Yuan Zhuang
B00755386
January 21, 2017

Assignment 1

CSCI 5408

A. Task Description

City Buses , as the most usable and economical transportation tools, are keep playing important roles on people's daily life today. And in order to make the most of them, more detail information are needed for passengers, such as stops and routes of a particular bus, or the arrival and departure time of a particular bus in a particular stops, etc. So applications base on such datas are widely developed("transit" is a good example).

To get to know more about the related applications and how it works, I get data from the website (https://www.elastic.co/products/elasticsearch) and has created database (with three tables, literally stops, stoptimes and trips) to simulate the whole working process of such application.

I've created two databases, on local mysql and on elasticsearch that base on AWS separately, and then did 4 query task on each searching engine, and compared the performance in terms of their query time. By the comparison, we can get to know the difference between local based search and cloud based search, as well as the complexity of operation on the two searching engining.

The instance of database on mysql are showing below:

local: mysql

1. instances of table stops:

←T→	stop_id	name_stop	latitude	longitude
Edit Copy Delete	1002	Hwy 101 / Magazine Hill / Bedford Bypass	44.72640000	-63.64230000
Edit Copy Delete	1073	Ferry Stop - Halifax	44.64950000	-63.57290000
Edit Copy Delete	1074	Ferry Stop - Alderney	44.66490000	-63.57020000
Edit Copy Delete	1075	Ferry Stop - Woodside	44.64860000	-63.54720000
Edit Copy Delete	6000	akerley Blvd after Burnside Dr	44.71480000	-63.58980000
Edit Copy Delete	6001	akerley Blvd after civic address 95	44.71380000	-63.59370000
Edit Copy Delete	6002	akerley Blvd [westbound] after Joe Zatzman Ave	44.71140000	-63.60050000
Edit Copy Delete	6003	akerley Blvd [eastbound] after Joe Zatzman Ave	44.71200000	-63.59820000
Edit Copy Delete	6004	akerley Blvd after Williams Ave	44.71630000	-63.58570000
Edit Copy Delete	6005	akerley Blvd in front of Windmill Rd	44.70400000	-63.61110000

2. instances of table stoptimes:

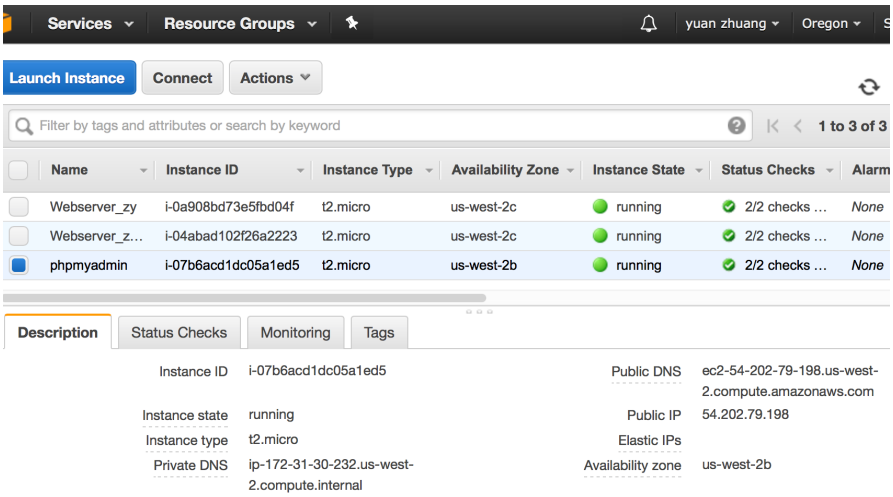
trip_id	stop_id	stop_sequence	arrival_time	departure_time
6513332-2012_05M-1205BRwd-Weekday-02	6940	1	13:00:00	13:00:00
6513332-2012_05M-1205BRwd-Weekday-02	9061	2	13:32:00	13:32:00
6528747-2012_08A-1208BRsa-Saturday-01	7615	1	20:55:00	20:55:00
6528747-2012_08A-1208BRsa-Saturday-01	8616	2	20:57:00	20:57:00
6528747-2012_08A-1208BRsa-Saturday-01	8612	3	20:57:00	20:57:00
6528747-2012_08A-1208BRsa-Saturday-01	8618	4	20:58:00	20:58:00
6528747-2012_08A-1208BRsa-Saturday-01	8615	5	20:58:00	20:58:00
6528747-2012_08A-1208BRsa-Saturday-01	8619	6	20:59:00	20:59:00
6528747-2012_08A-1208BRsa-Saturday-01	6019	7	20:59:00	20:59:00
6528747-2012_08A-1208BRsa-Saturday-01	6018	8	20:59:00	20:59:00
6528747-2012_08A-1208BRsa-Saturday-01	8418	9	20:59:00	20:59:00
6528747-2012_08A-1208BRsa-Saturday-01	8426	10	21:00:00	21:00:00
6528747-2012_08A-1208BRsa-Saturday-01	6930	11	21:01:00	21:01:00
6518384-2012_05M-1205BRsu-Sunday-02	8282	1	18:27:00	18:27:00
6518384-2012_05M-1205BRsu-Sunday-02	8281	2	18:27:00	18:27:00
6518384-2012_05M-1205BRsu-Sunday-02	7107	3	18:28:00	18:28:00
6518384-2012_05M-1205BRsu-Sunday-02	6331	4	18:28:00	18:28:00

