AN MRQL VISUALIZER USING JSON INTEGRATION

by

ROHIT BHAWAL

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

THE UNIVERSITY OF TEXAS AT ARLINGTON

May 2017

Supervising Committee:

Dr. Leonidas Fegaras, Supervising Professor

Dr. Ramez Elmasri, Committee Member

Mr. David Levine, Committee Member

To my family.

ACKNOWLEDGEMENTS

I would like to sincerely thank Dr. Leonidas Fegaras, my supervisor who has the attitude and the substance of a genius, to continuously support and motivate me throughout my thesis. Without his invaluable mentorship and encouragement, this would have not been possible.

I would also like to thank my committee members Dr. Ramez Elmasri and Mr. David Levine for giving their interest and precious time in my research. Special thanks to UTA Computer Science Engineering department for their support and cooperation.

In addition to this, I would like to thank some of my family members like Monika Debnath, Biswamber Pal, Somdutta Chakraborty Sarkar, Daipayan Sarkar and special friends like Achyut Paudel, Jasmine Varghese, Upa Gupta, Gokarna Neupane and Ishwor Timilsina for constantly supporting and believing in me.

I am grateful to my mother, my father and my brother who were always there for me when I need them.

I like to thank the open source contributors for Apache MRQL, Flask and D3JS for their clear documentation and examples.

April 13, 2017

ABSTRACT

AN MRQL VISUALIZER USING JSON INTEGRATION

Rohit Bhawal, MS

The University of Texas at Arlington, 2017

Supervising Professor: Leonidas Fegaras

Committee Member: Ramez Elmasri

Committee Member: David Levine

In today's world where there is no limit to the amount of data being collected from IOT devices, social media platforms, and other big data applications, there is a need for systems to process them efficiently and effortlessly. Analyzing the data to identify trends, detect patterns and find other valuable information is critical for any business application. The analyzed data when produced in visual format like graphs, enables one to grasp difficult concepts or identify new patterns easily. MRQL is an SQL-like query language for large scale data analysis built on top of Apache Hadoop, Spark, Flink and Hama which allows to write query for big data analysis.

In this thesis, the MRQL language has been enhanced by adding a JSON generator functionality that allows the language to output results in JSON format based on the input query. The structure of the generated JSON output is query dependent, in that the JSON output is in symmetry with the query specification. This functionality provides for feature integration of MRQL to any external system. In this context, a web application has been developed to use the JSON output and to generate a graphical visualization of query results. This visualizer is an example of integration of MRQL to an external system via the JSON data generator. This helps in providing vital visual information on the analyzed data from the query.

The developed web application allows a user to submit an MRQL query on their Big Data stored on a distributed file system and then to visualize the query result as a graph. The application currently supports MapReduce and Spark as platforms to run MRQL queries, using in-memory, Local, or Distributed mode, based on the environment on which it has been deployed. It enables a user to use the MRQL language to perform data analysis and then visualize the result.

Table of Contents

List of Illustrations

List of Abbreviations

MRQL: Map Reduce Query Language

IOT: Internet Of Things

TB: Tera Byte

I/O: Input Output

HDFS: Hadoop File System

SQL: Structured Query Language

BSP: Bulk Synchronous Parallel

XML: eXtensible Markup Language

JSON: JavaScript Object Notation

D3: Data Driven Documents

JS: Java Script

CSV: Comma Separated Values

UDF: User Defined Function

API: Application Programming Interface

RDBMS: Relational Database Management System

AST: Abstract Syntax Trees

ECMA: European Computer Manufacturers Association

RFC: Request For Comments

DOM: Document Object Model

HTML: Hyper Text Markup Language

SVG: Scalable Vector Graphics

CSS: Cascading Style Sheets

ML: Machine Learning

Chapter 1

Introduction

1.1. Introduction

We live in a world where there is no limit to how data is being captured. Data is being gathered in the form of sensor information from IOT devices, graph information from social media platforms, historical data from business firms, etc. all creates a massive pool of information ready to be tapped to get valuable information. With time the volume of such information grew and grew to such an extent that now the legacy systems processing them sequentially was not an option. New techniques were required to process such volume of data, hence need for distributed processing became a necessity.

1.2. Why Distributed?

As we can see data access speed is getting a scarce resource, moving data is getting more and more time-consuming and expensive. We can safely store our data on big centralized storages, this can be scaled to petabytes with no problem. The problem comes if we want to process and analyze the data. Data access speed of these storages are limited by the disk I/O speeds and network bottlenecks. Moving data over narrow channels somewhere else to process it is not a scalable solution for massive data. The key features of distributed storage and processing are

- Data is stored in servers with processing capabilities i.e. marriage of storage and processing.
- Data is stored in a distributed manner.
- Data is replicated sometimes geographically to avoid data loss.
- Data is processed locally minimizing high cost of moving data.
- Data is processed parallel so processing one TB of data can be done in seconds if we have enough nodes.

1.3. Big Data

Big data is a term for data sets that are so large or complex that traditional data processing application software is inadequate to deal with them. Challenges include capture, storage, analysis, data curation, search, sharing, transfer, visualization, querying, updating and information privacy. The term "big data" often refers simply to the use of predictive analytics, user behavior analytics, or certain other advanced data analytics methods that extract value from data, and seldom to a particular size of data set. Big data has following characteristics

- Volume: The quantity of generated and stored data. The size of the data determines the value and potential insight- and whether it can actually be considered big data or not.
- Variety: The type and nature of the data. This helps people who analyze it to effectively use the resulting insight.
- Velocity: In this context, the speed at which the data is generated and processed to meet the demands and challenges that lie in the path of growth and development.
- Variability: Inconsistency of the data set can hamper processes to handle and manage it.
- Veracity: The quality of captured data can vary greatly, affecting accurate analysis.

1.4. Why is Big Data Important ?

The importance of big data doesn’t revolve around how much data one have, but what can be done with it. One can take data from any source and analyze it to find answers that enable cost reductions, time reductions, new product development and optimized offerings, and smart decision making. When big data is combined with high-powered analytics, business-related tasks can be accomplished, such as:

- Determining root causes of failures, issues and defects in near-real time.
- Generating coupons at the point of sale based on the customer's buying habits.
- Recalculating entire risk portfolios in minutes.
- Detecting fraudulent behavior before it affects the organization.

The primary value from big data comes not from the data in its raw form, but from the processing and analysis of it and the insights, products, and services that emerge from analysis.

1.5. Apache Hadoop

Apache Hadoop is open-source software framework for distributed storage and distributed processing of very large data on computer cluster built from commodity hardware. It is design to automatically handle hardware failures. The storage part is called the Hadoop Distributed File System (HDFS) and the processing part is called the MapReduce. Hadoop splits files into large blocks and distributes them across nodes in a cluster. It then transfers packaged code into nodes to process the data in parallel. This approach takes advantage of data locality, where nodes manipulate the data they have access to. This allows the dataset to be processed faster and more efficiently than it would be in a more conventional supercomputer architecture that relies on a parallel file system where computation and data are distributed via high-speed networking.

Hadoop consists of Mappers and Reducers. Mappers are responsible for splitting the data into large blocks based on the criteria given in mapper code and Reducers are responsible for performing calculation or group operation on the data given by mapper.

The MapReduce framework works on <key, value> pairs i.e. the framework views the input to the job as a set of <key, value> pairs and produces a set of <key, value> pairs as the output of the job.

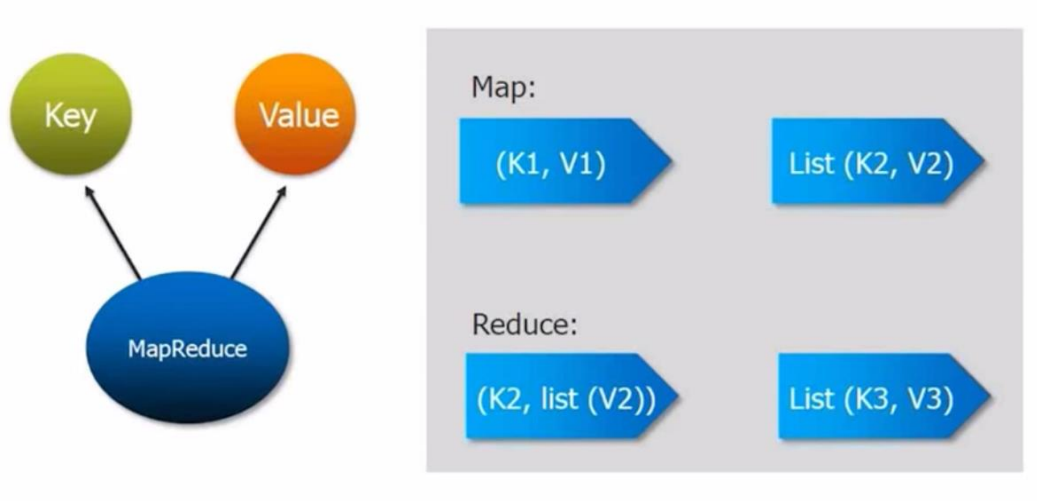

Figure 1-1 MapReduce Key Value

The framework makes MapReduce jobs in which the map tasks splits the input data into independent blocks in parallel manner. Then it sorts the output from map task and feeds the sorted data as input for the reduce tasks. The final result is given by output of the reduce task. The framework automatically schedules the tasks, monitors then and re-execute them in case of failure. (input) <k1, v1> -> map -> <k2, v2> -> combine -> <k2, v2> -> reduce -> <k3, v3> (output)

Following steps are followed in the MapReduce tasks:

- Map Step: Each worker node applies the map() function to the local data, and writes the output to a temporary storage. A master node ensures that only one copy of redundant input data is processed.
- Shuffle Step: Worker nodes redistribute data based on the output keys (produced by the map() function), such that all data belonging to one key is located on the same worker node.
- Reduce Step: Worker nodes now process each group of output data, per key, in parallel.

