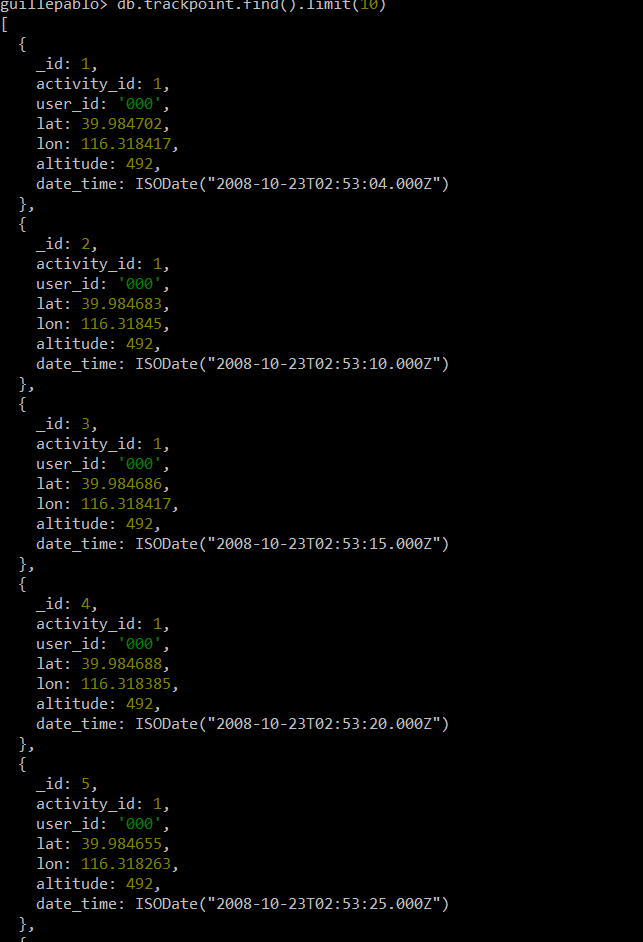
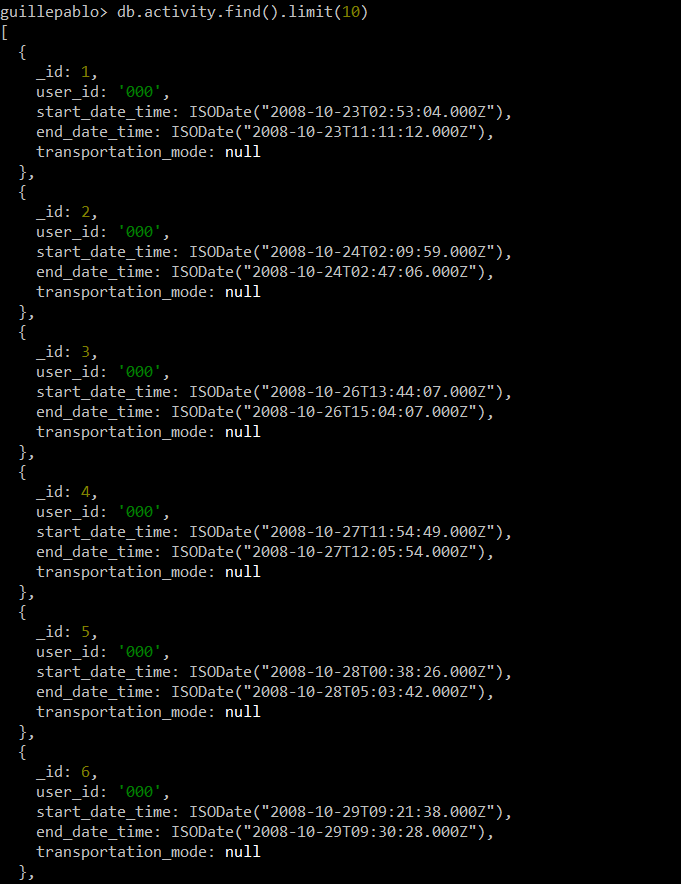
For the first task, we inserted all the data found on the dataset files. This time, we didn’t have to create the collections beforehand, since they can be created lazily when inserting objects into them. The data insertion was done similarly to the previous assignment, by first finding which users have labels and then doing a double loop on every user folder and activity subfolder found on the dataset.

This time, the batch insertion of trackpoints wasn’t done on activity packages. Instead, trackpoints were inserted in batches of equal length defined in the variable trackpoint\_batch\_size that was set to 100000. We opted for this approach because of the lack of foreign key constraints and a higher efficiency on the insertion.

We also introduced some *denormalization* to the schema, to make queries on the trackpoint collection easier, we added the user\_id attribute to it. This way, we compensate for the lack of joins, allowing us to do all the exercises on task 2 in only one query.