3. instances of table trips:

←T→	trip_id	block_id	route_id	trip_headsign	service_id	shape_id
Edit Copy Delete	5807912-2012_05M-12MferWD-Weekday-00	a_1968470	FerD-116	FERRY TO HALIFAX	2012_05M-12MferWD-Weekday-00	FerD0004
Edit Copy Delete	5807913-2012_05M-12MferWD-Weekday-00	a_1968470	FerD-116	FERRY TO HALIFAX	2012_05M-12MferWD-Weekday-00	FerD0004
Edit Copy Delete	5807914-2012_05M-12MferWD-Weekday-00	a_1968471	FerD-116	FERRY TO HALIFAX	2012_05M-12MferWD-Weekday-00	FerD0004
Edit Copy Delete	5807918-2012_05M-12MferWD-Weekday-00	a_1968470	FerD-116	FERRY TO HALIFAX	2012_05M-12MferWD-Weekday-00	FerD0004
Edit Copy Delete	5807919-2012_05M-12MferWD-Weekday-00	a_1968471	FerD-116	FERRY TO HALIFAX	2012_05M-12MferWD-Weekday-00	FerD0004
Edit Copy Delete	5807920-2012_05M-12MferWD-Weekday-00	a_1968470	FerD-116	FERRY TO HALIFAX	2012_05M-12MferWD-Weekday-00	FerD0004
Edit Copy Delete	5807921-2012_05M-12MferWD-Weekday-00	a_1968471	FerD-116	FERRY TO HALIFAX	2012_05M-12MferWD-Weekday-00	FerD0004
Edit Copy Delete	5807922-2012_05M-12MferWD-Weekday-00	a_1968470	FerD-116	FERRY TO HALIFAX	2012_05M-12MferWD-Weekday-00	FerD0004
Edit Copy Delete	5807923-2012_05M-12MferWD-Weekday-00	a_1968471	FerD-116	FERRY TO HALIFAX	2012_05M-12MferWD-Weekday-00	FerD0004
Edit Copy Delete	5807924-2012_05M-12MferWD-Weekday-00	a_1968470	FerD-116	FERRY TO HALIFAX	2012_05M-12MferWD-Weekday-00	FerD0004
Edit Copy Delete	5807925-2012_05M-12MferWD-Weekday-00	a_1968470	FerD-116	FERRY TO HALIFAX	2012_05M-12MferWD-Weekday-00	FerD0004
Edit Copy Delete	5807926-2012_05M-12MferWD-Weekday-00	a_1968470	FerD-116	FERRY TO HALIFAX	2012_05M-12MferWD-Weekday-00	FerD0004