The overall workflow looks like this:

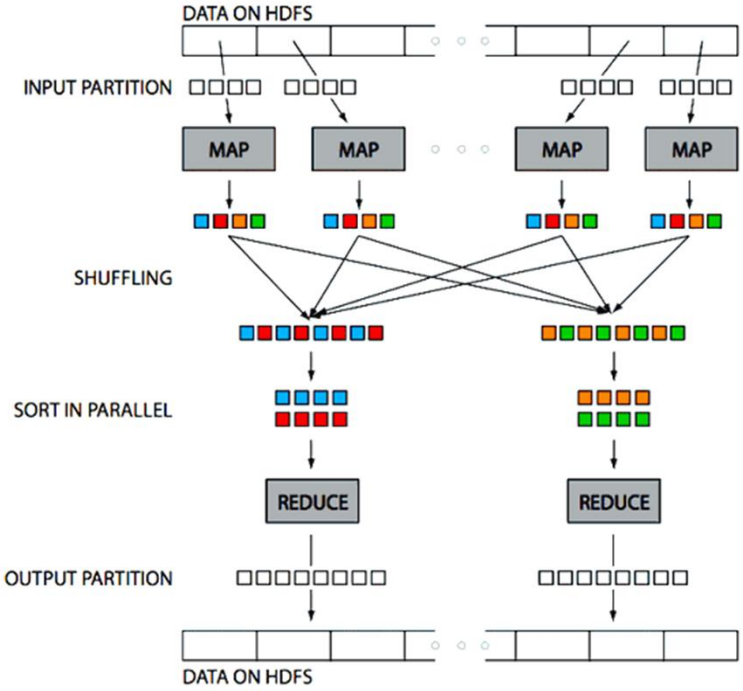

Figure 1-2 Map Reduce Work Flow

1.6. Why is Hadoop Important ?

Ability to store and process huge amounts of any kind of data, quickly: With data volumes and varieties constantly increasing, especially from social media and the Internet of Things (IoT), that's a key consideration.

Computing power: Hadoop's distributed computing model processes big data fast. The more computing nodes you use, the more processing power you have.

Fault tolerance: Data and application processing are protected against hardware failure. If a node goes down, jobs are automatically redirected to other nodes to make sure the distributed computing does not fail. Multiple copies of all data are stored automatically.

Flexibility: Unlike traditional relational databases, you don't have to preprocess data before storing it. You can store as much data as you want and decide how to use it later. That includes unstructured data like text, images and videos.

Low cost: The open-source framework is free and uses commodity hardware to store large quantities of data.

Scalability: You can easily grow your system to handle more data simply by adding nodes. Little administration is required.

Chapter 2

Apache MRQL

2.1. Introduction to Apache MRQL

Apache MRQL is a query processing and optimization system for large-scale, distributed data analysis, built on top of Apache Hadoop, Hama, Spark, and Flink. MRQL (pronounced miracle) is a query processing and optimization system for large-scale, distributed data analysis. MRQL (the MapReduce Query Language) is an SQL-like query language for large-scale data analysis on a cluster of computers. The MRQL query processing system can evaluate MRQL queries in four modes:

- Map-Reduce mode using Apache Hadoop,
- BSP mode (Bulk Synchronous Parallel mode) using Apache Hama,
- Spark mode using Apache Spark, and
- Flink mode using Apache Flink.

The MRQL query language is powerful enough to express most common data analysis tasks over many forms of raw in-situ data, such as XML and JSON documents, binary files, and CSV documents. MRQL is more powerful than other current high-level MapReduce languages, such as Hive and PigLatin, since it can operate on more complex data and supports more powerful query constructs, thus eliminating the need for using explicit MapReduce code. With MRQL, users can express complex data analysis tasks, such as PageRank, k-means clustering, matrix factorization, etc., using SQL-like queries exclusively, while the MRQL query processing system is able to compile these queries to efficient Java code.

Some of the language features include

- It supports custom aggregations/reductions using UDFs provided they have certain properties (associative & commutative)
- It supports iteration declaratively to capture iterative algorithms, such as PageRank
- It supports custom parsing and custom data fragmentation
- It provides syntax-directed construction/deconstruction of data to capture domain-specific languages

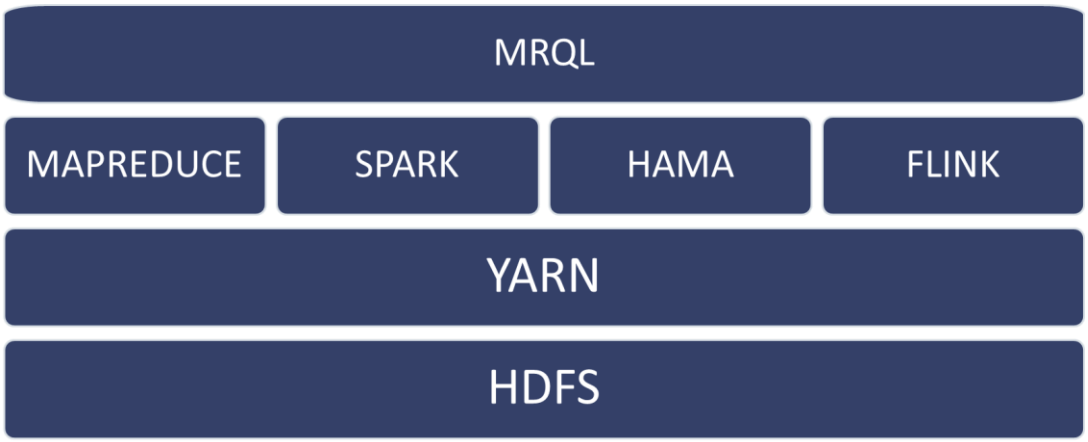

Figure 2-1 MRQL Architecture

2.2. Related Work: Apache Hive

Apache Hive is a data warehouse infrastructure built on top of Hadoop for providing data summarization, query, and analysis. Hive gives an SQL-like interface to query data stored in various databases and file systems that integrate with Hadoop. Traditional SQL queries must be implemented in the MapReduce Java API to execute SQL applications and queries over distributed data. Hive provides the necessary SQL abstraction to integrate SQL-like Queries (HiveQL) into the underlying Java API without the need to implement queries in the low-level Java API.

2.3. Apache MRQL vs Apache Hive

Following is brief comparison between Apache MRQL and Apache Hive

- **Meta Data:** MRQL does not require any metadata to be stored in RDBMS whereas Hive does.
- **Data Support:** MRQL supports nested collections, trees and custom complex data types whereas Hive only supports relational.
- **Group-By Operation:** MRQL allows group-by operation on arbitrary queries whereas Hive does not allow on subqueries.
- **Aggregation:** MRQL allows aggregation on arbitrary queries on grouped data whereas Hive only supports SQL aggregations.
- **Platforms:** MRQL supports Hadoop, Hama, Spark and Flink platforms whereas Hive supports Hadoop, Tez and Spark.
- **Data Source:** MRQL supports text, sequence, XML, JSON file formats as data source whereas Hive supports text, sequence, ORC and RCFile.
- **Iteration & Streaming:** MRQL support iteration and streaming whereas Hive does not.

2.4. MRQL Data Model

MRQL supports the following types:

- a basic type: bool, short, int, long, float, double, string.
- a tuple (t1,..., tn),
- a record < A1: t1, ..., An: tn >,
- a list (sequence) [t] or list(t),
- a bag (multiset) {t} or bag(t),
- a user-defined type

- a data type T
- a persistent collection !list(t), ![t], !bag(t), or !{t}

where t, t1,...,tn are types. MRQL supports the usual arithmetic and comparison operations for numbers. An integer constant is of type int, a real number constant is a float. They can be up-coerced using the syntax e as t. For example, 1 as float. Arithmetic expressions are overloaded to work on multiple numerical types, such as 10+3.4E2. A bool can only be true or false. Boolean conditions can be checked with the if e1 then e2 else e3 syntax and can be combined with the and, or, and not operators. Strings are concatenated with +. Tuples are constructed using (e1, ..., en) and records are constructed using < A1: e1, ..., An: en >, where e1, ..., en are expressions. To get the ith element of a tuple x (starting from 0), use x#i. To get the A component of a record x, use x.A.

Figure 2-2 MRQL Data Model

Lists are constructed using [e1, ..., en] while bags are constructed using { e1, ..., en }, where e1, ..., en are expressions. The difference between a list and a bag is that a list supports order-based operations, such as indexing e1[e2] and subsequence e1[e2:e3]. The range n..m, where n and m are MRQL expressions that evaluate to long integers, creates the list [n,n+1,...,m-1,m] of m-n+1 elements. Lists and bags can be queried using the select-query syntax and can be combined using union, intersect, and except. Any value x of type bag((k,v)) (ie, a bag of pairs), where k and v are arbitrary types, is also a map, which binds keys to values. In addition to bag operations, a map also supports key indexing x[key], for a key value key of type k, to return the value of type v associated with key, if exists (a run-time error otherwise). For example, {('a',1),('b',2)}['a'] returns 1. For a string key 'a' (a name), one may also write x.a for x['a'].

2.5. MRQL Patterns

Patterns are used in select-queries and case statements. They are compiled away from expressions before query optimization. In general, a pattern can be

- a pattern variable that matches any data and binds the variable to data,
- a constant basic value,
- a '*' that matches any data,
- a data construction C(p1, ..., pn),
- a tuple (p1, ..., pn),
- a record < A1: p1, ..., An: pn >,
- a list [p1, ..., pn],

where p1,...,pn are patterns. Note that a record pattern can match a record that has additional components and that the order of components is ignored. A pattern variable can appear multiple times in a pattern or across patterns in nested queries, which requires that these variables be bound to equal values. An irrefutable pattern is a pattern that does not contain constants, data

constructions, lists, or repeated variables. An irrefutable pattern matches any expression of the same type.

2.6. MRQL Accessing Data Sources

The MRQL expression that makes a directory of raw files accessible to a query is:

source(parser,path,...args)

where ‘path’ is the URI of the directory that contains the source files (a string), ‘parser’ is the name of the parser to parse the files, and ‘args’ are various parameters specific to the parsing method. It returns a !bag(t), for some t, that is, it returns a MapReduce type.

For Parsing Flat Files the ‘line’ parser parses record-oriented text documents that contain basic values separated by a user-defined delimiter (a string):

source(line, path, c, type(< A1: t1, ..., An: tn >))

or

source(line, path, c, type((t1, ..., tn)))

where ‘ti’ is a basic type or any, and ‘c’ is the delimiter.

For example, the expression

source(line,'employee.txt',',',type(<name:string,dno:int,phone:any,address:string>))

parses a CSV file and returns a bag(<name:string,dno:int,address:string>), since the phone is skipped.

Similarly, for XML parsing, the MRQL expression is:

source(xml, path, tags, xpath)

where ‘tags’ is a bag of synchronization tags and ‘xpath’ is the XPath expression used for fragmentation and for JSON parsing, the MRQL expression is:

source(json, path, names)

where ‘names’ is a bag of synchronization attribute names in JSON objects.

