Report - template

Assignment 3 - MongoDB

Group: 49
Students: Andreas Amundsen

Table of Contents

Introduction

I have solved the problem of iterating trough a large dataset and insert it into a MongoDB database. As with the previous assignment I chose to work alone.

Results

Solution to the tasks can be found in the file task2.py.
Images are screenshots from the 'Run' window in PyCharm

Task 1

```
users has 183 documents
activities has 16046 documents
trackpoints has 9676756 documents
```

Task 2

```
Average: 92.7514450867052
Min: 1
Max: 2102
```

Task 3

```
List of top 10 user with the most activities
User: 128 Activities: 2102
User: 153 Activities: 1793
User: 025 Activities: 715
User: 163 Activities: 704
User: 062 Activities: 691
User: 144 Activities: 563
User: 041 Activities: 399
User: 085 Activities: 364
User: 004 Activities: 346
User: 140 Activities: 345
```

Task 4

```
Number of users that have started an activity one day and ended it the next: 98
```

Task 5

No duplicate activities found

Task 6

For this task i used the [geoNear aggregation]

```
User close in time and space: 073
```

Task 7

Every user besides ’058’ has not taken a taxi

['135', '132', '104', '103', '168', '157', '150', '159', '166', '161', '102', '105', '133', '134', '160', '158', '167', '151', '169', '156', '024', '023', '015', '012', '079', '046', '041', '048', '077', '083', '084', '070', '013', '014', '022', '025', '071', '085', '049', '082', '076', '040', '078', '047', '065', '091', '096', '062', '054', '053', '098', '038', '007', '000', '009', '036', '031', '052', '099', '055', '063', '097', '090', '064', '030', '008', '037', '001', '039', '006', '174', '180', '173', '145', '142', '129', '116', '111', '118', '127', '120', '143', '144', '172', '181', '175', '121', '119', '126', '110', '128', '117', '153', '154', '162', '165', '131', '136', '109', '100', '107', '138', '164', '163', '155', '152', '106', '139', '101', '137', '108', '130', '089', '042', '045', '087', '073', '074', '080', '020', '027', '018', '011', '016', '029', '081', '075', '072', '086', '044', '088', '043', '017', '028', '010', '026', '019', '021', '003', '004', '032', '035', '095', '061', '066', '092', '059', '050', '057', '068', '034', '033', '005', '002', '056', '069', '051', '093', '067', '060', '094', '112', '115', '123', '124', '170', '177', '148', '141', '146', '179', '125', '122', '114', '113', '147', '178', '140', '176', '149', '171']

Task 8

```
taxi 1 distinct users
walk 9 distinct users
bike 2 distinct users
bus 1 distinct users
car 1 distinct users
```

Task 9

```
2008-11 had the most activities
User 062 has a total of 130 with a total duration of 47.25 hours recorded in november of 2008
User 128 has a total of 75 with a total duration of 68.2 hours recorded in november of 2008
```

User with the second most activites spent more hours than the user with the most activities

Task 10

```
0 kilometres
```

I found no match with entries from *label.txt* file of user 112 and therefore the result will be 0 kilometers.

Task 11

User ID	Altitude gained (m)
012	1660
122	1517.45
158	997
060	698.819
064	371
099	321.522
031	311
063	302.117
028	292
096	285
050	269
171	209.974
152	179
069	167.323
124	148
113	115
038	108.2
175	104.987
098	98.4252
088	95.2

Task 12

```
014 has 1 invalid activities
```

Discussion

Implementation decisions

Contrary to the previous exercise I matched an entry from labels.txt file, on the first and last line of a trackpoint file. Not suprisingly this reduced execution time, from 2.5 hours to about 15 minutes. I also experimented with the *insert_many()* function and found out it was far more effective to call the function after a single trackpoint file was parsed, rather than after every file belonging to every user was parsed. Although the limit for *insert_many()* increased to 100 000 in MongoDB 3.6 and the database would automatically split up inserts if this was exceeded, it became apparent that it is more effective to manually limit this. At one point I

went from a execution time of 21 minutes to just 4.5 minutes, just by adjusting where I called *insert_many()*. This time later increased to 15 minutes due to neccecary parsing.

I chose to make seperate collections for users, activities and trackpoint. A user would have references on ObjectID to a activity and activities would have refrence on ObjectID to a trackpoint. In addition, I added parent refrencing from trackpoint to activity and user and from activity to user. The parent referencing would take some extra exection time in terms of inserting, but I figured out that they would come in handy when solving the tasks, which they did.

I used GeoJSON to store longitude and latitude data, which allowed me to generate 2D spheres for each location. This meant that I could use the geoNear aggregation, which came in handy when solving task 6.

Pain problems

1. GeoJSON requires longitude and latitude to be inserted in that exact order, while in a trackpoint file latitude would be listed first. With this ordering error, 2D sphere indexing would return an error. This took me a long time to figure out.

2. Although execution time was reduced to just 15 minutes, I often came over implementation errors such as altitude not being inserted as a float in the database. This meant I had to fix it and insert the data again all over, which took time. I would have benefited from deciding on data types, before implementing the program.

Learning outcome

1. How many items you pass to *insert_many()* has a great effect on the total execution time.

2. MongoDB aggregations are very powerful, ecpecially when it comes to geo localization.

MySQL vs MongoDB

Key differences

In MongoDB a data record, such as a user, is stored as a document, consisting of key-value pairs in BSON format. Each document belongs to a collection. Every In MySQL a record is stored as a row in a table. MySQL uses Structured Query Language (SQL) for accessing the data while MongoDB uses aggregations, which shares similarities with SQL, but has a different format.

Pros and cons

Ease of use
MongoDB is more simple to understand and utilize, because it builds upon simple principles such as key-value pairs and documents. Using MySQL often requires knowledge of normalization and relational database design, which can take time to learn.

Flexibility
Because MongoDB is schemaless, it allows for iterativley changing the database over time. Both keys and value types of records belonging to the same collection can differ. In MySQL schemas are predefined which can lead to a lot of work later if types need to be changed. In this regard, M**o**ngoDB is far more flexibile.

Selecting, inserting and updating
MySQL is faster when it comes to selecting a large number of records, while MongoDB is more efficent at inserting and updating records. MySQL can only insert data row by row, while MongoDB can bulk insert using *insert_many()*.

Documentation and maturity
Even though MongoDB is well established, MySQL has the benefit from being almost 30 years old. Many IT professionals have experience with it and there is a lot of documentation online.

My prefered system

I found MongoDB to be my prefered system to use, mainly because of how easy it was to insert data and understand how it behaved. I enjoy MongoDB aggregations over SQL, because I found it easier to work with, ecpecially when using MongoDB Compass. Parent refrencing also helped a lot when solving the tasks. I assume you can do the same with MySQL, but it was very easy to simply just add an refrence in MongoDB.

Feedback

I really enjoy the freedom we get and the heavy focus on programming. Easily one of the more exciting assignments I have done at NTNU. I would highly recomend that you give students next year the same assignments.

In this assignment we were tasked to discuss implementation, pain points, learning outcome and differences between MySQL and MongoDB. Taking this into concideration, I feel like ”restricting” the report to 1-2 pages is too little.