Cloud: AWS & Elasticsearch

Firstly, I've create a cloud hosting on Amazon Web Service in order to store my database and do elasticsearch:

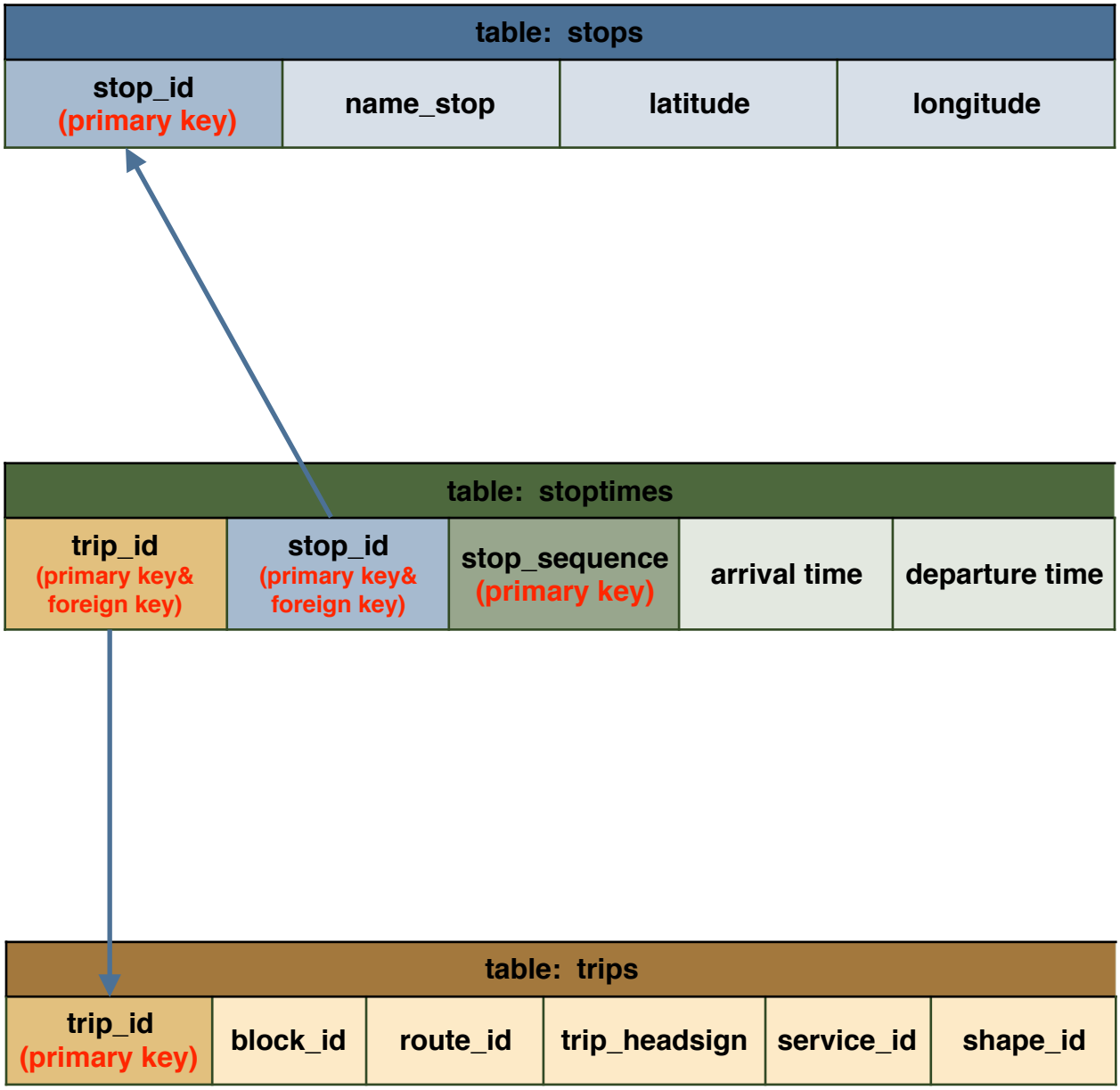


Then uploaded the same data that I've put in mysql, namely the three tables: stops,stoptimes and trips:

