CSCI 5408, Summer 2018

**Assignment 1: Search Query Implementation using Relational Database and Elastic Search**

Due: May 24, 2018, 11:59PM

Andrea Christians (B00412494) and Navkaran Kumar (B00782012))

Dalhousie University

Contents

[Business Scenario 4](#_Toc514905578)

[Data Analysing and Normalising 4](#_Toc514905579)

[Relational Database Management System (RDBMS) 4](#_Toc514905580)

[SQL Scripts: 5](#_Toc514905581)

[Testing: 6](#_Toc514905582)

[Elastic Search 6](#_Toc514905583)

[Relational Database 6](#_Toc514905584)

[Entitiy Relationship Diagram 6](#_Toc514905585)

[Application Queries 7](#_Toc514905586)

[MySQL queries 7](#_Toc514905587)

[Elastic Search Queries 7](#_Toc514905588)

[Test Results 7](#_Toc514905589)

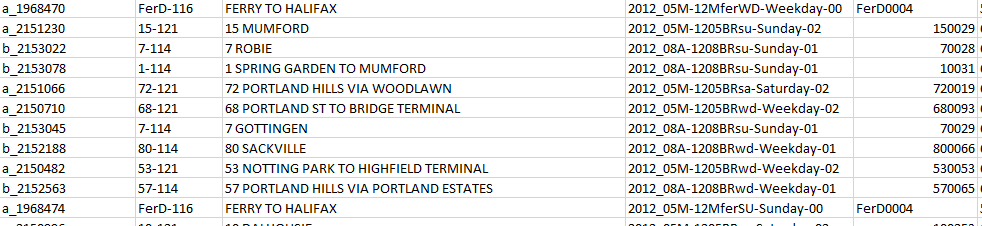
[MySQL 7](#_Toc514905590)

# Business Scenario

# Data Analysing and Normalising

## Relational Database Management System (RDBMS)

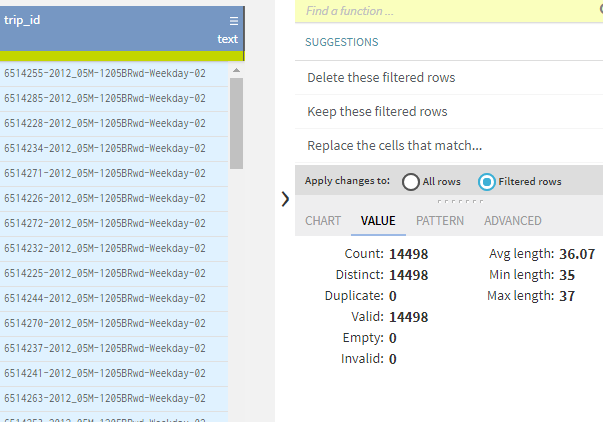
1. For all the transit routes involving *Ferry*, there was a difference in the way *route\_id* and *shape\_id* was represented in the trips.csv.



**Figure X:**

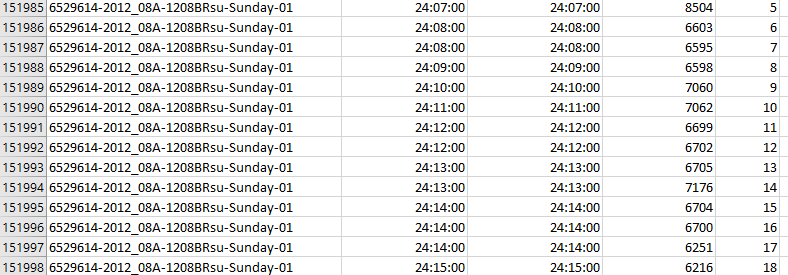
While importing data into RDBMS, *shape\_id* was allocated with a datatype of *INTEGER*, which led to slow data import times and data truncation.Additionally,since the frequency of entries were high, it still appeared to be valid data. Finally, while importing the data from the *trips* table, *shape\_id* was given the datatype of *varchar()* to overcome this scenario.

1. While creating relations among tables in MySQL, the length of the column needed to be specified in order to make it a primary or foreign key. In order to determine the lengths of data entries in *trip\_id*, Talend Data Preparation Software was used to display the tables and it was found that they did not exceed 50 characters in length. Thus, the datatype was changed for *trip\_id* was changed to *varchar (50)* in both the trips and stoptimes table using MySQL Workbench.



**Figure X:**

1. Finally, the *stoptimes* table had arrival and departure time that exceeded 24 hours and were not compliant with the 24-hour format.



