CSCI 5408 ASSIGNMENT 1

By: Bhargav Dalal (B00785773) & Hardik Galiawala (B00777450)

MAY 23, 2018

DALHOUSIE UNIVERSITY
Submitted to: Suhaib Qaiser

Table of Contents

1. Objective:

The main purpose of this assignment is to review conventional RDBMS, learn Infrastructure Services on a cloud system, learn Elastic Search concepts and implementation, and learn installation of MySQL (RDBMS) on a Cloud system.

2. Source code:

Link to github account.

3. Task Description:

The task was to compare the performance of the RDBMS structure database and NOSQL structure database. The data was obtained from website. The data includes information about the number of bus stops(stops), trips of each bus(trips), bus names and routes(stoptimes). Four tasks were given to assess the performance. These four tasks were first performed on MYSQL (RDBMS structure-based database). Later, these tasks were performed on Elastic Search (NOSQL structure-based database).

We have installed conventional RDBMS (MySQL) in an instance which is created on EC2 AWS cloud infrastructure. Then we created a schema in MySQL where we created the structure of the tables along with the relationship. The dataset was given in the form of insert statements as well as .csv files. We used insert statements to populate the data in tables. So, by now our relational database is ready with the required data. Then we used MySQL Workbench as a client to interact with our database in the cloud. We fired the below-mentioned queries and captured their response time.

4. Relational Database Design:

As we have already mentioned in the above section about the database being ready, it is a good idea to understand the relationship of the database objects in detail. This scenario consists of three database objects namely STOPS, STOPTIMES, and TRIPS.