B. Relation Database Design:

For table stops and trips, primary keys are stop_id and trip_id respectively, and for table stoptimes, stop_sequence,stop_id and trip_id are combined as its primary key, besides stop_id and trip_id are also considered as foreign keys.

And the diagram are shown bellow:

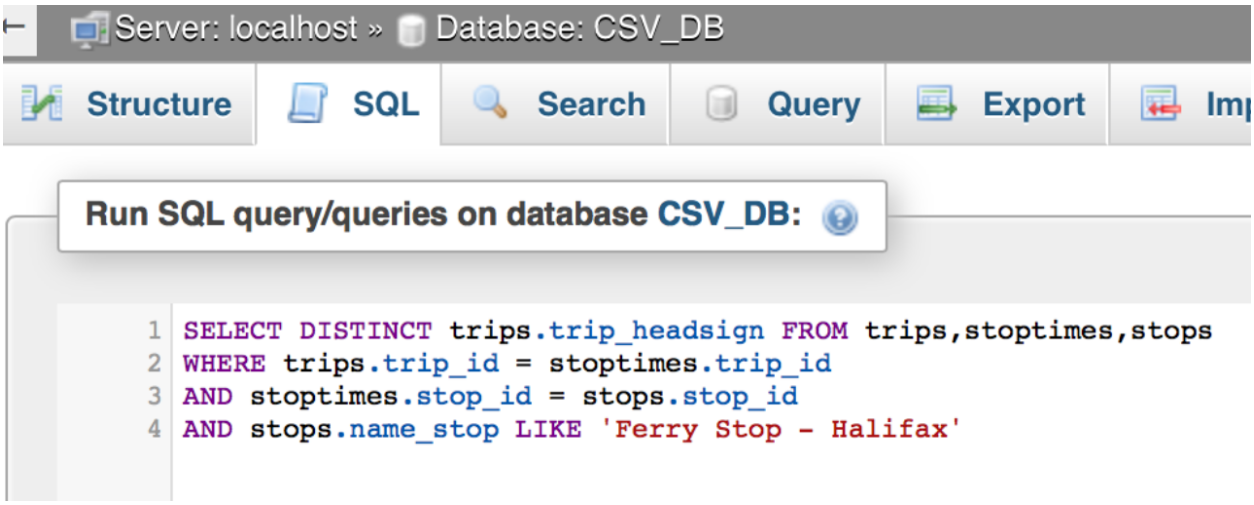


C&D. Application queries and Test Result

Mysql:local

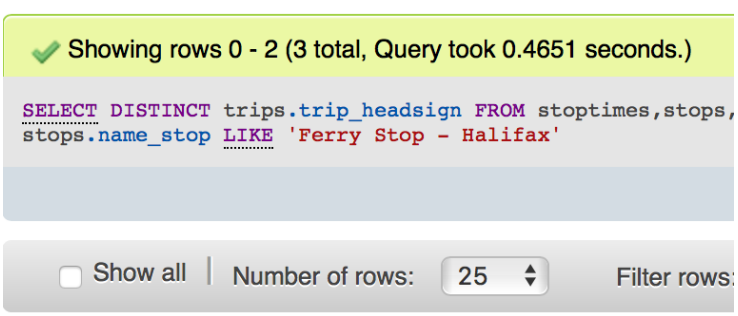
Task 1 find all buses for a particular Bus stop

1. input: Bus Stop Name:



Description: in order to find all bus for a chosen stop, as the UI application has the "join query" function, we can just simply join the three tables by two foreign keys in stoptimes(stop_id and trip_id respectively) then search the name_stop (in table stops) and get the trip_headsign list (in the table trips), namely the name of buses.