2.7. MRQL Query Syntax

The Query syntax is as follows:

select [**distinct**] e
from p_1 **in** e_1, ..., p_n **in** e_n
[**where** e_c]
[**group by** p': e' [**having** e_h]]
[**order by** e_0 [**limit** e_l]]

- **[]** is optional and means sequence of similar bindings.
- **e, e_1,...,e_n, e_c, e', e_h, e_0,** and **e_l** are arbitrary MRQL expressions. It can contain other nested queries.
- **from p in e** are forms of query bindings where p is a pattern and e is MRQL expression that returns a collection. The pattern p matches each element in the collection e, binding its pattern variables to the corresponding values in the element i.e. it specifies an iteration over the collection e, one element at a time, causing the pattern p to be matched with the current collection element.
- **where** e_c is used for applying filter conditions to the query.
- **group by p': e'** partitions the query results into groups allowing each member of group to have the same e' value. The pattern p' is bound to the group-by value, which is unique for each group and is common across the group members.
- **having e_h** is used for applying filter conditions on the groups.
- **order by e_0** is used for ordering the result of query by e_0 values. By default ordering is done in ascending but if **inv()** is used then ordering is inverted i.e. done in descending manner.
- **limit e_l** is used to specify number of records to return.

MRQL query works on collections of values, which are treated as bags by the query, and returns a new collection of values. If it is an order-by query, the result is a list, otherwise, it is a bag.

2.8. MRQL Query Execution Flow

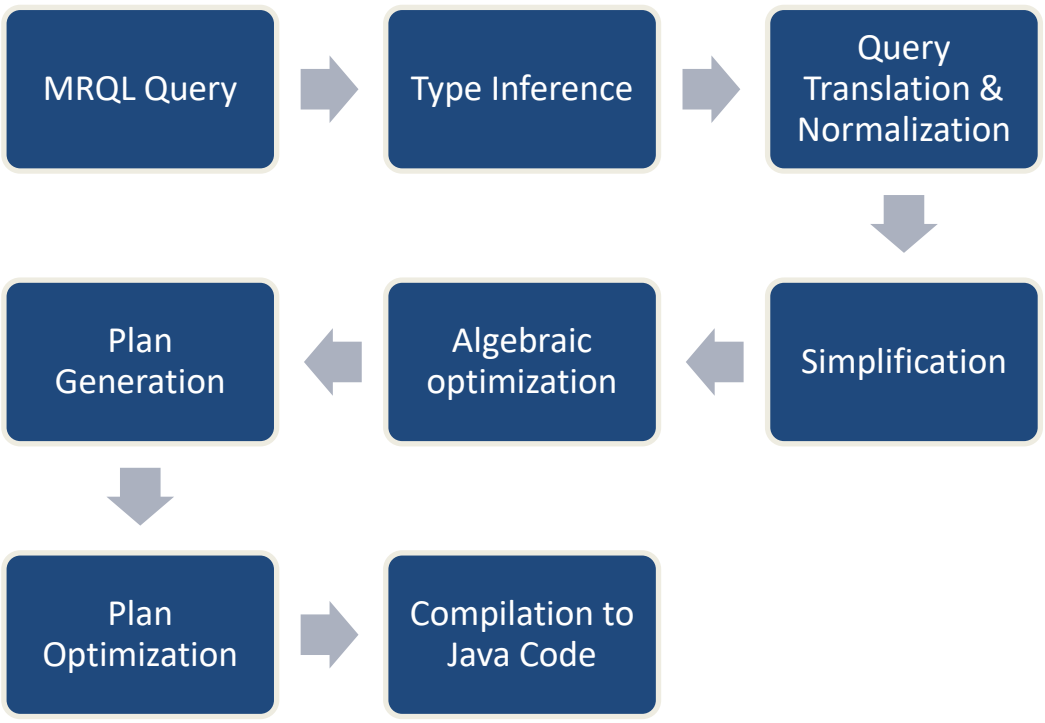

Figure 2-3 MRQL Query Execution Flow

Chapter 3

Problem Statement

3.1. Motivation

The output currently generated by MRQL is not in a standard format. It changes with change in query translation. For instance, if query consists of tuple in selection then output format is different than if the query had record in selection. Due to this it is difficult to integrate the system with external system if the given input query is not consistent. Moreover, the external system has to create their own parser in order to extract information from the generated output and this acts as an extra overhead for different types of generated output. In this thesis, the problem of non-standard output generation by MRQL is addressed by creating a system function within the language which when used in the query generates standard JSON output. Then this JSON output is used to perform visualization using d3js library. This thesis, is divided into two phases

Phase One: To develop system function in MRQL which generates json output

Phase Two: To integrate MRQL with Web Application and perform visualization with result

3.2. Phase One

This phase involves to generate json output from MRQL which follows the ECMA-404 and RFC 7159 standards allowing it to be parsed in any programming language by any application. MRQL uses the Gen package to construct and transform algebraic expressions and physical plans. Gen is a Java preprocessor that extends the Java language with special syntax to make the task of handling Abstract Syntax Trees (ASTs) easier. The MRQL scanner is described in mrql.lex, which provides the class MRQLLex. The MRQL syntax (grammar) is described in mrql.cgen, which provides the class MRQLParser. This is a .gen file that needs to be converted to a .cup file using Gen, and then to be processed by the CUP parser generator, which is a bottom-up shift-reduce parser generator. A custom function called ‘dumpjson’ is created which is written as a token in the

mrql.cgen file. The link between the token and system call is written in mrql.lex. The system call is used to call the actual method which performs the json output creation which is written in Evaluater.java. To summarize,

- Adding ‘dumpjson’ as system function in MRQL language to generate json output
- First specifying ‘dumpjson’ as token_name and linking with tokens in mrql.lex

  <YYINITIAL> "dumpjson" { return symbol(sym.DUMPJSON); }
- Then stating the token to a method call in mrql.cgen

  DUMPJSON String:s FROM expr:e {: RESULT = #<dump_json(`(new StringLeaf(s)),`e)>; :}
- Finally specifying the method call in TopLevel.gen

```
private final static void dump_json ( String file, Tree e ) {
        MRData res = expression(e,false);
        if (res != null)
        try {
                Evaluator.evaluator.dump_json(file,query_type,res);
        } catch (Exception x) {
                throw new Error(x);
        }
}
```

To generate the json output using the ‘dumpjson’ system function, the syntax is

dumpjson ‘location’ from
select [**distinct**] e
from p_1 **in** e_1, ..., p_n **in** e_n
[**where** e_c]
[**group by** p': e' [**having** e_h]]
[**order by** e_0 [**limit** e_l]]

3.3. Why JSON ?

For quite sometime XML was the only choice for open data interchange. But over the years there has been a lot of transformation in the world of open data sharing. The more lightweight JSON (JavaScript Object Notation) has become a popular alternative to XML for various reasons.

- **Less verbose:** XML uses more words than necessary
- **JSON is faster:** Parsing XML software is slow and cumbersome. Many of these DOM manipulation libraries can lead applications to use large amount of memory due to the verbosity and cost of parsing large XML files.

- **JSON data model's structure matches the data:** JSON's data structure is a map whereas XML is a tree. A map works in key/value pairs which is easier to interpret and is predictable.

- **In code:** Items are represented the same way in code. In many languages, especially dynamic ones, JSON can be interpreted using predefine library and immediately get domain object. It is easy to go from objects in JSON to the objects in code because they align. When going from objects in XML to objects in code they do not align and there is a lot of room for interpretation.

3.4. Phase Two

This phase includes development of a web application which allows user to write MRQL query and then visualize the json result. Visualization is done using D3JS library to visualize the output in line/bar graph. The web application is created using Flask microframework in python. It application is designed to accept any MRQL query and then show the output on the screen and the option to visualize the output. Internally, the web application passes the user written MRQL query

to MRQL system by adding the 'dumpjson' in the query and then fetches the generated output from MRQL to show it as result enabling a visualize button. When the visualize button is clicked, then graph is showed using the json output with the help of D3JS library.

3.5. Why D3JS ?

D3JS is a JavaScript library for manipulating documents based on data. D3 stands for Data Driven Document. It allows to bind arbitrary data to a Document Object Model (DOM), and then apply data-driven transformations to the document. It helps bringing data to life using HTML, SVG, and CSS. It's emphasis on web standards gives the full capabilities of modern browsers without tying to a proprietary framework, combining powerful visualization components and a data-driven approach to DOM manipulation. It is extremely fast, supporting large datasets and dynamic behaviors for interaction and animation. Its functional style allows code reuse through a diverse collection of components and plugins.

Chapter 4

Design

4.1. Work Flow Diagram

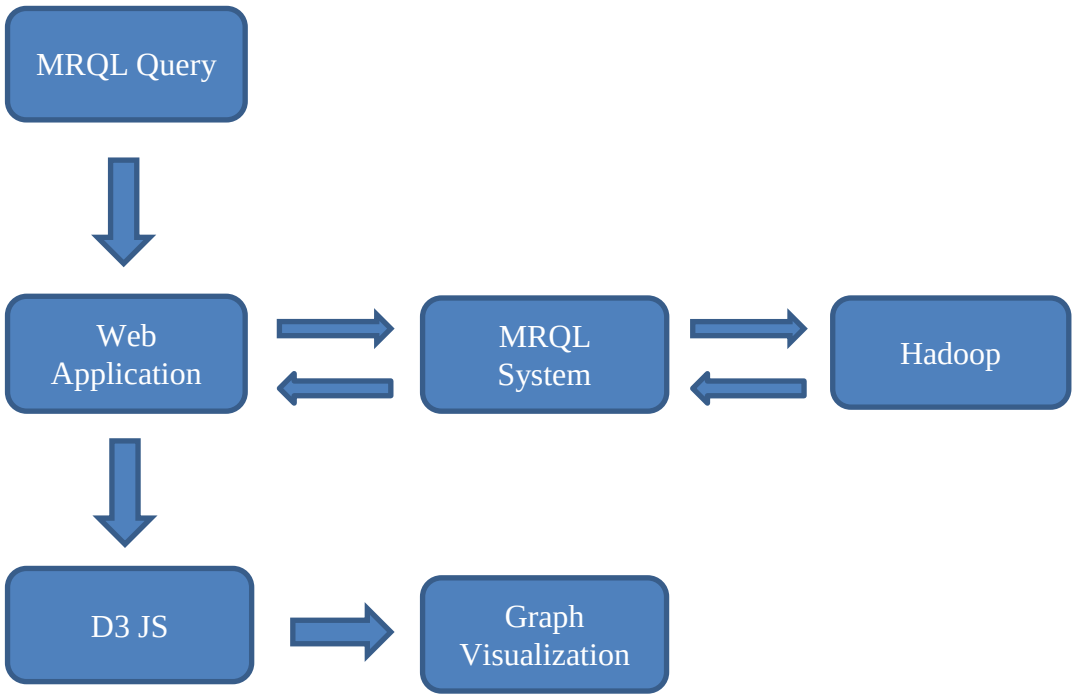

Figure 4-1 Work Flow

4.2. Description

The MRQL query and the data is given as input to the Web Application. The web application takes the input and puts both in a working directory which is set in the web application configuration. It then appends ‘dumpjson’ with output location to the query and sends it to the MRQL System. MRQL translates the query, generates the query plan and then creates the require .jar and sends it to Hadoop. Once Hadoop completes the processing, MRQL retrieves the result in stream and based on the query type translation generates the json output. This generated json output is then used by the web application to render graph visualization using D3js library.