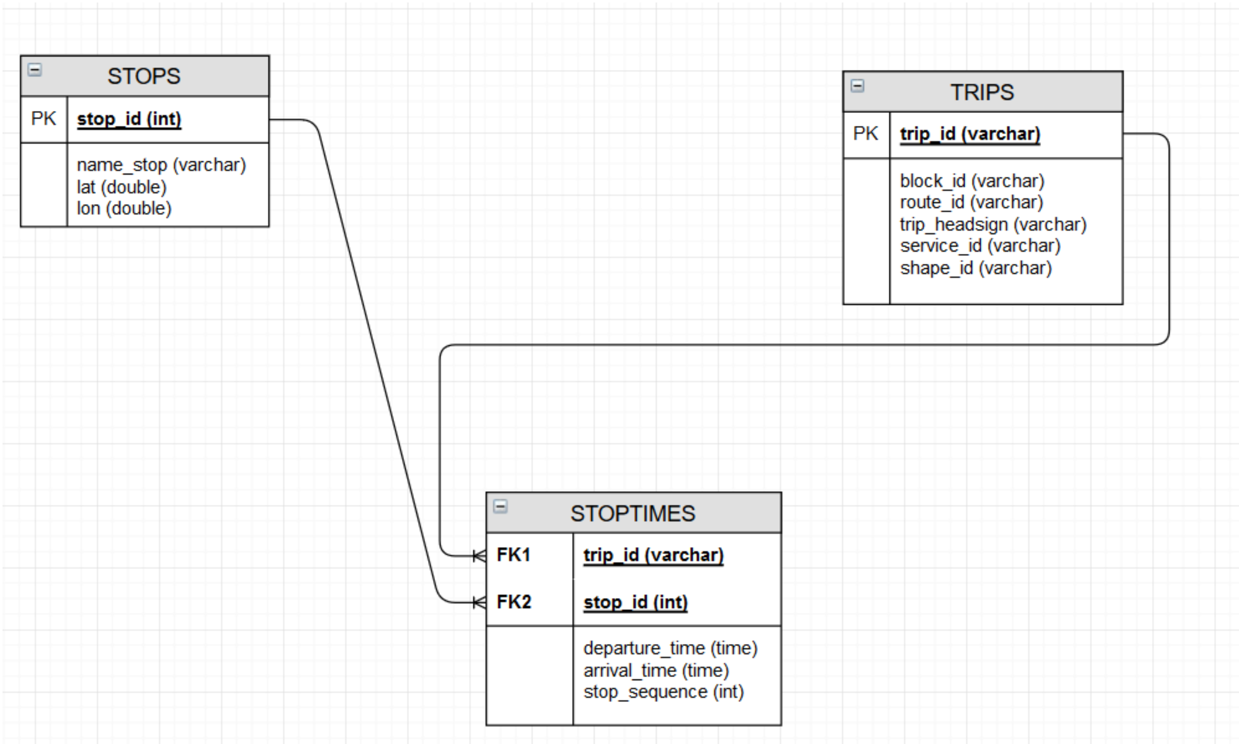


Figure: 3.1 ER Diagram

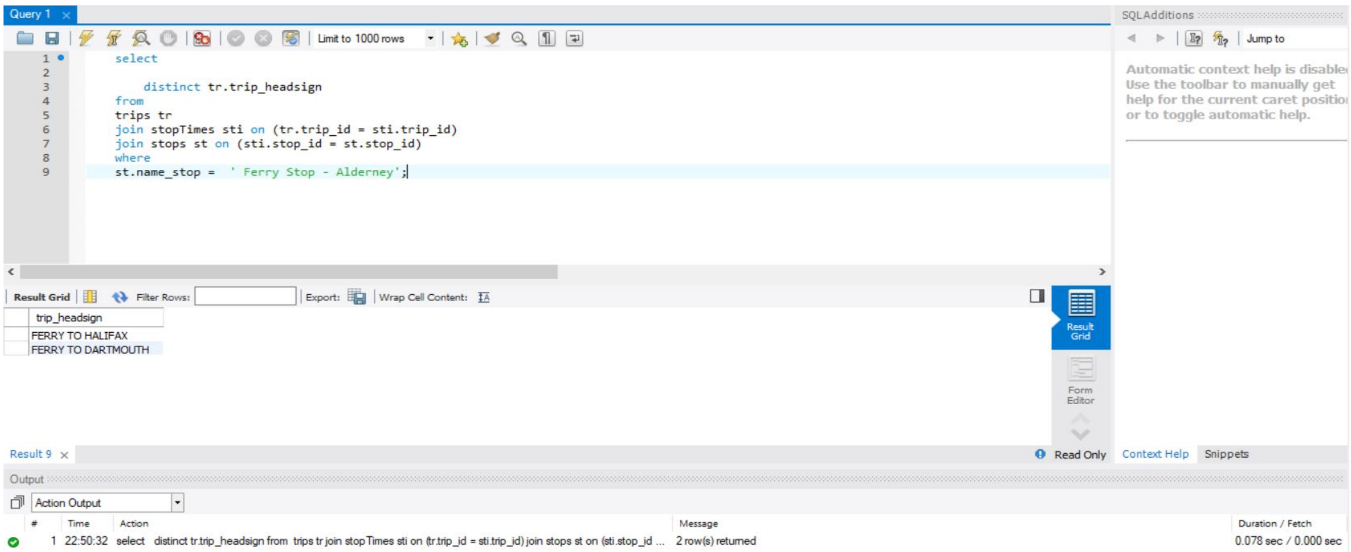
As from the Figure 3.1, it is clear that STOPS.stop_id is the primary key and is referenced to STOPTIMES.stop_id as a foreign key. Similarly, TRIPS.trip_id is a primary key which is referenced to STOPTIMES.trip_id. STOPTIMES.trip_id is a foreign key. It is important to note that both the columns with primary key have an index on them. The storage engine creates it by default.

5. Application Queries:

5.1. MySQL Queries:

a. Find all buses for a particular Bus Stop

1. Input: Bus Stop Name
2. Output: List of all buses, response time for the search query



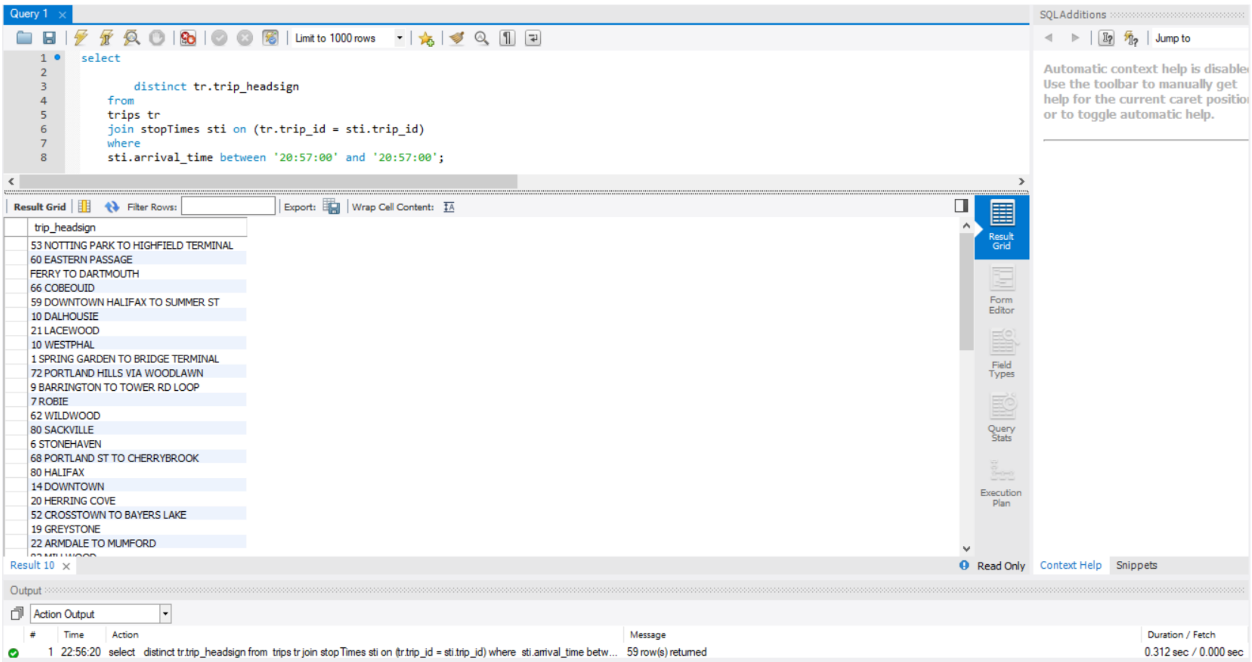
Ans:

1. Ferry to Dartmouth
2. Ferry to Halifax

Total time taken –78 ms

b. Find buses between two range

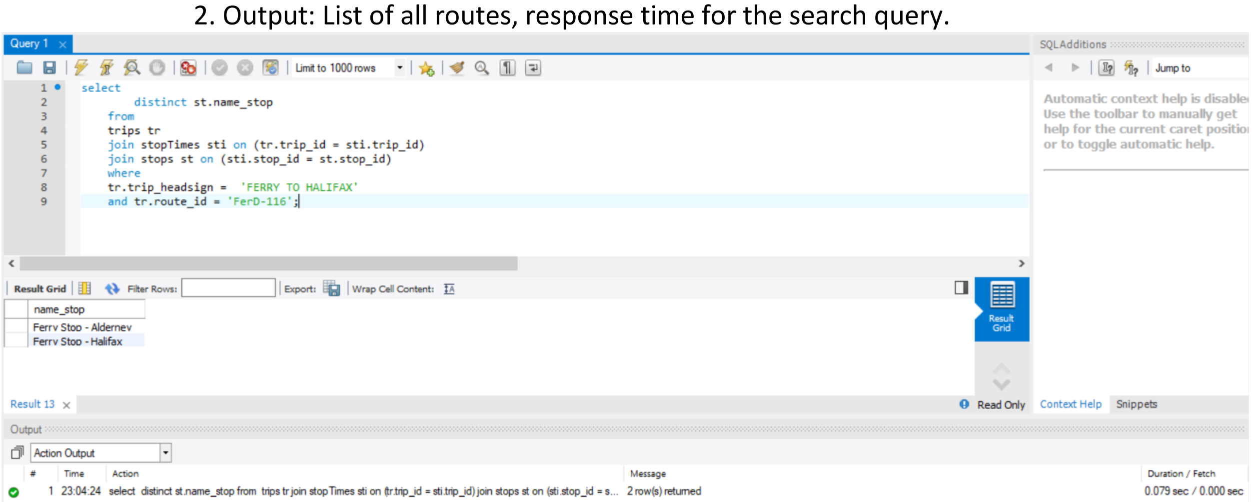
1. Input: Time Range 1 (hh:mm:ss), Time Range 2 (hh:mm:ss)
2. Output: List of all buses, response time for the search query