2.Output: List of all buses, response time for the search query



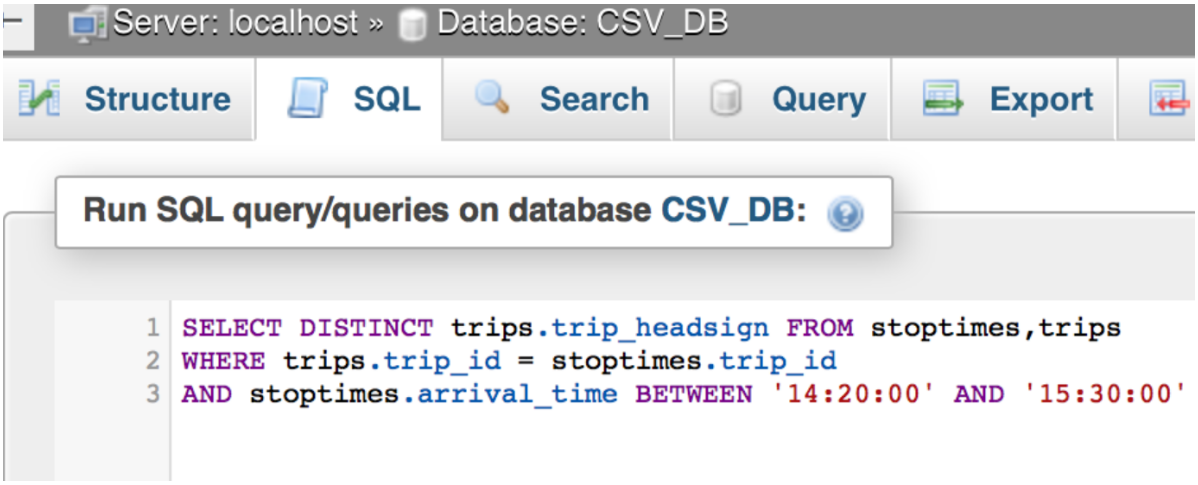
+ Options

trip_headsign
FERRY TO HALIFAX
FERRY TO WOODSIDE
FERRY TO DARTMOUTH

** the query time is 465.1 ms*

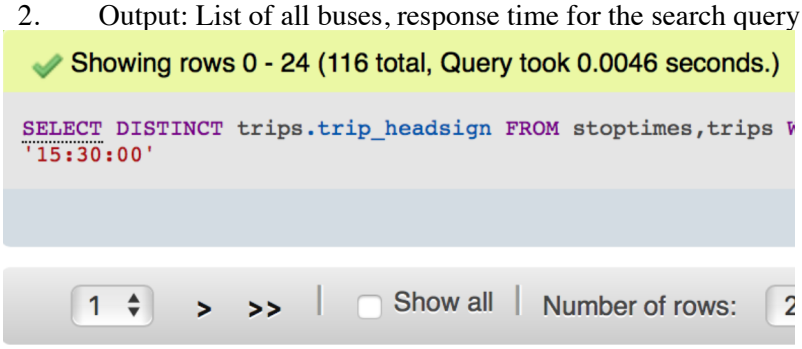
Task2 Find buses between two time ranges

1. Input: Time Range 1 (hh:mm:ss), Time Range 2 (hh:mm:ss)



Description: The task required us to find out all the buses that are arrived in stops during a chosen time gap, to achieve this goal, we can simply join two tables - trips and stoptimes that contains all the datas we need, then get the bus names list by search a time range(between … and …).