Chapter 5

Implementation

5.1. Phase One: Tokenization

Tokens are defined by regular expressions, which are understood by a lexical analyzer generator such as lex. The lexical analyzer reads in a stream of characters, identifies the lexemes in the stream, and categorizes them into tokens. Here ‘dumpjson’ token is created and specified in mrql.lex file.

```
.
.
Line 174:
<YYINITIAL> "dumpjson"                    { return symbol(sym.DUMPJSON); }
<YYINITIAL> "incr"                { return symbol(sym.INCR); }
<YYINITIAL> "type"                { return symbol(sym.TYPE); }
<YYINITIAL> "data"                { return symbol(sym.DATA); }
<YYINITIAL> "case"                { return symbol(sym.CASE); }
<YYINITIAL> "xpath"               { return symbol(sym.XPATH); }
<YYINITIAL> "repeat"              { return symbol(sym.REPEAT); }
<YYINITIAL> "step"                { return symbol(sym.STEP); }
<YYINITIAL> "limit"               { return symbol(sym.LIMIT); }
<YYINITIAL> "import"              { return symbol(sym.IMPORT); }
<YYINITIAL> "parser"              { return symbol(sym.PARSER); }
<YYINITIAL> "include"             { return symbol(sym.INCLUDE); }
<YYINITIAL> "aggregation"         { return symbol(sym.AGGREGATION); }
<YYINITIAL> "trace"           { return symbol(sym.TRACE); }

<YYINITIAL> {ID}                  { return symbol(sym.Variable,yytext()); }

<YYINITIAL> "/*"              { yybegin(COMMENT); }
<COMMENT> "*/"              { yybegin(YYINITIAL); }
<COMMENT> [ \t\f]           { }
<COMMENT> [\r\n]            { prev_char_pos = yychar; }
.
.
.
.
```

5.2. Phase One: Scanner & Evaluator

Scanner has encoded within it information on the possible sequences of characters that can be contained within any of the tokens it handles. Individual instances of these character sequences are termed lexemes. To construct a token, the lexical analyzer goes over the characters of the lexeme to produce a value. The lexeme's type combined with its value is what properly constitutes a token, which can be given to a parser. In this, it is specified in mrql.cgen file.

```
.
.
.
Line 18:
package org.apache.mrql;

import java_cup.runtime.*;
import org.apache.mrql.gen.*;

parser code {:

    static int[] tokens = {
        sym.error, sym.IF, sym.THEN, sym.ELSE, sym.SELECT, sym.FROM, sym.HAVING,
        sym.LB, sym.RB, sym.LP, sym.RP, sym.LSB, sym.RSB,
        sym.PLUS, sym.MINUS, sym.TIMES, sym.DIV, sym.MOD, sym.EQ,
        sym.NEQ, sym.LT, sym.LEQ, sym.GT, sym.GEQ, sym.SHARP, sym.AS,
        sym.AND, sym.OR, sym.NOT, sym.UNION,
        sym.INTERSECT, sym.EXCEPT, sym.EXISTS, sym.IN, sym.DOT, sym.COLON, sym.COMMA,
        sym.SEMI, sym.ASSIGN, sym.WHERE, sym.ORDER, sym.GROUP, sym.BY, sym.ASCENDING,
        sym.DESCENDING, sym.FUNCTION, sym.DISTINCT, sym.BSLASH, sym.SOME, sym.ALL,
        sym.SOME, sym.ALL, sym.GTR, sym.SEP, sym.STORE, sym.DUMP, sym.DUMPJSON, sym.TYPE,
sym.DATA, sym.REPEAT,
        sym.STEP, sym.LIMIT, sym.LET, sym.ATSYM, sym.EXCLAMATION,
        sym.Variable, sym.Integer, sym.Double, sym.String, sym.Decimal,
        sym.START_TEMPLATE, sym.END_TEMPLATE, sym.TEXT, sym.TRACE, sym.INCR
    };

    static String[] token_names = {
        "error", "if", "then", "else", "select", "from", "having",
        "[", "]", "(", ")", "{", "}",
        "+", "-", "*", "/", "mod", "=",
        "<>", "<", "<=", ">", ">=", "#", "as",
        "and", "or", "not", "union",
        "intersect", "except", "exists", "in", ".", ":", ",",
        ";", ":=", "where", "order", "group", "by", "ascending",
        "descending", "function", "distinct", "\\", "some", "all",
        "some", "all", ">", "|", "store", "dump","dumpjson", "type", "data", "repeat",
        "step", "limit", "let", "@", "!",
        "Variable", "Integer", "Double", "String", "Decimal",
        "[|", "|]", "Text", "trace", "incr"
    };
.
```

```
.
.
.
Line 95:
/* Terminals (tokens returned by the scanner). */
terminal IF, THEN, ELSE, SELECT, FROM, HAVING, LB, RB, LP, RP, LSB, RSB, LDOT, SHARP,
        PLUS, MINUS, TIMES, DIV, MOD, EQ, NEQ, LT, LEQ, GT, GEQ, AND, OR, NOT, AS,
        UNION, INTERSECT, EXCEPT, EXISTS, IN, COMMA, DOT, COLON, ASSIGN, SEMI, WHERE,
        ORDER, GROUP, BY, ASCENDING, DESCENDING, UMINUS, FUNCTION, DISTINCT,
BSLASH,
        SOME, ALL, GTR, SEP, STORE, TYPE, DATA, CASE, ATSYM, XPATH, REPEAT, STEP, LIMIT,
        LET, IMPORT, PARSER, AGGREGATION, INCLUDE, EXCLAMATION, MACRO, DUMP,
DUMPJSON, TRACE, INCR;

terminal String         Variable;
terminal Long           Integer;
terminal Double         Double;
terminal String         String;
terminal Double         Decimal;
terminal String         START_TEMPLATE;
terminal String         END_TEMPLATE;
terminal String         TEXT;

non terminal            prog;
non terminal Tree       item, expr, var, const, mrql, pattern, opt_where, opt_orderby,
                        mode, opt_groupby, opt_having, opt_distinct, type, xpath, xname,
                        xpred, opt_limit, unit;
non terminal Trees      expl, name_binds, pat_list, pat_binds,
                        binds, order_binds, groupby_binds, typel, type_binds,
                        data_binds, cases, fnl, template, template_pat, fnc_params, var_list;

precedence nonassoc     LDOT, ASSIGN, LIMIT;
precedence nonassoc     ELSE, COLON;
precedence nonassoc     ORDER, GROUP, HAVING, WHERE;
precedence left         INTERSECT, UNION, EXCEPT, IN;
precedence nonassoc     ASCENDING, DESCENDING;
precedence nonassoc     COMMA, LP;
precedence right        OR;
precedence right        AND;
precedence nonassoc     NOT;
precedence nonassoc     EQ, LT, GT, LEQ, GEQ, NEQ;
precedence left         PLUS, MINUS;
precedence left         TIMES, DIV, MOD;
precedence nonassoc     DOT, SHARP, LB, AS;
precedence nonassoc     UMINUS;
.
.
.
```

```
.
.
.
Line 135:
start with prog;

prog          ::= item:i SEMI                    {: Translator.top_level(i); :}
            |  prog item:i SEMI                  {: Translator.top_level(i); :}
            ;
item          ::= expr:e                         {: RESULT = #<expression(`e)>; :}
            |  var:v EQ expr:e                   {: RESULT = #<assign(`v,`e)>; :}
            |  STORE var:v ASSIGN expr:e         {: RESULT = #<store(`v,`e)>; :}
            |  STORE String:s FROM expr:e        {: RESULT = #<dump(`(new StringLeaf(s)),`e)>; :}
            |  DUMP String:s FROM expr:e         {: RESULT = #<dump_text(`(new StringLeaf(s)),`e)>; :}
            |  DUMPJSON String:s FROM expr:e     {: RESULT = #<dump_json(`(new StringLeaf(s)),`e)>; :}
            |  INCR expr:e                       {: RESULT = #<incr(`e)>; :}
            |  TYPE var:v EQ type:t              {: RESULT = #<typedef(`v,`t)>; :}
            |  DATA var:v EQ data_binds:nl       {: RESULT = #<datadef(`v,union(...nl))>; :}
.
.
.
```

5.3. Phase One: Query Translation to JSON output

The linking of token to actual functional call is mentioned in TopLevel.gen file

```
.
.
.
Line 221:
/** dump the result of evaluating the MRQL query e to a text JSON file*/
   private final static void dump_json ( String file, Tree e ) {
      MRData res = expression(e,false);
      if (res != null)
         try {
                  Evaluator.evaluator.dump_json(file,query_type,res);
         } catch (Exception x) {
            throw new Error(x);
         }
   }
.
.
.
```

During query translation, type inference is done i.e. the output structure based on input query is build. This is used to build the JSON output. For example consider below query

select (k,sum(o.TOTALPRICE))
from o in O, c in C
where o.CUSTKEY=c.CUSTKEY
group by k: c.NAME

The type inference for above query will be *bag(tuple(String, Double))* i.e.