**Figure X:**

The values were replaced and corrected with actual times using a SQL script. Before running the script, the data was imported with a text data type and then it was converted into a ‘*TIME’* datatype and there was no data truncation. The script was run which updated the values of the arrival and departure time in the *stoptimes* table (see below). For example, 25:02:50 would be changed to 01:02:50 (that is 1 AM).

### SQL Scripts:

**The following scripts were used to configure the database:**

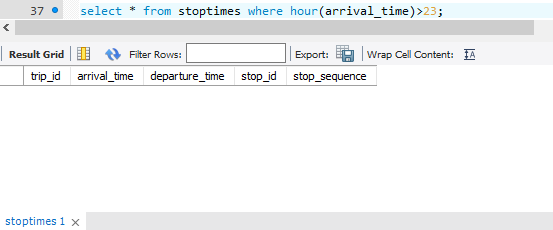
* update stoptimes set arrival\_time=hour(arrival\_time)-23

where hour(arrival\_time)>23;

* update stoptimes set departure\_time=hour(departure\_time)-23

where hour(departure\_time)>23;

### Testing:

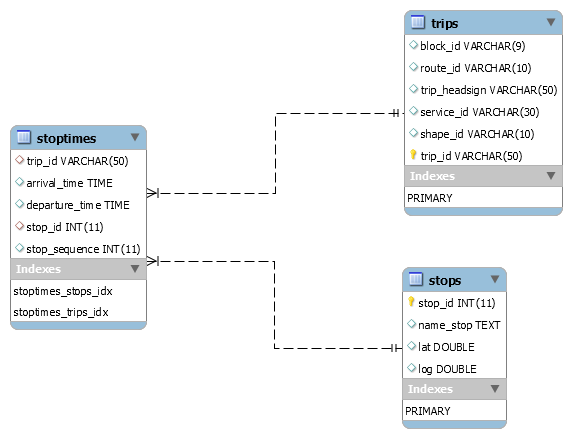


## Elastic Search

put in the github link for ruby script. And explain about it.

# Relational Database

## Entity Relationship Diagram



**Figure X:** An ERD diagram depicting the joins and schema.

# Application Queries

## MySQL queries

1. **Query:** select distinct(trip\_headsign) from trips

join stoptimes s on trips.trip\_id = s.trip\_id

join stops s2 on s.stop\_id = s2.stop\_id

where s2.stop\_id =

(select stop\_id from stops where name\_stop = "south Park St [northbound] after South St");

1. **Query:** select distinct(trip\_headsign) from trips

join stoptimes s on trips.trip\_id = s.trip\_id

where s.arrival\_time between "18:00:00" and "22:00:00";

1. **Query:** select trip\_id,name\_stop,route\_id,trip\_headsign,arrival\_time,departure\_time,

stop\_sequence,lat,log from trips

join stoptimes using(trip\_id)

join stops using(stop\_id)

where trip\_headsign="330 HALIFAX" and route\_id="330-114";

1. **Query:** select count(route\_id) as BD,name\_stop from trips

join stoptimes using(trip\_id)

join stops using(stop\_id)

group by stop\_id

order by BD desc limit 3;

## 

## Elastic Search Queries

1. **Query:**

{

"query": {

"has\_child": {

"type": "stoptimes",

"query": {

"bool": {

"must": [

{

"match": {

"name\_stop": "south Park St [northbound] after South St"

}

}

]

}

}

}

},

"size":0,

"aggregations": {

"distinct": {

"terms": {

"field": "trip\_headsign"

}

}

}

}

1. **Query:**

{

"query": {

"has\_child": {

"type": "stoptimes",

"query": {

"range" : {

"arrival\_time" : {

"gte": "18:00:00",

"lte": "22:00:00",

"format": "HH:mm:ss"

}

}

}

}

},

"size":0,

"aggregations": {

"distinct": {

"terms": {

"field": "trip\_headsign"

}

}

}

}

**C:**

{

"query": {

"has\_parent": {

"parent\_type": "trips",

"query": {

"bool": {

"must": [

{

"match": {

"trip\_headsign": "330 HALIFAX"

}

},

{

"match": {

"route\_id": "330-114"

}

}

]

}

}

}

}

}

**D:**

{

"size": 0,

"aggs": {

"group\_by\_state": {

"terms": {

"field": "stop\_id",

"size" : 3

},

"aggregations": {

"top\_three\_busy\_stops": {

"top\_hits": {

"\_source": {

"includes": [ "name\_stop" ]