2. Output: List of all buses, response time for the search query



+ Options

trip_headsign
320 AIRPORT VIA FALL RIVER
54 MONTEBELLO
14 DOWNTOWN
320 DOWNTOWN HALIFAX VIA BRIDGE TERMINAL
51 WINDMILL TO BURNSIDE
83 SPRINGFIELD

* *the query time is 4.6ms*

* *Task3 Find route information of a particular bus on a particular route*

1. Input: Bus Name, Route Name

Server: localhost » Database: CSV_DB

Structure | SQL | Search | Query | Export | Import | Operati

Run SQL query/queries on database CSV_DB:

```
SELECT DISTINCT stops.name_stop,stoptimes.stop_sequence,stoptimes.trip_id AS Trip
FROM stops,stoptimes
WHERE stops.stop_id = stoptimes.stop_id
AND stoptimes.trip_id IN
    (SELECT DISTINCT trips.trip_id FROM trips
     WHERE trips.route_id LIKE '53-121'
     AND trips.trip_headsign LIKE '53 NOTTING PARK TO HIGHFIELD TERMINAL')
```

Description: This task is a requirement about searching information of routes for particular bus , it means that we need to search for trip_id in table trips first in terms of a route and a bus name, then combine the two table stops and stoptimes by key "stop_id", finally get all the stops and is sequence of trips on that route.

2. Output: List of all routes, response time for the search query

Showing rows 0 - 24 (1086 total, Query took 0.0120 seconds.)

```
SELECT DISTINCT stops.name_stop,stoptimes.stop_sequence,stoptimes.trip_id AS Trip FROM stops,s
stops.stop_id = stoptimes.stop_id AND stoptimes.trip_id IN (SELECT DISTINCT trips.trip_id FROM
trips.route_id LIKE '53-121' AND trips.trip_headsign LIKE '53 NOTTING PARK TO HIGHFIELD TERMIN
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create P

1 > >> | Number of rows: 25 Filter rows: Search this table

+ Options

name_stop	stop_sequence	Trip
bridge Terminal [to Dart] and [lanes 1 & 2]	1	6511464-2012_05M-1205BRwd-Weekday-02
wyse Rd before Boland Rd	2	6511464-2012_05M-1205BRwd-Weekday-02
wyse Rd after Boland Rd	3	6511464-2012_05M-1205BRwd-Weekday-02
wyse Rd in front of civic address 200	4	6511464-2012_05M-1205BRwd-Weekday-02
wyse Rd in front of civic address 254	5	6511464-2012_05M-1205BRwd-Weekday-02
wyse Rd before Albro Lake Rd	6	6511464-2012_05M-1205BRwd-Weekday-02
albro Lake Rd after Richmond St	7	6511464-2012_05M-1205BRwd-Weekday-02
albro Lake Rd after Chapman St	8	6511464-2012_05M-1205BRwd-Weekday-02
victoria Rd after Albro Lake Rd	9	6511464-2012_05M-1205BRwd-Weekday-02
victoria Rd before Primrose St	10	6511464-2012_05M-1205BRwd-Weekday-02
highfield Terminal [southbound]	11	6511464-2012_05M-1205BRwd-Weekday-02
bridge Terminal [to Dart] and [lanes 1 & 2]	1	6518225-2012_05M-1205BRsu-Sunday-02
wyse Rd before Boland Rd	2	6518225-2012_05M-1205BRsu-Sunday-02

** the query time is 12ms*

Task4 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

Server: localhost » Database: CSV_DB

Structure | SQL | Search | Query | Export | Import

Run SQL query/queries on database CSV_DB:

```
SELECT stops.name_stop, COUNT(*) AS COUNT_ROUTE FROM stops,stoptimes,trips
WHERE stops.stop_id = stoptimes.stop_id
AND stoptimes.trip_id = trips.trip_id
GROUP BY stops.name_stop
ORDER BY COUNT_ROUTE DESC
LIMIT 3
```

Description: To find out the top 3 busiest stop, we only need to count the volumes of buses with the same id, and get stop_id of top 3 counts, then find out the stop's name in terms of the stop_id in table stops.

2. Output: List of Bus Name, response time for the search query

Showing rows 0 - 2 (3 total, Query took 1.1457 seconds.)