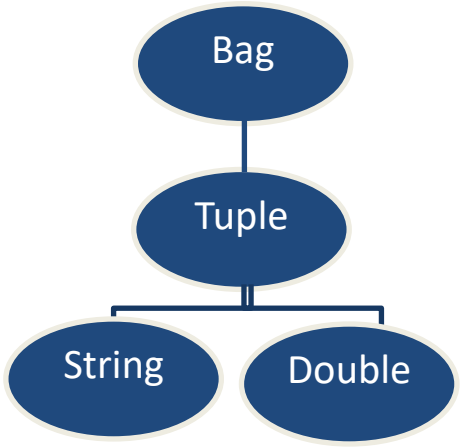

Figure 5-1 Type Inference

The type inference is parsed as a tree to get the structure and then based on data type the json is formed. Along with type inference, a counter is kept to take count of auto attributes generated for tuple data type. The logic is written in Evaluator.java file.

```
…
Line 164:
/** dumpjson MRQL data into a JSON file */
public void dump_json ( String file, Tree type, MRData data ) throws Exception {
   int ps = Config.max_bag_size_print;
          Config.max_bag_size_print = -1;
      if (!Config.hadoop_mode) {
         File parent = new File(file).getParentFile();
         if (parent != null && !parent.exists())
           parent.mkdirs();
      };
          final PrintStream out = (Config.hadoop_mode)
                        ? Plan.print_stream(file)
                        : new PrintStream(file);

          print_json(data, type, 0, out);

          Config.max_bag_size_print = ps;
          out.close();
   }
```

```
private void print_json ( MRData x, Tree type, int attrCount, PrintStream out ) {
          try {
                    if (x instanceof Inv){
                              print_json(((Inv)x).value(),type, attrCount, out);
                              return;
                    }
         if (type.equals(new VariableLeaf("XML"))){
            printtype_XML((Union)x, out);
            return;
         }
         if (type.equals(new VariableLeaf("JSON"))){
            printtype_JSON((Union)x, out);
            return;
         }

         boolean FOUND_1 = false;
        Tree E_1 =  TypeInference.expand(type) ;

                     if ((E_1 instanceof Node) && ((Node)E_1).name==Tree.add("persistent") &&
((Node)E_1).children.tail!=null && ((Node)E_1).children.tail.tail==null)
                    {
                              Tree tp = ((Node)E_1).children.head;  FOUND_1=true;
                              print_json(x,tp, attrCount, out);
                              return;
                    }

                     if (!FOUND_1)
                    {
                              if ((E_1 instanceof Node) && ((Node)E_1).name==Tree.add("Bag") &&
((Node)E_1).children.tail!=null && ((Node)E_1).children.tail.tail==null)
                                        {
                                        Tree tp = ((Node)E_1).children.head;
                                        FOUND_1=true;
                                        if (x instanceof MR_dataset) {
                                                  DataSet ds = ((MR_dataset)x).dataset();
                                                  List<MRData> vals = ds.take(Config.max_bag_size_print);
                                                  if (vals.size() == 0){
                                                            out.print("[]");
                                                            return;
                                                  }
                                                  out.print("[ ");
                                                  print_json(vals.get(0),tp, attrCount, out);
                                                  for ( int i = 1; i < vals.size(); i++ ){
                                                            out.print(", ");
                                                            print_json(vals.get(i),tp, attrCount, out);
                                                  }

                                                  out.print(" ]");
                                                  return;
                                        }
```

```
                                        else{
                                                print_json(Evaluator.evaluator.toBag(x),new
Node("bag",Trees.nil.append(Meta.escape(tp))), attrCount, out);
                                                return;
                                        }
                                }
                        }

                        if (!FOUND_1)
                        {
                                if ((E_1 instanceof Node) && ((Node)E_1).name==Tree.add("List") &&
((Node)E_1).children.tail!=null && ((Node)E_1).children.tail.tail==null)
                                {
                                        Tree tp = ((Node)E_1).children.head;
                                        FOUND_1=true;
                                        if (x instanceof MR_dataset) {
                                                DataSet ds = ((MR_dataset)x).dataset();
                                                List<MRData> vals = ds.take(Config.max_bag_size_print);
                                                if (vals.size() == 0){
                                                        out.print("[]");
                                                                return;
                                                }
                                                out.print("[ ");
                                                print_json(vals.get(0),tp, attrCount, out);
                                                for ( int i = 1; i < vals.size(); i++ ){
                                                        out.print(", ");
                                                        print_json(vals.get(i),tp, attrCount, out);
                                                }
                                                out.print(" ]");
                                                return;
                                        }
                                        else{
                                                print_json(Evaluator.evaluator.toBag(x),new
Node("list",Trees.nil.append(Meta.escape(tp))), attrCount, out);
                                                return;
                                        }
                                }

                        }

                        if (!FOUND_1)
                        {
                                if ((E_1 instanceof Node) && ((Node)E_1).name==Tree.add("bag") &&
((Node)E_1).children.tail!=null && ((Node)E_1).children.tail.tail==null)
                                {
                                        Tree tp = ((Node)E_1).children.head;
                                        FOUND_1=true;
                                        Bag b = (Bag)x;
                                        Iterator<MRData> bi = b.iterator();
                                        if (!bi.hasNext()){
```

```
                                        out.print("[]");
                                        return;
                                }
                                out.print("[ ");
                                print_json(bi.next(),tp, attrCount, out);
                                for ( long i = 1; bi.hasNext() ; i++ ){
                                        out.print(", ");
                                        print_json(bi.next(),tp, attrCount, out);
                                }
                                out.print(" ]");
                                return;
                        }
                }

                if (!FOUND_1)
                {
                        if ((E_1 instanceof Node) && ((Node)E_1).name==Tree.add("list") &&
((Node)E_1).children.tail!=null && ((Node)E_1).children.tail.tail==null)
                        {
                                Tree tp = ((Node)E_1).children.head;
                                FOUND_1=true;
                                Bag b = (Bag)x;
                                Iterator<MRData> bi = b.iterator();
                                if (!bi.hasNext()){
                                        out.print("[]");
                                        return;
                                }
                                out.print("[ ");
                                print_json(bi.next(),tp, attrCount, out);
                                for ( long i = 1; bi.hasNext(); i++ ){
                                        out.print(", ");
                                        print_json(bi.next(),tp, attrCount, out);
                                }
                                out.print(" ]");
                                return;
                        }
                }

                if (!FOUND_1)
                {
                        if ((E_1 instanceof Node) && ((Node)E_1).name==Tree.add("tuple"))
                        {
                                Trees el = ((Node)E_1).children;
                                FOUND_1=true;
                                Tuple t = (Tuple)x;
                                if (t.size() == 0){
                                        out.print("{}");
                                        return;
                                }
```

```
                                        out.print("{"+quotedString("~"+String.valueOf(attrCount))+":");
                                        print_json(t.get((short)0),el.nth(0), attrCount, out);
                                        for ( short i = 1; i < t.size(); i++ ){
                                                attrCount++;

        out.print(","+quotedString("~"+String.valueOf(attrCount))+":");
                                                print_json(t.get(i),el.nth(i), attrCount, out);
                                        }
                                        out.print("}");
                                        return;
                                }
                        }

                        if (!FOUND_1)
                        {
                                if ((E_1 instanceof Node) && ((Node)E_1).name==Tree.add("record"))
                                {
                                        Trees el = ((Node)E_1).children;  FOUND_1=true;
                                        Tuple t = (Tuple)x;
                                        if (t.size() == 0){
                                                out.print("{}");
                                                return;
                                        }
                                        out.print("{");

                                        boolean FOUND_2 = false;
                                        Tree E_2 =  el.nth(0) ;

                                        if ((E_2 instanceof Node) && ((Node)E_2).name==Tree.add("bind")
&& ((Node)E_2).children.tail!=null && ((Node)E_2).children.tail.tail!=null &&
((Node)E_2).children.tail.tail.tail==null)
                                        {
                                                Tree a = ((Node)E_2).children.head;
                                                Tree tp = ((Node)E_2).children.tail.head;
                                                FOUND_2=true;
                                                out.print(quotedString(a.toString()) +": ");
                                                print_json(t.get((short)0),tp,attrCount,out);
                                        }

                                        for ( short i = 1; i < t.size(); i++ )
                                        {
                                                boolean FOUND_3 = false;
                                                Tree E_3 =  el.nth(i) ;

                                                if ((E_3 instanceof Node) &&
((Node)E_3).name==Tree.add("bind") && ((Node)E_3).children.tail!=null &&
((Node)E_3).children.tail.tail!=null && ((Node)E_3).children.tail.tail.tail==null)
                                                {
                                                        Tree a = ((Node)E_3).children.head;
                                                        Tree tp = ((Node)E_3).children.tail.head;
```

```
                                        FOUND_3=true;
                                        out.print(", "+quotedString(a.toString())+": ");
                                        print_json(t.get(i),tp,attrCount,out);
                                }
                        }

                        out.print("}");
                        return;
                }
        }

        if (!FOUND_1)
        {
                if ((E_1 instanceof Node) && ((Node)E_1).name==Tree.add("union"))
                {
                        Trees el = ((Node)E_1).children;  FOUND_1=true;
                        Union u = (Union)x;

                        boolean FOUND_2 = false;
                        Tree E_2 =  el.nth(u.tag()) ;

                        if ((E_2 instanceof Node) && ((Node)E_2).children.tail!=null &&
(((Node)E_2).children.head instanceof Node) &&
((Node)((Node)E_2).children.head).name==Tree.add("tuple") && ((Node)E_2).children.tail.tail==null)
                        {
                                String c = ((Node)E_2).name;
                                Trees ts = ((Node)((Node)E_2).children.head).children;
                                FOUND_2=true;
                                out.print(c);
                                print_json(u.value(),new
Node("tuple",Trees.nil.append(ts)),attrCount,out);
                                return;
                        }
                        if (!FOUND_2)
                        {
                                if ((E_2 instanceof Node) && ((Node)E_2).children.tail!=null
&& ((Node)E_2).children.tail.tail==null)
                                {
                                String c = ((Node)E_2).name;
                                Tree tp = ((Node)E_2).children.head;
                                FOUND_2=true;
                                out.print(c+"{");
                                print_json(u.value(),tp,attrCount,out);
                                out.print("}");
                                return;
                                }

                        }
                }
        }
```

```
        String final_result = "";
                    try{
           double temp = Double.valueOf(x.toString());
           final_result = x.toString();
        }
        catch (NumberFormatException nfExp){
          final_result = x.toString().replace("\"","");
           final_result = quotedString(final_result);
        }

                    out.print(final_result);
                    return;
          }
          catch (Exception ex) {
        throw new Error(ex);
      }
   }

private String quotedString(String input){
                    return "\"" + input + "\"";
          }

private static void printtype_XML ( final Union x,  PrintStream out) {
      if (x.tag() == 1){
           out.print(((MR_string)x.value()).get());
           return;
      }
      Tuple t = (Tuple)x.value();
      out.print("<"+((MR_string)t.get(0)).get());
      for ( MRData a: (Bag)t.get(1) ) {
         Tuple attr = (Tuple)a;
         out.print(" "+((MR_string)attr.first()).get()+"=\""
                  +((MR_string)attr.second()).get()+"\"");
      };
      Bag c = (Bag)t.get(2);
      if (c.size() == 0){
           out.print("/>");
           return;
      }
      out.print(">");
      for ( MRData e: c ){
           printtype_XML((Union)e, out);
      }
      out.print("</"+((MR_string)t.get(0)).get()+">");
      return;
   }
```

```
private static void printtype_JSON ( final Union x, PrintStream out) {
            Boolean first = Boolean.TRUE;
            switch (x.tag()) {
       case 0:
            first = Boolean.TRUE;
            out.print("{ ");
          for ( MRData e: (Bag)x.value() ) {
            if (!first){
                        out.print(", ");
                        first = Boolean.FALSE;
            }
             Tuple t = (Tuple)e;
             out.print(t.get(0)+": ");
             printtype_JSON((Union)t.get(1), out);
          };
          out.print(" }");
          return;
       case 1:
            first = Boolean.TRUE;
            out.print("[ ");
          for ( MRData e: (Bag)x.value() ){
            if (!first){
                        out.print(", ");
                        first = Boolean.FALSE;
            }
            printtype_JSON((Union)e, out);
          }
          out.print(" ]");
          return;
       };
       out.print(""+x.value());
       return;
    }
…
```