Ans:
Total time taken – 312 ms

c. Find route information of a particular bus on a particular route

1. Input: Bus Name, Route Name
2. Output: List of all routes, response time for the search query.

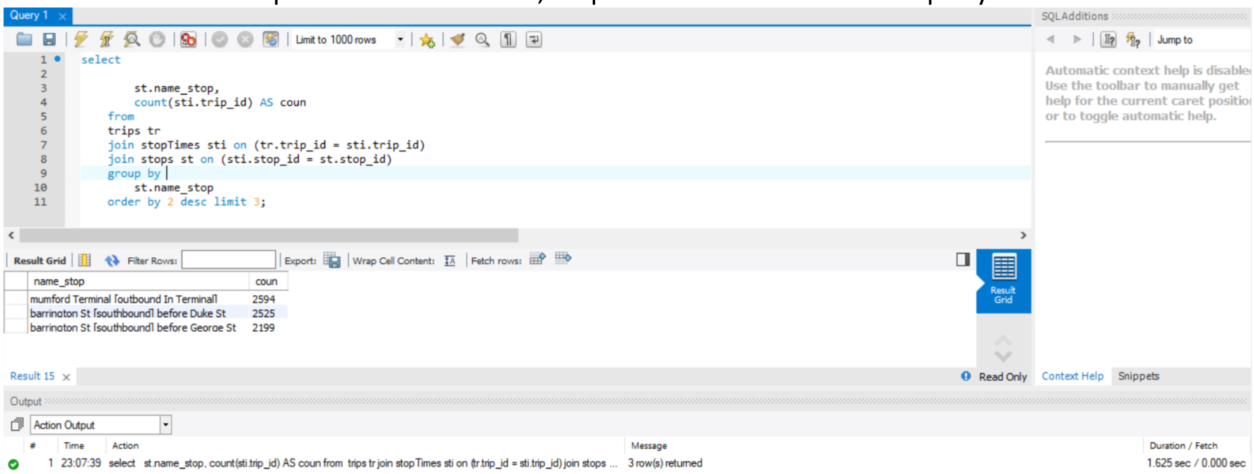


Ans:
1. Ferry to Dartmouth
2. Ferry to Halifax

Total time taken – 79 ms

d. Find top 3 bus stops that are the busiest throughout the day in terms of bus routes. (Hint: The bus stops with high volume of bus routes and close time gaps would be considered as busiest).

1. Input: None
2. Output: List of Bus Name, response time for the search query



Ans.

1. mumford Terminal [outbound In Terminal]
2. barrington St [southbound] before Duke St
3. barrington St [southbound] before George St

Total time taken – 1625 ms

5.2. Elastic Search:

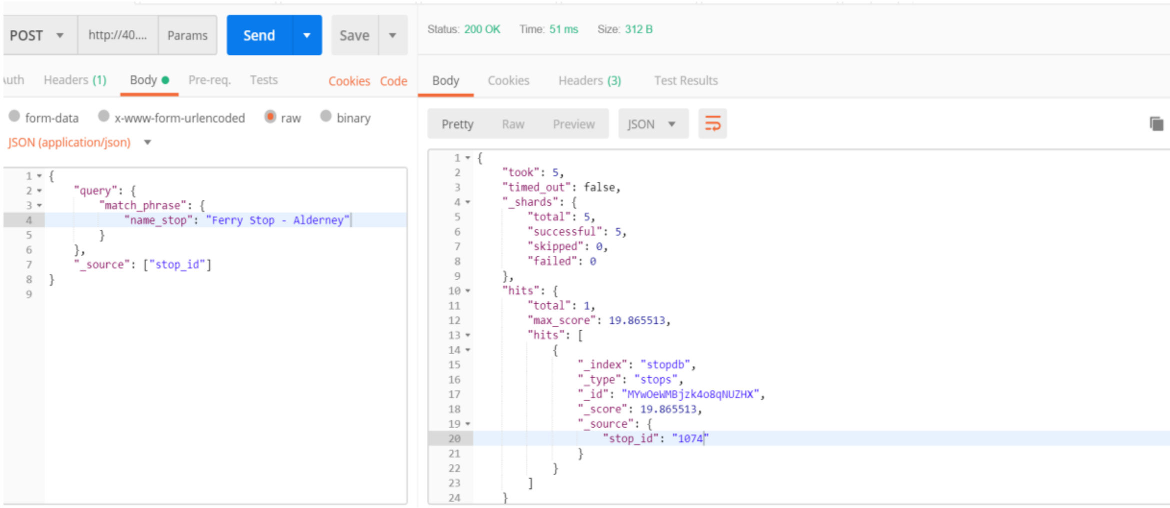
In our approach, we have created separate index and document for each schema (tripdb/trips, stopdb/stops, and stopttimesdb/stoptimes). We have also used multiple features of elastic search such as range, match, filters, etc for searching the document. The execution of the elastic search query is as follows:

The output of the first query is taken as input for the second query and is continued until we get the final output.