```
SELECT stops.name_stop, COUNT(*) AS COUNT_ROUTE FROM stops,sto
stoptimes.stop_id AND stoptimes.trip_id = trips.trip_id GROUP
LIMIT 3
```

Profiling [Edit inlin

+ Options

name_stop	COUNT_ROUTE 1
mumford Terminal [outbound In Terminal]	2594
barrington St [southbound] before Duke St	2525
barrington St [southbound] before George St	2199

** the query time is 1145.7ms*

Cloud: AWS & Elasticsearch

comments: For the reason that Elasticsearch cannot join tables and do combined search, I write four python code file by write and read file function combined all the os command together and implement in terminal.

Take task one as an example:

- I firstly use os.system run the **curl** command in terminal to search for a stop name in table stops,"Ferry" for instance.
- Then write all the out put message in a txt file named "result_in_stops",.
- Then read and parse the txt in "result_in_stops", and get all the "stop_id" as a list and use it in the following search.
- Finally, I write down the final result in a txt file "result_for_a".

**all the files and resource code are attached.*

Task 1 find all buses for a particular Bus stop

The requirement of this task has already explained before, and to do it by elasticsearch ,firstly, I need to get stop_id for a stop name in the table stops, then get the relative trip_id in table stoptimes, finally search for the trip_headsign in table trips in terms of the trip_id in table trips:

Implement processing:

```
task_a — -bash — 80×12
Last login: Tue Jan 24 12:31:59 on ttys000
T9E68:~ zhuangyuan$ cd ~/Desktop/A1_elasticsearch/task_a
T9E68:task_a zhuangyuan$ python3 a_find_all_bus.py
Please input the name of the stop: Ferry Stop - Halifax
```

result:

```
FERRY TO WOODSIDE
FERRY TO HALIFAX
FERRY TO DARTMOUTH


**************************************
the query time is : 48ms.
**************************************
```

* *I've searched the same key word, and get the query time: 48ms*

Task2 Find buses between two time ranges

To require this task in elasticsearch, firstly to search for stop_id for a particular time range in the table stoptimes, then get the stop name list in table stops by the stop_id:

Implement processing:

```
task_b — -bash — 80×15
T9E68:task_b zhuangyuan$ python3 b_find_bus_in_range.py
Please input range of time(), separated by commas : 14:20:00,15:30:00
```

result:

```
19 GREYSTONE
53 NOTTING PARK TO BRIDGE TERMINAL
17 LACEWOOD
61 PORTLAND ST TO AUBURN- N. PRESTON
52 CROSSTOWN TO LACEWOOD
80 HALIFAX
60 EASTERN PASSAGE
68 PORTLAND ST TO CHERRYBROOK
57 PORTLAND HILLS VIA PORTLAND ESTATES
16 LACEWOOD VIA PARKLAND
8 WATERFRONT TO PIER 21
68 PORTLAND ST TO BRIDGE TERMINAL
81 DOWNTOWN
72 DARTMOUTH CROSSING VIA WRIGHT AVE
17 SAINT MARY'S
53 NOTTING PARK TO HIGHFIELD TERMINAL
2 WEDGEWOOD VIA MAIN
9 BARRINGTON TO MUMFORD
83 SPRINGFIELD
56 PORTLAND HILLS
1 SPRING GARDEN TO BRIDGE TERMINAL
8 WATERFRONT TO SCOTIA SQUARE
10 WESTPHAL
4 DOWNTOWN VIA NORTH
185 HALIFAX