Usage example

- Query containing record in select statement
  select <node : x.id, rank: x.rank> where x.id = int and x.rank = int

  Output JSON
  [{"node": 0, "rank": 0.6}, {"node": 1, "rank": 0.7},...{"node": 100, "rank": 0.3}]

- Query containing tuple in select statement
  select (x.id, x.rank) where x.id = int and x.rank = int

  Output JSON auto generates the json attribute as ~0 , ~1 ,etc.
  [{"~0": 0, "~1": 0.6}, {"~0": 1, "~1": 0.7},...{"~0": 100, "~1": 0.3}]

5.4. Phase Two: Web Application

The web application is in python using Flask micro framework which is based on Werkzeug WSGI toolkit libraries and the Jinja 2 template engine. Jinja provides python like expression and ensures that the templates are evaluated in a sandbox. It can be used to generate any markup as it is text-based template language. It is flask's default template engine. Server side code below

```
'''
Created on Jan 28, 2017

@author: Rohit Bhawal


References : 1. http://flask.pocoo.org/docs/
             2. https://pythonprogramming.net/
             3. http://stackoverflow.com
'''
import hashlib
import os
from flask import Flask, render_template, request, url_for, redirect, flash, session
from flask import make_response, current_app
from datetime import timedelta
from flask_session import Session
from functools import wraps, update_wrapper
import subprocess
import tempfile
from werkzeug import secure_filename
import shutil
from shutil import copyfile
import json
from threading import Thread
import ast

sess = Session()
app = Flask(__name__)
app.secret_key = 'rohitbhawalskynet'

Uname = ''

# MRQL_EXEC_DIR = '/home/rohitbhawal/Workspace/apache-mrql-0.9.6-incubating/bin/' #Dev Env
# MRQL_EXEC_DIR = '/home/ubuntu/apache-mrql-0.9.6-incubating/bin/' #Test Env
MRQL_EXEC_DIR = '/export/home/rohitbhawal/apache-mrql-0.9.6/bin/'  #Prod Env

# HADOOP_EXEC_DIR = ['/usr/local/hadoop/bin/hadoop'] #Dev Env & Test Env
HADOOP_EXEC_DIR = ['/share/apps/hadoop/bin/hadoop', '--config', '/share/apps/conf-yarn'] #Prod Env

# HADOOP_CLUSTER_LINK = 'http://rohitbhawal.com:8088/cluster' #Test Env
HADOOP_CLUSTER_LINK = 'http://hadoop.uta.edu:8088/cluster' #Prod Env
```

```
HOME_DIR = os.path.dirname(__file__)
WORK_DIR = os.path.join(tempfile.gettempdir(),'WebMRQL')
HDFS_DIR = '/user/rohitbhawal'
DATAFILELIST = 'userDataFilesList.txt'
LOGINFILE = 'logins.txt'
DATAFOLDER = 'data'
QUERYFOLDER = 'queries'
QUERYFILE = 'query.mrql'
QUERYFILE_DIST = 'query_dist.mrql'
THREADINFOFILE = 'thread_info.mrqlweb'
RESULTJSON = 'result.json'
JSONKEYSEP = "_"
NETID_LOGIN = True
SESSION_ALIVE_TIMEOUT_MINUTES = 60
SUB_URL = '/mrql'    #Test Env & Prod Env
# SUB_URL = "    #Dev Env


def crossdomain(origin=None, methods=None, headers=None,
                max_age=21600, attach_to_all=True,
                automatic_options=True):
    if methods is not None:
        methods = ', '.join(sorted(x.upper() for x in methods))
    if headers is not None and not isinstance(headers, basestring):
        headers = ', '.join(x.upper() for x in headers)
    if not isinstance(origin, basestring):
        origin = ', '.join(origin)
    if isinstance(max_age, timedelta):
        max_age = max_age.total_seconds()

    def get_methods():
        if methods is not None:
            return methods

        options_resp = current_app.make_default_options_response()
        return options_resp.headers['allow']

    def decorator(f):
        def wrapped_function(*args, **kwargs):
            if automatic_options and request.method == 'OPTIONS':
                resp = current_app.make_default_options_response()
            else:
                resp = make_response(f(*args, **kwargs))
            if not attach_to_all and request.method != 'OPTIONS':
                return resp
            h = resp.headers

            h['Access-Control-Allow-Origin'] = origin
            h['Access-Control-Allow-Methods'] = get_methods()
```

```
            h['Access-Control-Max-Age'] = str(max_age)
            if headers is not None:
                h['Access-Control-Allow-Headers'] = headers

            return resp

        f.provide_automatic_options = False
        return update_wrapper(wrapped_function, f)
    return decorator

def login_required(f):
    @wraps(f)
    def wrap(*args, **kwargs):
        if 'logged_in' in session:
            return f(*args, **kwargs)
        else:
            error = "You need to login first"
            return redirect(url_for('loginPage', error=error))
    return wrap

@app.route('/')
def homepage():
    return render_template("main.html", subURL = SUB_URL)

@app.route('/login/', methods=["GET","POST"])
def loginPage():
    error = ''
    if 'error' in request.args:
        error = request.args['error']
    try:
        if request.method == "POST":
            uname = request.form['username']
            upwd = request.form['password']
            valid = checkLogin(uname, upwd)
            if valid:
                Uname = uname.upper()
                session['logged_in'] = True
                session['username'] = Uname
                return redirect(url_for('upload', Uname=Uname))
            else:
                error = 'Invalid login credentials ! Try Again...'

        #return render_template("login.html", error = error)
    except Exception as e:
        flash(e)
    return render_template("login.html", error = error, subURL = SUB_URL)

def checkLogin(uname, upwd):
    allowed = False
```

```
    loginFile = os.path.join(HOME_DIR, LOGINFILE)
    with open(loginFile, 'r') as login:
        for data in login:
            value = data.split(',')
            if uname.lower() == value[0].strip().lower():
                if NETID_LOGIN and not value[1].strip():
                    allowed = netID_login(uname, upwd)
                else:
                    if upwd.strip() == value[1].strip():
                        allowed = True
                break

    return allowed

def netID_login(uname, upwd):
    allowed = False

    args = ['kinit', uname]

    output = executeShell(args, upwd);

    if not output['Error']:
        allowed = True

    return allowed

@app.route('/logout/')
@login_required
def logout():
#     removeUserData(session['username'].lower())
    session.clear()
    error = "You have Successfully Logged Out !"
    return redirect(url_for('loginPage', error = error))

@app.route('/graph/', methods=['GET', 'POST'])
@login_required
def graph():
    Uname = request.args['Uname']
    xaxis = str(request.args['xaxis']).strip()
    yaxis = str(request.args['yaxis']).strip()
    jsonKeys = processJSON(Uname.lower())
    result = getResult(Uname.lower())

    return render_template("graph.html", Uname=Uname, xaxis = xaxis, yaxis=yaxis, jsonKeys =
jsonKeys, result=result, subURL = SUB_URL)

def getResult(uname):
    wrkDir = getWrkDir(uname)
    resultjsonpath = os.path.join(wrkDir, RESULTJSON)
```

```
    result = ''
    if os.path.exists(resultjsonpath):
        jsonoutput = open(resultjsonpath, 'r').readlines()
        result = jsonoutput[0]
    return result

@app.route('/graphdata/', methods=['GET', 'POST'])
@login_required
@crossdomain(origin='*')
def graphdata():
    uname = session['username']
    xaxis = request.args['xaxis']
    yaxis = request.args['yaxis']

    result = getData(uname.lower(), xaxis, yaxis)

    return result

def getData(uname, xaxis, yaxis):
    wrkDir = getWrkDir(uname)
    resultjsonpath = os.path.join(wrkDir, RESULTJSON)
    resultdata = []
    if os.path.exists(resultjsonpath):
        jsonoutput = open(resultjsonpath, 'r').readlines()
        jsonoutput = jsonoutput[0]
        try:
            jsondata = json.loads(jsonoutput)
            xaxis_data = parseJSONData(jsondata, xaxis)
            yaxis_data = parseJSONData(jsondata, yaxis)

            for i in range(max(len(xaxis_data), len(yaxis_data))):
                resultdata.append(str(xaxis_data[i])+','+str(yaxis_data[i]))
        except Exception as e:
            print 'Error in Graph GetData:' +e
            return ''
    head = 'xAxis,yAxis\n'
    resultdata = head + '\n'.join(resultdata)
    return resultdata

def parseJSONData(jsonData, attr):
    data = []
    if jsonData.__class__.__name__ in ('list', 'tuple'):
        for i in range(len(jsonData)):
            data = data + parseJSONData(jsonData[i], attr)
    else:
        try:
            attrList = attr.split(JSONKEYSEP)
            jkey = attrList[0]
            attrList.remove(jkey)
```

```
            attr = '_'.join(attrList)
            if attr == '':
                data.append(jsonData[jkey])
            else:
                data = parseJSONData(jsonData[jkey], attr)
        except Exception as e:
            print 'Error in parseJSONData:' +e
            return ''
    return data

@app.route('/checkThread/', methods=["GET", "POST"])
@login_required
def checkThread():

    uname = request.args['Uname']

    threadInfo = getThreadInfo(uname.lower())
    if threadInfo:
        if threadInfo['status'] == 'finished':
            result = "thread_finished"
        else:
            result = "thread_working"
    else:
        result = "no_thread"

    return result

@app.route('/upload/', methods=["GET", "POST"])
@login_required
def upload():
    result = ''
    query = ''
    runType= ''
    runMode = ''
    optMode = ''
    nodes = ''
    jsonKeys = []

    Uname = request.args['Uname']
    queryList = buildSelectQuery()
    dataList = buildPredefineDataFileList()
    selectQuery = request.args.get('selectQuery', 'Select Query')
    wrkDir = getWrkDir(Uname.lower())
    queryFile = os.path.join(wrkDir, QUERYFILE)
    resultjsonpath = os.path.join(wrkDir, RESULTJSON)
    threadInfoFile = os.path.join(wrkDir, THREADINFOFILE)

    try:
        if request.method == "POST":
```

```
        if 'visualize' in request.form:
            xaxis = request.form['x_axis']
            yaxis = request.form['y_axis']
            return redirect(url_for('graph',Uname=Uname, xaxis = xaxis, yaxis = yaxis))

        if 'loadPrevSession' in request.form:
            if checkThreadFinished(Uname.lower()):#not thread.isAlive():
                params = getThreadInfo(Uname.lower(), True)
                if params:
                    runType = params['runType']
                    runMode = params['runMode']
                    nodes = params['runType']
                    nodes = params['nodes']

                if os.path.exists(queryFile):
                    query = open(queryFile, 'r').read()
                else:
                    query = "No Previous Query Found !"

                if os.path.exists(resultjsonpath):
                    result = open(resultjsonpath, 'r').read()
                    jsonKeys = processJSON(Uname.lower())
                else:
                    result = ""

                if os.path.exists(threadInfoFile):
                    os.remove(threadInfoFile)

            else:
                query = request.form['query']
                runType = request.form['run_type']
                runMode = request.form['run_mode']
                if 'nodes' in request.form:
                    nodes = request.form['nodes']
                optMode = request.form['opt_mode']
                result = "Processing Previous Query !!!"
        else:
            query = request.form['query']
            runType = request.form['run_type']
            runMode = request.form['run_mode']
            if 'nodes' in request.form:
                nodes = request.form['nodes']
            optMode = request.form['opt_mode']

            if query:
                fileList = request.files.getlist("dataFile[]")
                if fileList[0]:
                    wrkDir = getWrkDir(Uname.lower())
                    saveFileList(wrkDir, fileList)
```

```
                dataChoiceList = request.form.getlist("dataChoice[]")
                UserDataChoiceList = request.form.getlist("UserDataChoice[]")
    #             if runMode.lower().strip() == 'distributed':
                if checkThreadFinished(Uname.lower()): #not thread.isAlive():
                    if runMode.lower().strip() == 'distributed':
                        if not fileList[0] and not dataChoiceList and not UserDataChoiceList:
                            result = "Choose/Upload Data Files Again to use Distributed Mode"
                        else:
                            processThreadForDistMode(query, Uname.lower(), runType, runMode, optMode,
                                    nodes, fileList, dataChoiceList, UserDataChoiceList)
                            result = "Query Submitted !!!\nYou can check progress in \n\n" +
HADOOP_CLUSTER_LINK
                    else:
                        result = processQuery(query, Uname.lower(), runType, runMode, optMode,
                                nodes, fileList, dataChoiceList, UserDataChoiceList)
                        jsonKeys = processJSON(Uname.lower())
                else:
                    runMode = 'distributed'
                    result = "Processing Previous Query !!!"

            else:
                result = 'Enter Query !'
        else:
            if(selectQuery in 'Select Query'):
                query = ""
            else:
                query = getSelectQuery(selectQuery)

            threadInfo = getThreadInfo(Uname.lower())
            if threadInfo:
                params = threadInfo['params']
                if params:
                    runType = params['runType']
                    runMode = params['runMode']
                    nodes = params['runType']
                    nodes = params['nodes']

                if os.path.exists(queryFile):
                    query = open(queryFile, 'r').read()