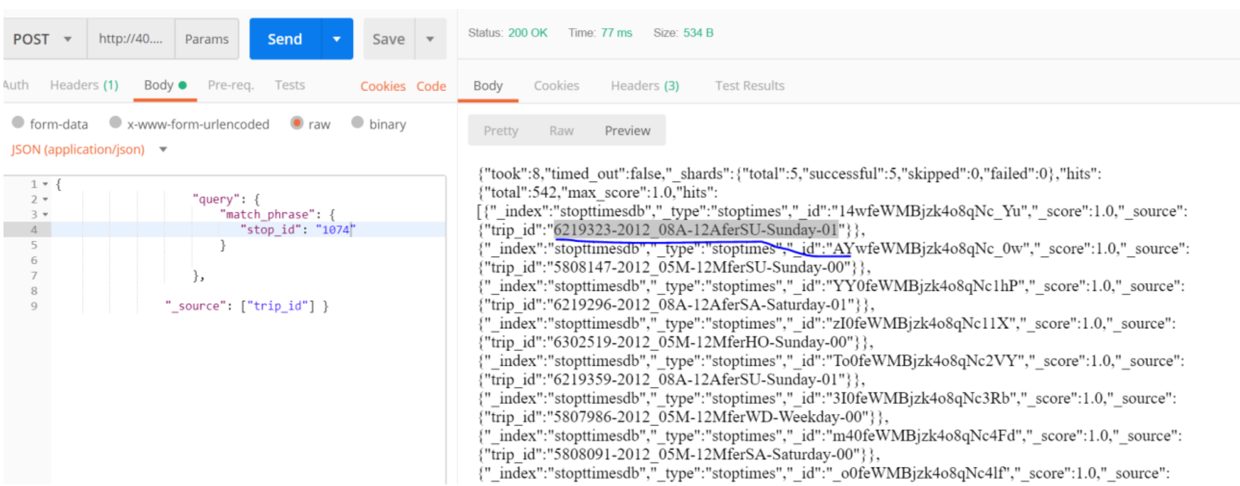
a. Find all buses for a particular Bus Stop

1. Input: Bus Stop Name
2. Output: List of all buses, response time for the search query

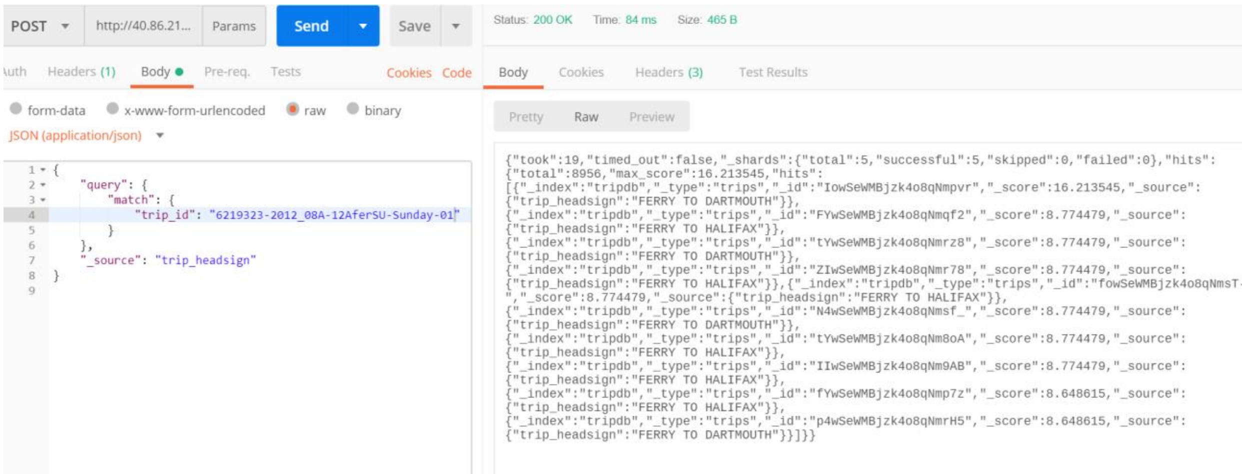
Query 1: POST http://40.86.219.24:9200/stopdb/stops/_search



Query 2: POST http://40.86.219.24:9200/stopttimesdb/stoptimes/_search



Query 3: POST http://40.86.219.24:9200/tripdb/trips/_search



Ans:

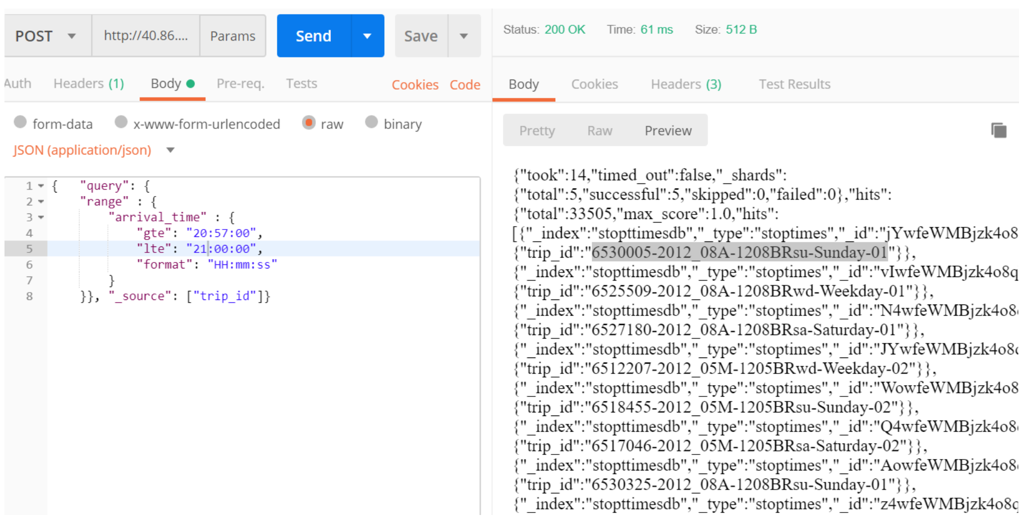
1. Ferry to Dartmouth
2. Ferry to Halifax

Total time taken – 212 ms.