We now present 3 screenshots featuring the first 10 documents of every collection inside the DB:





**Task 2**

We now present a list of the results we got from executing the queries.

* Query 1 - How many users, trackpoints and activities are there in the dataset (after it is inserted into the database):

Total Users Total Activities Total Trackpoints

------------- ------------------ -------------------

182 16048 9681756

Query done in 5.65 seconds

* Query 2 - Find the average number of activities per user:

Average

---------

92.763

Query done in 0.03 seconds

* Query 3 - Find the top 20 users with the highest number of activities:

Users Activities

------- ------------

128 2102

153 1793

025 715

163 704

062 691

144 563

041 399

085 364

004 346

140 345

167 320

068 280

017 265

003 261

014 236

126 215

030 210

112 208

011 201

039 198

Query done in 0.02 seconds

* Query 4 - Find all users who have taken a taxi:

Taxi Users

------------

10

Query done in 0.05 seconds

* Query 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:

Transportation Mode Count

--------------------- -------

subway 133

airplane 3

bike 263

boat 1

walk 480

run 1

train 2

car 419

taxi 37

bus 199

Query done in 0.11 seconds

* Query 6
  + a) Find the year with the most activities:

Year Activities

------ ------------

2008 5895

Query done in 0.09 seconds

* + b) Is this also the year with the most recorded hours?:

Year Hours

------ -------

2009 11636

Query done in 0.10 seconds

As we see, 2008 with the most activities, but 2009 has more hours recorded

* Query 7 - Find the total distance (in km) walked in 2008, by user with id=112:

Total Distance

----------------

3316.68

Query done in 7.67 seconds

* Query 8 - Find the top 20 users who have gained the most altitude meters:

Top User ID Altitude gained

----- --------- -----------------

1 128 2.13567e+06

2 153 1.82074e+06

3 004 1.08936e+06

4 041 789924

5 003 766613

6 085 714053

7 163 673472

8 062 596107

9 144 588719

10 030 576377

11 039 481311

12 084 430319

13 000 398638

14 002 377503

15 167 370650

16 025 358132

17 037 325573

18 140 311176

19 126 272394

20 017 205319

Query done in 127.55 seconds

* Query 9 - Find all users who have invalid activities, and the number of invalid activities per user:

User Number of invalid activities

------ ------------------------------

000 101

001 45

002 98

003 179

004 219

005 45

006 17

007 30

008 16

009 31

010 50

011 32

012 43

013 29

014 118

015 46

016 20

017 129

018 27

019 31

020 20

021 7

022 55

023 11

024 27

025 263

026 18

027 2

028 36

029 25

030 112

031 3

032 12

033 2

034 88

035 23

036 34

037 100

038 58

039 147

040 17

041 201

042 55

043 21

044 32

045 7

046 13

047 6

048 1

050 8

051 36

052 44

053 7

054 2

055 15

056 7

057 16

058 13

059 5

060 1

061 12

062 249

063 8

064 7

065 26

066 6

067 33

068 139

069 6

070 5

071 29

072 2

073 18

074 19

075 6

076 8

077 3

078 19

079 2

080 6

081 16

082 27

083 15

084 99

085 184

086 5

087 3

088 11

089 40

090 3

091 63

092 101

093 4

094 16

095 4

096 35

097 14

098 5

099 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

117 3

118 3

119 22

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

138 10

139 12

140 86

141 1

142 52

144 157

145 5

146 7

147 30

150 16

151 1

152 2

153 557

154 14

155 30

157 9

158 9

159 5

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

179 28

180 2

181 14

Query done in 103.09 seconds

* Query 10 - Find users who have tracked an activity in the Forbidden City of Beijing:

User ID

---------

004

018

019

131

Query done in 5.28 seconds

* Query 11 - Find all users who have registered transportation\_mode and their most used transportation\_mode:

User Most used transportation mode

------ -------------------------------

010 taxi

020 bike

021 walk

052 bus

056 bike

058 car

060 walk

062 bus

064 bike

065 bike

067 walk

069 bike

073 walk

075 walk

076 car

078 walk

080 bike

081 bike

082 walk

084 walk

085 walk

086 car

087 walk

089 car

091 bus

092 bus

097 bike

098 taxi

101 car

102 bike

107 walk

108 walk

111 taxi

112 walk

115 car

117 walk

125 bike

126 bike

128 car

136 walk

138 bike

139 bike

144 walk

153 walk

161 walk

163 bike

167 bike

175 bus

Query done in 0.02 seconds

**DISCUSSION**

We observe that most of the queries are a bit slower compared to the ones in the previous assignment. However the queries where the Trackpoint table had to be joined with the Activity one, are executed faster, since we introduced denormalization making the process faster.

In this assignment we learned about the differences between SQL and NoSQL database systems, and learned to work with MongoDB.