                    query = "/** Loading Previous Query **/\n\n" + query
                if threadInfo['status'] == 'finished':
                    if os.path.exists(resultjsonpath):
                        result = open(resultjsonpath, 'r').read()
                        jsonKeys = processJSON(Uname.lower())
                    os.remove(threadInfoFile)
                else:
                    result = "Processing Previous Query !!!"

    except Exception as e:
```

```
        print e
        os.chdir(HOME_DIR)
        result = e

    userDataList = buildUserDataFileList(Uname.lower())

    return render_template("upload.html", Uname=Uname, result=result, query=query,
selectQuery=selectQuery, queryList=queryList, \
                        dataList = dataList,runT=runType, runM=runMode, optM=optMode, nodes=nodes,
userdataList = userDataList, \
                        jsonKeys = jsonKeys, subURL = SUB_URL)

def processQuery(query, uname, runType, runMode, optMode, nodes, fileList, dataChoiceList,
UserDataChoiceList, threadMode = False):
    global thread_finished
    distMode = False
    holdQuery = query
    param = ''
    if threadMode:
        param = {'runType': runType, 'runMode': runMode,'optMode': optMode, 'nodes': nodes}
        updateThreadInfo(uname, 'started', str(param))

    runArg = chooseRunType(runType, runMode, optMode, nodes)

    if '-dist' in runArg:
        distMode = True

    wrkDir = getWrkDir(uname)

    query = addDumpJson(query, wrkDir)

    mrqlFileDir = os.path.join(wrkDir, QUERYFILE)
    mrqlFileDir_DIST = os.path.join(wrkDir, QUERYFILE_DIST)
    resultjsonpath = os.path.join(wrkDir, RESULTJSON)

    if distMode:
        #Clear all files from HDFS for the user
        removeFileFromHDFS(os.path.join(wrkDir, "*"))


    with open(mrqlFileDir, 'w') as qFile:
        qFile.write(query)
        qFile.close()

    if distMode:
        with open(mrqlFileDir_DIST, 'w') as qFile:
            qFile.write(query)
            qFile.close()
```

```
#     fileList = request.files.getlist("dataFile[]")
    if fileList[0]:
#         saveFileList(wrkDir, fileList)
        if distMode:
            makeProperQueryFileForDistMode(uname, mrqlFileDir, mrqlFileDir_DIST, fileList)

#     dataChoiceList = request.form.getlist("dataChoice[]")
#     UserDataChoiceList = request.form.getlist("UserDataChoice[]")

    if dataChoiceList:
        copySampleQueryData(dataChoiceList, wrkDir)
        saveFileList(wrkDir, dataChoiceList, False)
        if distMode:
            makeProperQueryFileForDistMode(uname, mrqlFileDir, mrqlFileDir_DIST, dataChoiceList)

    if UserDataChoiceList and distMode:
        makeProperQueryFileForDistMode(uname, mrqlFileDir, mrqlFileDir_DIST, UserDataChoiceList)

    os.chdir(wrkDir)

    if distMode:
        if os.path.exists(mrqlFileDir_DIST):
            runArg.append(str(mrqlFileDir_DIST))
        else:
            raise Exception('Choose Data Files for Distributed Mode Query Processing !')
    else:
        runArg.append(str(mrqlFileDir))



    if os.path.exists(resultjsonpath):
        os.remove(resultjsonpath)


    if threadMode:
        updateThreadInfo(uname, 'running', str(param))

    #Sending Query to MRQL
    output = executeShell(runArg)

    if distMode:
        moveFileFromHDFS(resultjsonpath)

    #If resultjson file got created that means no error occurred while processing the query
    if os.path.exists(resultjsonpath):
        jsonoutput = open(resultjsonpath, 'r').read()
#         jsonoutput = jsonoutput[0]
    else:
        if output['Error']:
```

```
            jsonoutput = output['Error']  #output + '\n' +
            print 'Error in MRQL Query Processing: ' + output['Error']
            if threadMode:
                with open(resultjsonpath, 'w') as qFile:
                    qFile.write(jsonoutput)
                    qFile.close()

        else:
            jsonoutput = output['Output']

    if '-trace' in runArg or '-info' in runArg:
        jsonoutput = output['Output'] + '\nResults:\n' + jsonoutput

    with open(mrqlFileDir, 'w') as qFile:
        qFile.write(holdQuery)
        qFile.close()

    os.chdir(HOME_DIR)
#     removeUserData(wrkDir)

    if threadMode:
        updateThreadInfo(uname, 'finished', str(param))
        #thread_finished = True

    return jsonoutput

def processJSON(uname):
    jsonKeys = ''
    wrkDir = getWrkDir(uname)
    resultjsonpath = os.path.join(wrkDir, RESULTJSON)
    if os.path.exists(resultjsonpath):
        jsonoutput = open(resultjsonpath, 'r').read()
#         jsonoutput = jsonoutput[0]

        try:
            jsondata = json.loads(jsonoutput)
            jsonKeys = getJsonKeys(jsondata)
            jsonKeys = jsonKeys.keys()
            if jsonKeys[0] == '':
                jsonKeys = ''
        except Exception as e:
            jsonKeys = ''

    return jsonKeys

def getJsonKeys(jsonData, level = -1, parentKey = ''):
    jsonKeys = dict()
    if jsonData.__class__.__name__ in ('list', 'tuple'):
        if (len(jsonData) > 0):
```

```
            jsonKeys.update(getJsonKeys(jsonData[0], level, parentKey))
    else:
        try:
            jsonKeysList = jsonData.keys()
            level += 1
            for jKey in jsonKeysList:
                nKey = jKey
                if parentKey:
                    nKey = parentKey + JSONKEYSEP + jKey

                jsonKeys.update(getJsonKeys(jsonData[jKey], level, nKey))
        except Exception as e:
            jsonKeys[parentKey] = level

    return jsonKeys

def chooseRunType(runType, runMode, optMode, nodes):
    runArg = []
    distMode = False
    if runType.lower().strip() == 'mapreduce':
        arg = "mrql"
    else:
        if runType.lower().strip() == 'spark':
            arg = "mrql.spark"
        else:
            arg = ""
    if arg:
        arg = MRQL_EXEC_DIR + arg
    runArg.append(arg)
    if runMode.lower().strip() == 'memory':
        arg = ""
    else:
        if runMode.lower().strip() == 'local':
            arg = "-local"
        else:
            if runMode.lower().strip() == 'distributed':
                arg = "-dist"
                distMode = True
            else:
                arg = ""
    runArg.append(arg)
    if distMode:
        arg = "-nodes " + str(nodes)
        runArg.append(arg)

    if optMode.lower().strip() == 'trace':
        arg = "-trace"
        runArg.append(arg)
    else:
```

```
        if optMode.lower().strip() == 'info':
            arg = "-info"
            runArg.append(arg)


    return runArg

def getWrkDir(uname):

    reqDir = os.path.join(WORK_DIR, uname)

    if not os.path.exists(reqDir):
        os.makedirs(reqDir)
#     os.chmod(reqDir, 777)
    return reqDir

#Save DataFile File in Working directory and update user Data File List
def saveFileList(wrkDir, fileList, saveData = True):
    userFileExist = False
    readList = []
    filepath = os.path.join(wrkDir, DATAFILELIST)
    if os.path.exists(filepath):
        userDataFilesList = open(filepath, 'r')
        for fname in userDataFilesList.readlines():
            readList.append(fname.rstrip())
        userDataFilesList.close()
    userDataFilesList = open(filepath, 'a')

    for file in fileList:
        if file:
            if saveData:
                filename = secure_filename(file.filename)
                file.save(os.path.join(wrkDir, filename))
            else:
                filename = file
            userFileExist = True
            if not filename in readList:
                userDataFilesList.write(filename)
                userDataFilesList.write('\n')
    userDataFilesList.close()
    return userFileExist

def copySampleQueryData(dataChoiceList, wrkDir):
    dataLoc = os.path.join(HOME_DIR, DATAFOLDER)
#     dataList = dataChoiceList
    for dataFile in dataChoiceList:
        src = os.path.join(dataLoc, dataFile)
        dst = os.path.join(wrkDir, dataFile)
```

```
      #If file chosen is not from User Uploaded Data Files then Copy Again
#         if not uname.upper() in dataFile:
       copyfile(src, dst)
#         else:
       #If file chosen is from User Uploaded Data Files then rename filename by removing User Name
from it
#             dataChoiceList.remove(dataFile)
#             dataFile = dataFile.replace(uname.upper(), "")
#             dataChoiceList.append(dataFile.strip())


def addDumpJson(query, path):
    result = query.split(";")
    resultdumppath = os.path.join(path, RESULTJSON)

    pos = len(result) - 2
    result[pos] = '\ndumpjson "' + resultdumppath + '" from ' + result[pos]

    return '; '.join(result)

def getSelectQuery(queryName):
    query = ''
    qpath = os.path.join(HOME_DIR, QUERYFOLDER)
    queryName = queryName + '.mrql'
    qpath = os.path.join(qpath, queryName)
    with open(qpath, 'r') as QFile:
        query = QFile.read()

    return query

def buildSelectQuery():
    queryList = []
    qpath = os.path.join(HOME_DIR, QUERYFOLDER)

    dirs = os.listdir( qpath )
    for file in dirs:
        if '~' not in file:
            if '.mrql' in file:
                file = file.replace('.mrql', '')
                queryList.append(file)

    return queryList

def buildPredefineDataFileList():
    dataList = []
    qpath = os.path.join(HOME_DIR, DATAFOLDER)

    dirs = os.listdir( qpath )
    for file in dirs:
```

```
        dataList.append(file)

    return dataList

def buildUserDataFileList(uname):
    wrkDir = getWrkDir(uname)
    filepath = os.path.join(wrkDir, DATAFILELIST)
    dataList = []
    filelist = ''
    if os.path.exists(filepath):
        filelist = open(filepath, 'r')
        for file in filelist.readlines():
            dataList.append(file.rstrip())

    return dataList


def removeUserData(uname):
    WORK_DIR = tempfile.gettempdir()
    reqDir = os.path.join(WORK_DIR, uname)
    if os.path.exists(reqDir):
        shutil.rmtree(reqDir)

def getHash(buf):
    hasher = hashlib.md5()
    hasher.update(buf)
    return hasher.hexdigest()

def executeShell(runArg, data = ''):
    proc = subprocess.Popen(runArg,
                    stdin=subprocess.PIPE,
                    stdout=subprocess.PIPE,
                    stderr=subprocess.PIPE)

    if data:
        proc.stdin.write(data)
    proc.stdin.flush()
    output, stderr = proc.communicate()

    result = {'Output': output, 'Error': stderr}

    return result

def makeProperQueryFileForDistMode(uname, queryFileLoc, distQueryFileLoc, fileList):
    wrkDir = getWrkDir(uname)
    tmpQuery = str(open(queryFileLoc, 'r').read())
    for file in fileList:
        try:
            tmpQuery = tmpQuery.replace(file.filename, wrkDir + '/' + file.filename)
```

```
            moveFileToHDFS(uname, file.filename)
        except Exception as e:
            tmpQuery = tmpQuery.replace(file, wrkDir + '/' + file)
            moveFileToHDFS(uname, file)

    with open(distQueryFileLoc, 'w') as qFile:
        qFile.write(tmpQuery)
        qFile.close()

def moveFileToHDFS(uname, filename):
    wrkDir = getWrkDir(uname)
    filename = os.path.join(wrkDir, filename)


#     hdfs_dir = str(HDFS_DIR + '/' + uname)

    args = HADOOP_EXEC_DIR + ['fs', '-mkdir', '-p', wrkDir]

    try:
        output = executeShell(args)

        args = HADOOP_EXEC_DIR + ['fs', '-put', filename, wrkDir]
        output = executeShell(args)
#         if not output['Output']:
#             raise Exception('Error in HDFS File Transfer:' + output['Output'])
#         else:
#             raise Exception('Error in HDFS DIR Creation:' + output['Output'])
    except Exception as e:
        print str(e)

def checkThreadFinished(uname):
    result = False
    threadInfo = getThreadInfo(uname)
    if threadInfo:
        if threadInfo['status'] == 'finished':
            result = True
    else:
        result = True

    return result

def updateThreadInfo(uname, status, param = ''):
    wrkDir = getWrkDir(uname)
    threadInfoFile = os.path.join(wrkDir, THREADINFOFILE)
    with open(threadInfoFile, 'w') as tFile:
        tdata = {'status': status, 'params' : param}
        tFile.write(str(tdata))
        tFile.close()
```

```
def getThreadInfo(uname, paramsOnly = False):
    wrkDir = getWrkDir(uname)
    threadInfoFile = os.path.join(wrkDir, THREADINFOFILE)
    if os.path.exists(threadInfoFile):
        tdata = open(threadInfoFile, 'r').read()
        tdata = ast.literal_eval(tdata)
        result = ast.literal_eval(tdata['params'])
        if not paramsOnly:
            tdata['params'] = result
            result = tdata
    else:
        result = ''
    return result

def moveFileFromHDFS(path):
    args = HADOOP_EXEC_DIR + ['fs', '-get', path]
    output = executeShell(args)

def removeFileFromHDFS(path):
    args = HADOOP_EXEC_DIR + ['fs', '-rm', path]
    output = executeShell(args)

def processThreadForDistMode(query, uname, runType, runMode, optMode, nodes, fileList,
dataChoiceList, UserDataChoiceList):

    #thread_finished = False
    thread = Thread(target = processQuery, args = (query, uname, runType, runMode, optMode, nodes,
                                        fileList, dataChoiceList, UserDataChoiceList, True))
    thread.start()

@app.errorhandler(404)
def pageNotFound(e):
    return render_template("404.html")

@app.errorhandler(405)
def methodNotFound(e):
    return render_template("404.html")

@app.before_request
def make_session_permanent():
    session.permanent = True
    app.permanent_session_lifetime = timedelta(minutes=SESSION_ALIVE_TIMEOUT_MINUTES)


port = os.getenv('VCAP_APP_PORT','80')
if __name__ == '__main__':

    app.config['SESSION_TYPE'] = 'filesystem'
```

```
    sess.init_app(app)

#     app.run(debug=True) #Dev Env

    app.run(host='0.0.0.0/mrql', port=int(port),debug=True)     #Test Env & Prod Env
```