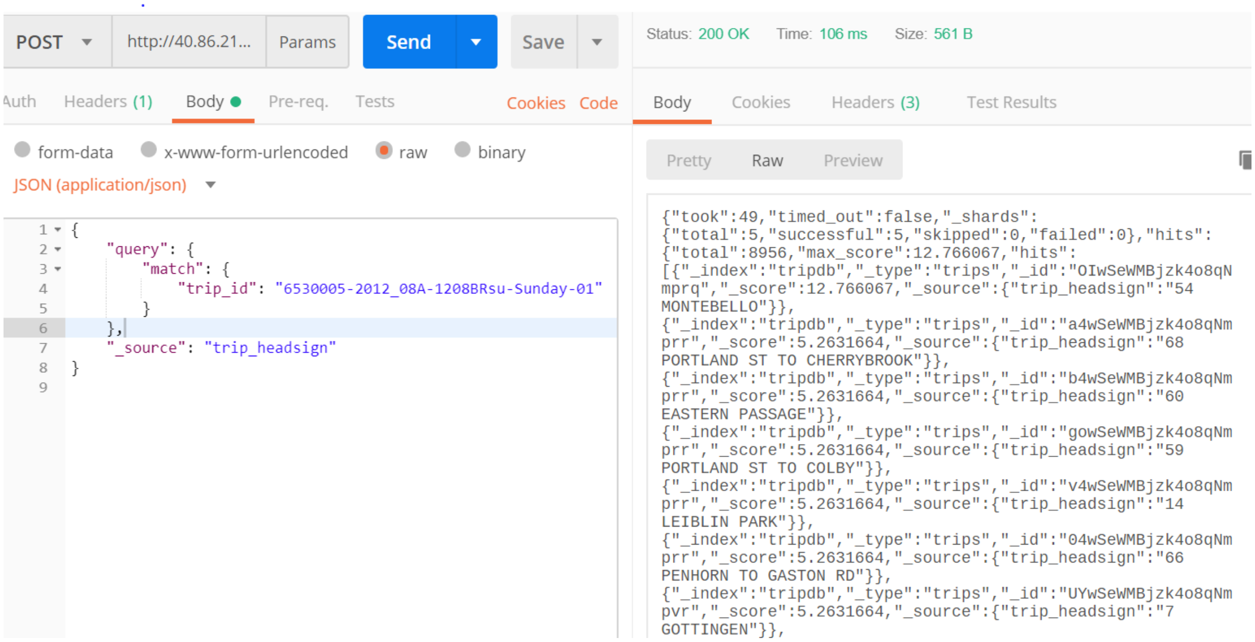
b. Find buses between two range

1. Input: Time Range 1 (hh:mm:ss), Time Range 2 (hh:mm:ss)
2. Output: List of all buses, response time for the search query