},

"size" : 1

}

}

}

}

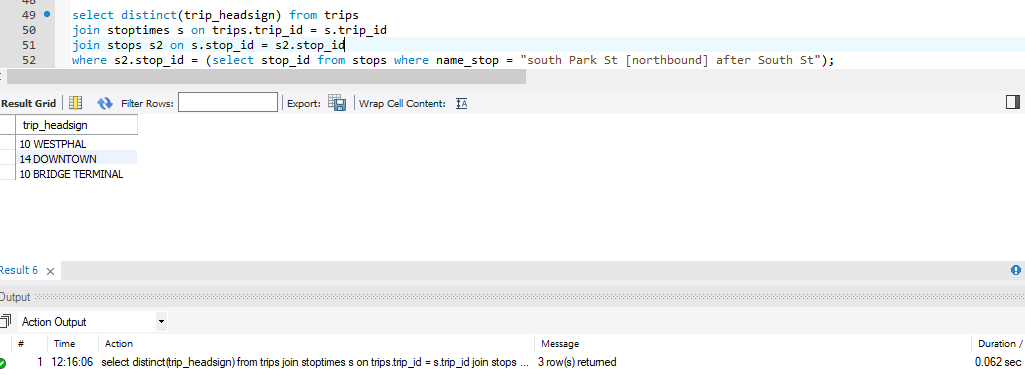
}

}

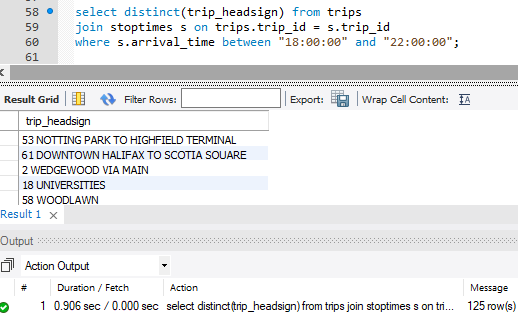
# Test Results

## MySQL

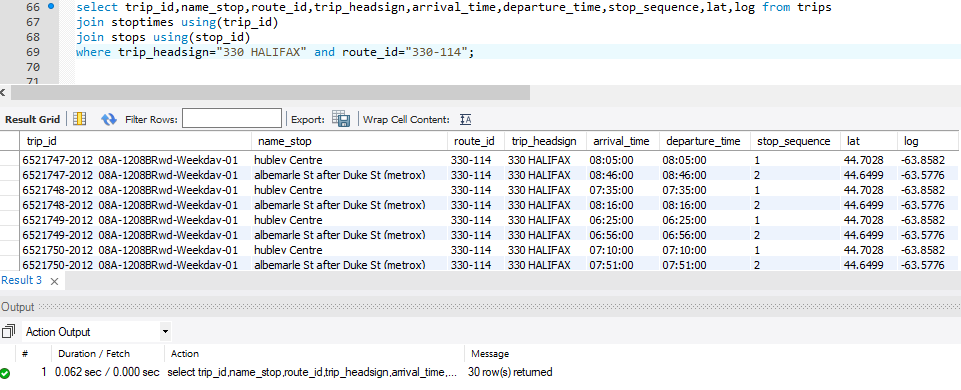
1. This query stated to print out the buses that pass through a particular bus stop. I have chosen bus stop name ‘south Park St [northbound] after South St’.



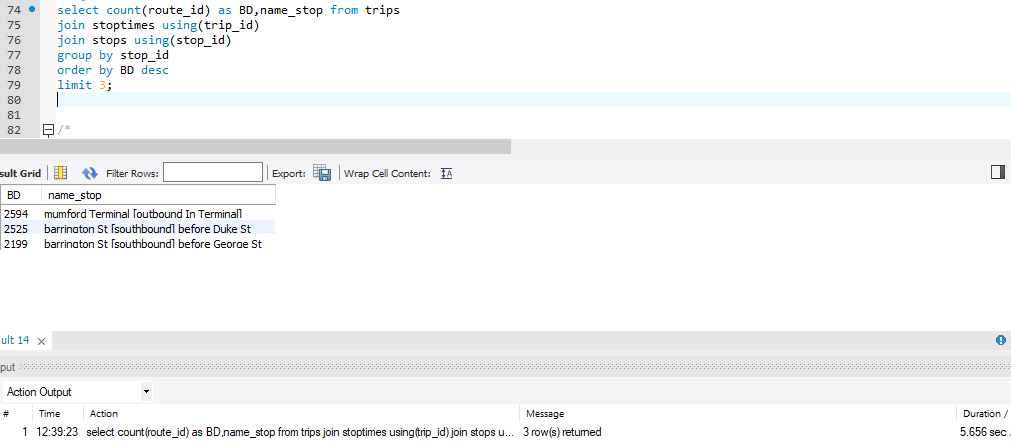
**B.**



**C.**



**D.**



## Elastic Search

## **A.**

## 

## **B.**

## 

## **C.**

## 

## **D.**

## 

## 