****************************************
the query time is : 21ms.
****************************************
```

* *I've searched the same key word, and get the query time: 21ms*

* ***<u>Task3 Find route information of a particular bus on a particular route</u>***

In terms of this task, firstly, to get the trip_id by search for bus name and the route_in in table trips, then get the stop_id and sequence_id of each trips in table stoptimes, finally get stops name by stop_id in table stops:

Implement processing:

```
task_c — -bash — 80×29
T9E68:task_c zhuangyuan$ python3 c_find_route_info.py
Please input the route id : 53-121
Please input the bus name : 53 NOTTING PARK TO HIGHFIELD TERMINAL
```

result:

```
result_for_c
6518231-2012_05M-1205BRsu-Sunday-02: wyse Rd after Boland Rd  3
6518231-2012_05M-1205BRsu-Sunday-02: albro Lake Rd after Chapman St  8
6518231-2012_05M-1205BRsu-Sunday-02: bridge Terminal [to Dart] and [lanes 1 & 2]  1
6518231-2012_05M-1205BRsu-Sunday-02: wyse Rd in front of civic address 200  4
6518231-2012_05M-1205BRsu-Sunday-02: wyse Rd in front of civic address 254  5
6518231-2012_05M-1205BRsu-Sunday-02: wyse Rd before Albro Lake Rd  6
6518236-2012_05M-1205BRsu-Sunday-02: wyse Rd before Boland Rd  2
6518236-2012_05M-1205BRsu-Sunday-02: highfield Terminal [southbound]  11
6518236-2012_05M-1205BRsu-Sunday-02: wyse Rd in front of civic address 200  4
6518236-2012_05M-1205BRsu-Sunday-02: wyse Rd in front of civic address 254  5
6518236-2012_05M-1205BRsu-Sunday-02: wyse Rd before Albro Lake Rd  6
6518236-2012_05M-1205BRsu-Sunday-02: bridge Terminal [to Dart] and [lanes 1 & 2]  1
6518236-2012_05M-1205BRsu-Sunday-02: albro Lake Rd after Richmond St  7
6518236-2012_05M-1205BRsu-Sunday-02: victoria Rd before Primrose St  10
6518236-2012_05M-1205BRsu-Sunday-02: wyse Rd after Boland Rd  3
6518236-2012_05M-1205BRsu-Sunday-02: albro Lake Rd after Chapman St  8

|

*****************************************************************************
the query time is : 130ms.
*****************************************************************************
```

* *<u>I've searched the same key word, and get the query time: 130ms</u>*

<u>Task4 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).</u>

This one is to count the time that each stop_id has shown up in table stoptimes, and get the top 3 id, then find there names in table stops:

implement processing:

```
task_d — -bash — 80×15
T9E68:task_c zhuangyuan$ cd ~/Desktop/A1_elasticsearch/task_d
T9E68:task_d zhuangyuan$ python3 d_find_busy_bus.py
```

result:

```
the top 3 busiest bus is:
barrington St [southbound] before Duke St
barrington St [southbound] before George St
mumford Terminal [outbound In Terminal]


**************************************************
the query time is : 49ms.
**************************************************
```

E. Summary

a. performance:

	task 1	task 2	task 3	task 4
local(mysql)	465.1	4.6	12	1145.7
cloud(elasticsearch)	48	21	130	49

a. we can find the when tasks need to join more then three tables or need to count in table with large volume(stoptimes), the cloud search has a better performance, otherwise, mysql can do better, so I may assume that elasticsearch on AWS can perform better when database was quite large.

b. For the reason that the cloud cannot do join search so that user must do it step by step, it may took more time for user on operating than mysql and the complex steps may also lead to wrong manipulation as well.

c. Mysql has a better UI view, the query result are shown more readable for users, but the result shown in sense (UI application of elasticsearch) has many unnecessary message in it, besides, it cannot show the whole query result, for example, the total of hits of result may be 90 or more,but only 10 are shown.

So, personally speaking, as the volume of dataset given is not that large, I would say that mysql has a better performance.