Query 1: POST http://40.86.219.24:9200/stopttimesdb/stoptimes/_search



Query 2: POST http://40.86.219.24:9200/tripdb/trips/_search

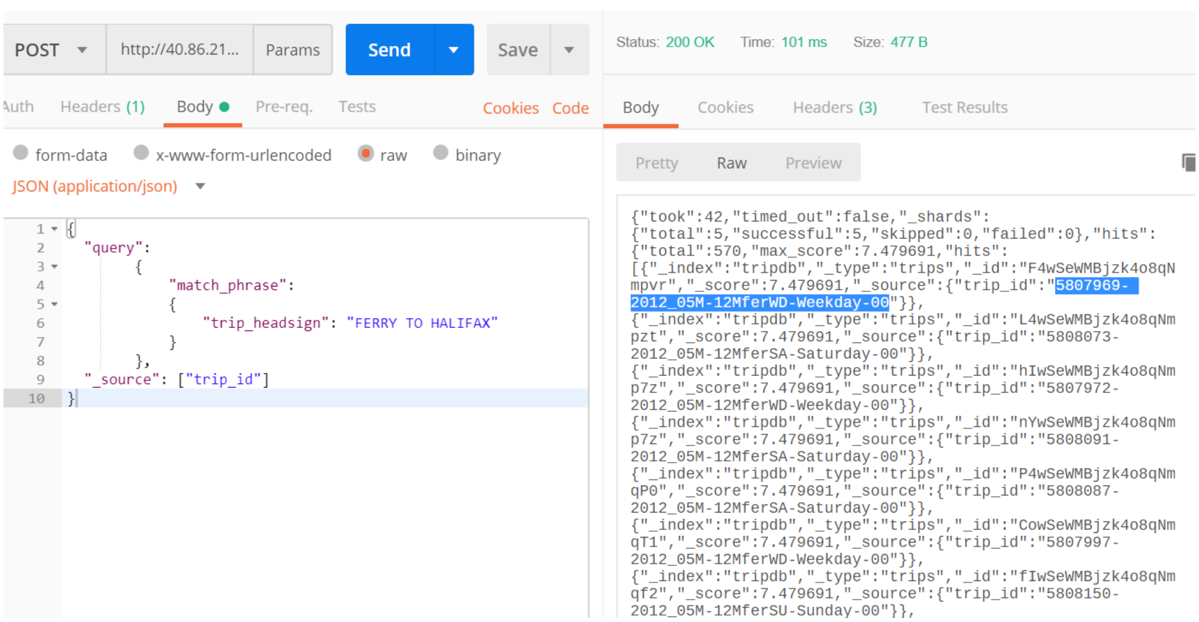


Ans.
Total time taken – 61+106 = 167 ms

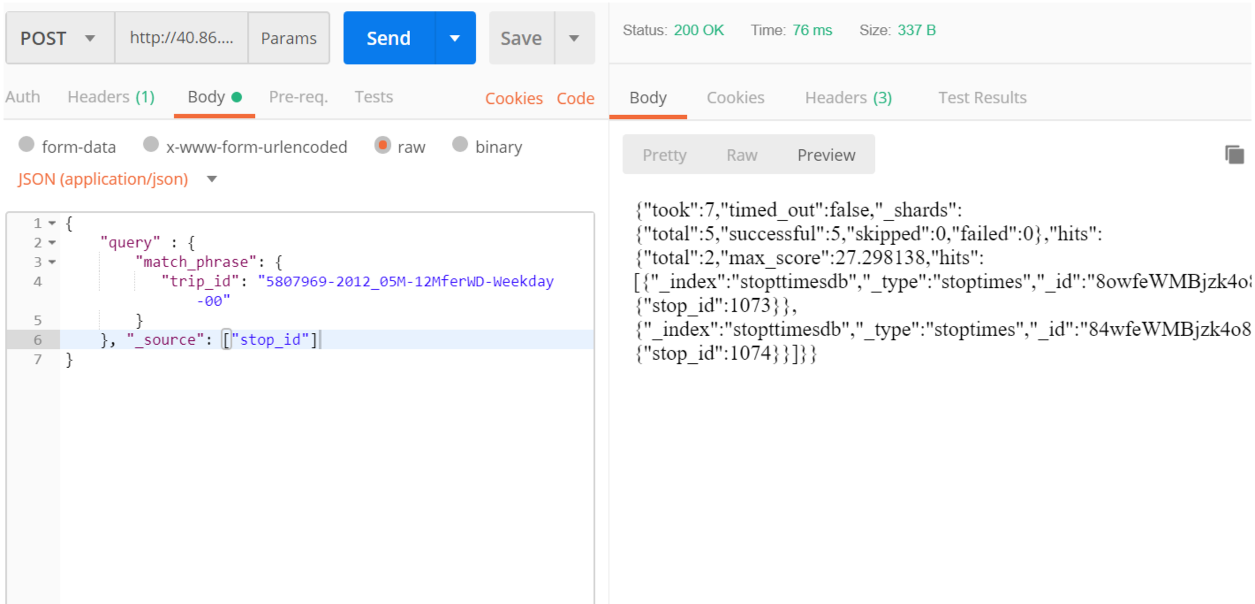
c. Find route information of a particular bus on a particular route

1. Input: Bus Name, Route Name
2. Output: List of all routes, response time for the search query.

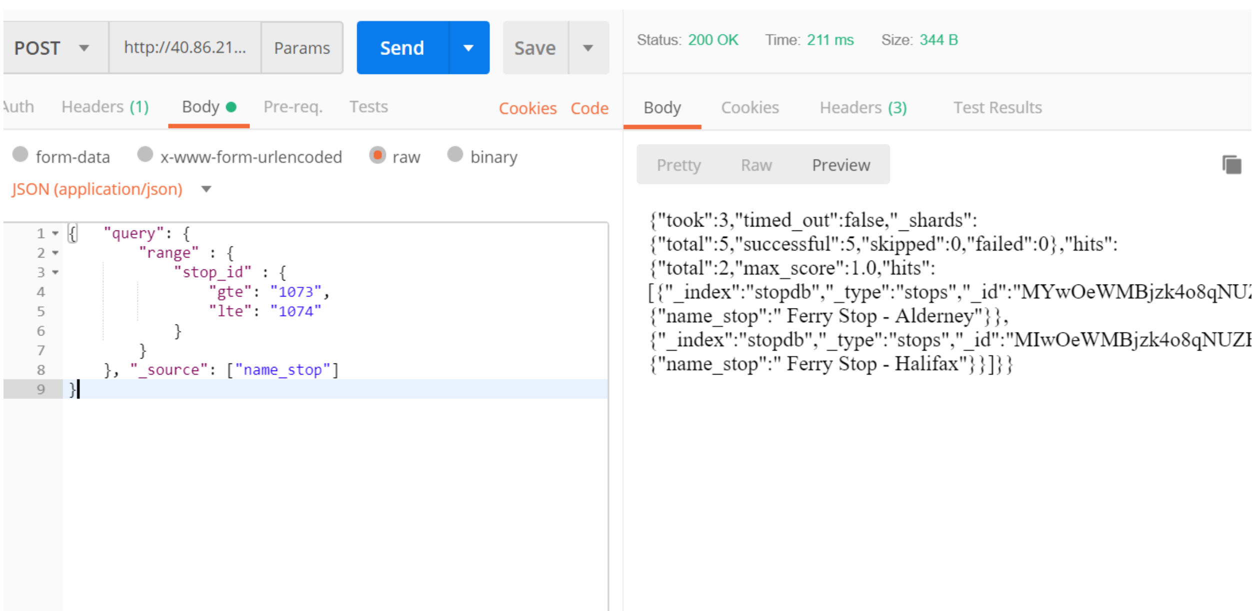
Query 1: POST http://40.86.219.24:9200/tripdb/trips/_search?q=route_id:FerD-116



Query 2: POST http://40.86.219.24:9200/stopttimesdb/stoptimes/_search



Query 3: POST http://40.86.219.24:9200/stopdb/stops/_search



Ans.

Stops for route id = FerD-116 and bus name = Ferry to Halifax are:

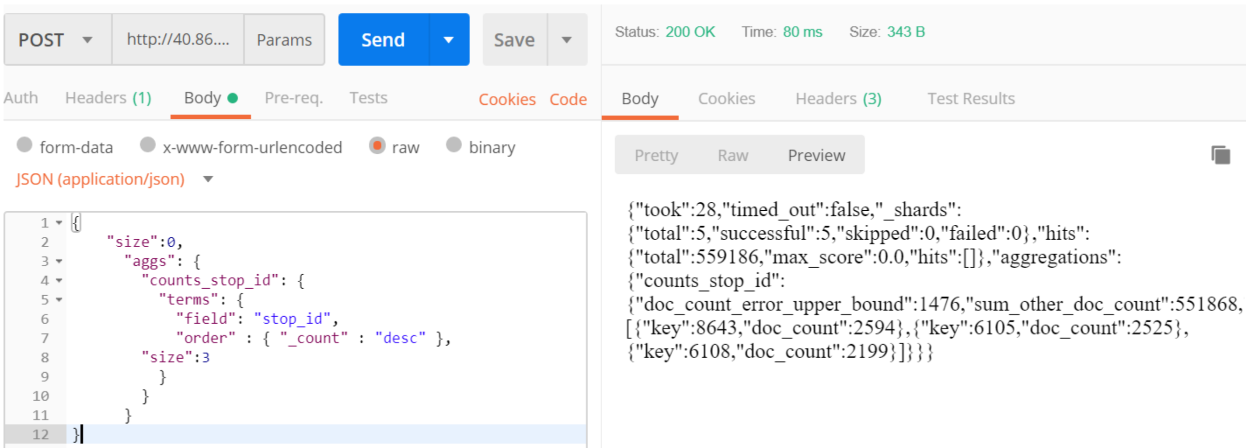
1. Ferry Stop – Alderney
2. Ferry Stop – Halifax

Total time taken – 101+76+211 = 388 ms

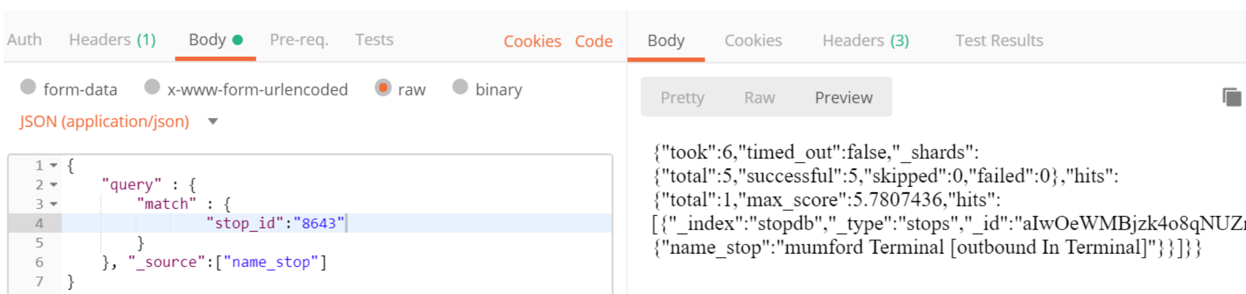
d. Find top 3 bus stops that are the busiest throughout the day in terms of bus routes. (Hint: The bus stops with high volume of bus routes and close time gaps would be considered as busiest).

1. Input: None
2. Output: List of Bus Name, response time for the search query

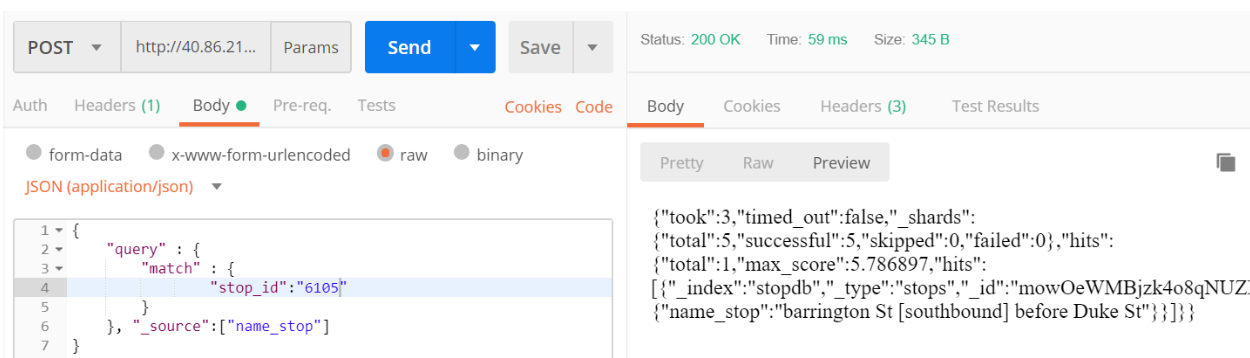
Query 1: POST http://40.86.219.24:9200/stopttimesdb/stoptimes/_search



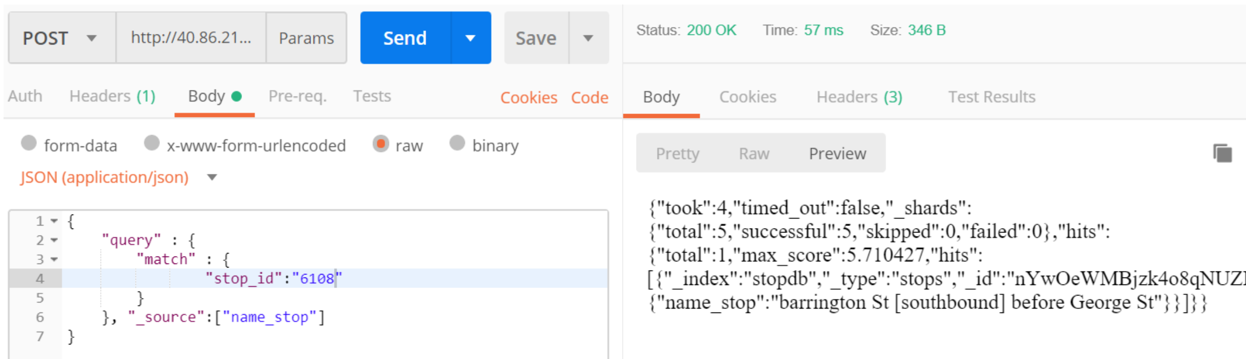
Query 2: POST http://40.86.219.24:9200/stopdb/stops/_search



Query 3: POST http://40.86.219.24:9200/stopdb/stops/_search



Query 4: POST http://40.86.219.24:9200/stopdb/stops/_search



Ans.
1. mumford Terminal [outbound In Terminal]
2. barrington St [southbound] before Duke St
3. barrington St [southbound] before George St
Total time taken – 80+59+60+57 = 256 ms.

6. Test results and summary:

As we observed two different databases with same datasets, it is a good idea to compare their performance with few metrics as given below.

6.1. Data extraction and manipulation:

While extracting the data, we can see that there is not a significant difference between the time taken by both the databases except for question “d”.

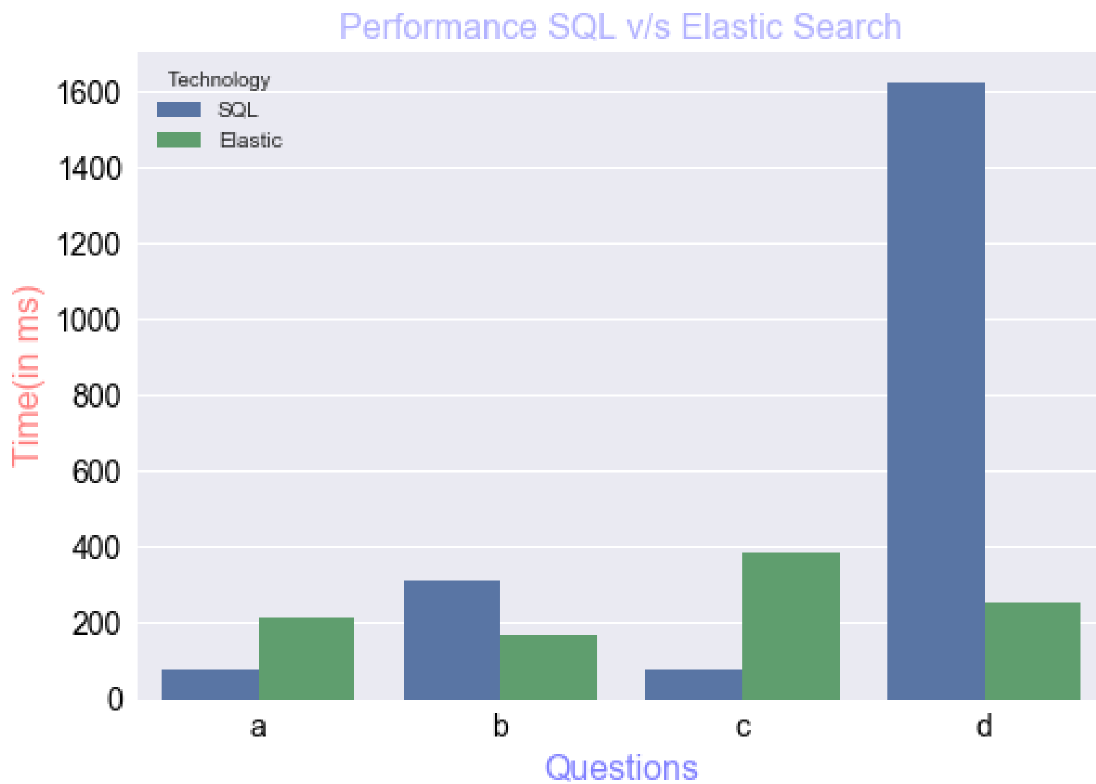


Figure: 6.1. SQL v/s Elastic Search

From the above figure, one can easily understand that we have Time (in milliseconds) on the Y-axis and all the questions are on the X-axis with blue colored bars representing SQL and green ones representing Elastic.

It is also a good idea to discuss another metric i.e., the time taken for manipulation. The data for MYSQL was inserted remotely from MYSQL workbench. For Elastic Search data was first converted into JSON format and then using REST client(POSTMAN) data was imported. Elastic Search took around twenty minutes to import the data while MYSQL took around thirty minutes to import it. But it is worth noting that both the databases are installed on different clouds with different hardware configurations. We have emphasized on this point later in the next part.

6.2. Cloud computing power:

MYSQL is based on RDBMS while Elastic Search is based on NoSQL database. The environment setup for MYSQL as well as for elastic search was easy. But performance-wise, the server was getting freeze due to the shortage of memory (RAM) on Amazon Web Service. While MYSQL was performing well on the same server. Due to this memory issue, we used another cloud service provider Azure to get better performance for elastic search. Elastic Search was then performing better at Azure. For the record, Amazon Web has one GB of RAM allocated per instance, while Azure has four GB. Hence, the Elastic search is not lightweight in terms of memory usage as compared to MYSQL.

7. Conclusion:

Based on the points emphasized in section 5. of this report we conclude that given the dataset, it is a fair choice of using traditional database over NoSQL database (Elastic Search). Even though non-traditional databases may be good at handling large data, but at the same time, it demands more computing resources. Given the resources provided (as we shifted to Microsoft Azure), we did not observe any strong improvement in data extraction process.