Screenshot of main query page below

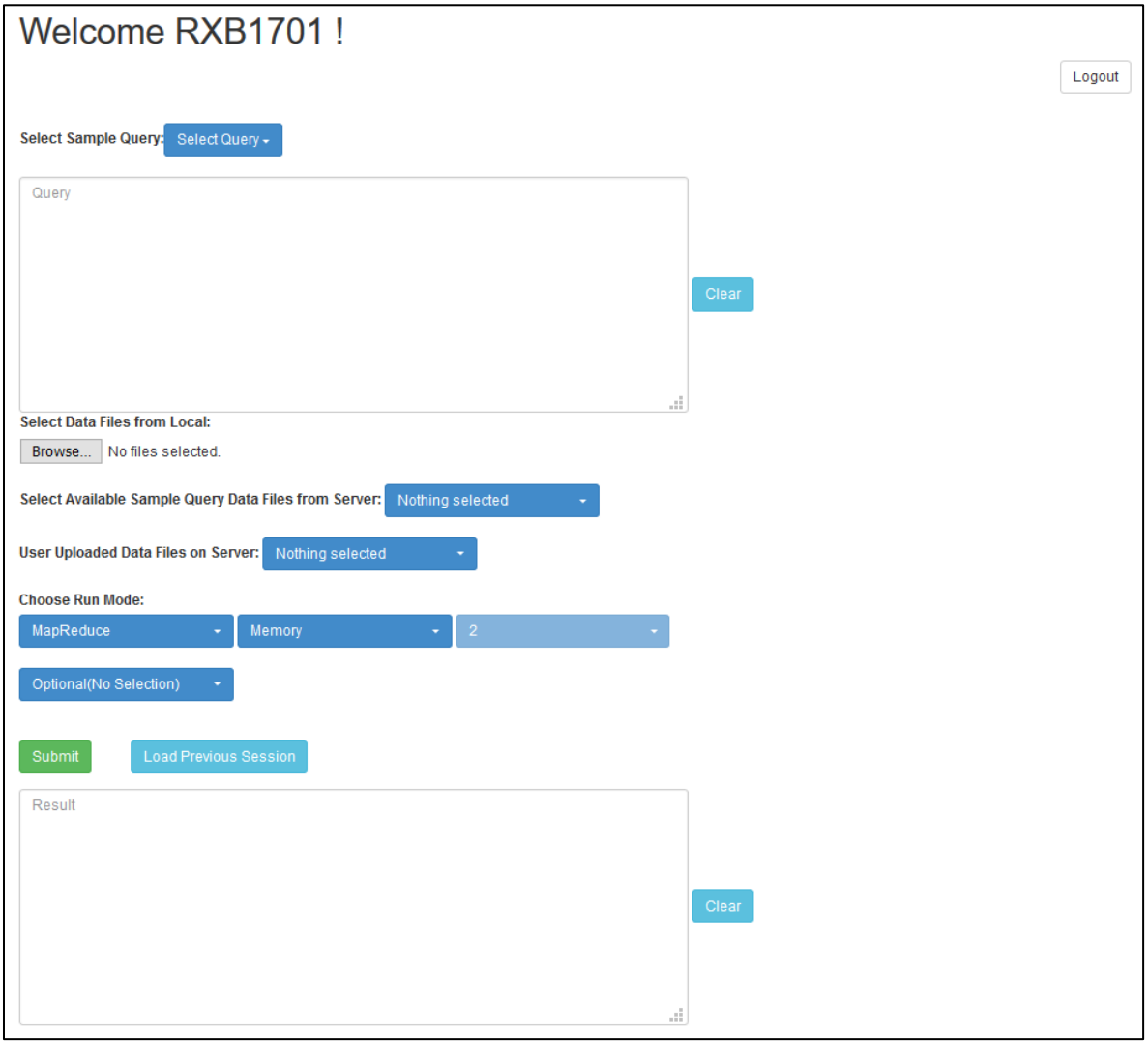

Figure 5-2 Web Application

Attribute information from JSON output is extracted by the web application so that user is able to choose X-axis and Y-axis for the visualization. Assumption made here is that the Attributes value will be uniform throughout the json output. This is done using the function *getJsonKeys*. Once the attributes are fetched, the Web application allows user to choose any of the attribute as X-axis and Y-axis as shown below

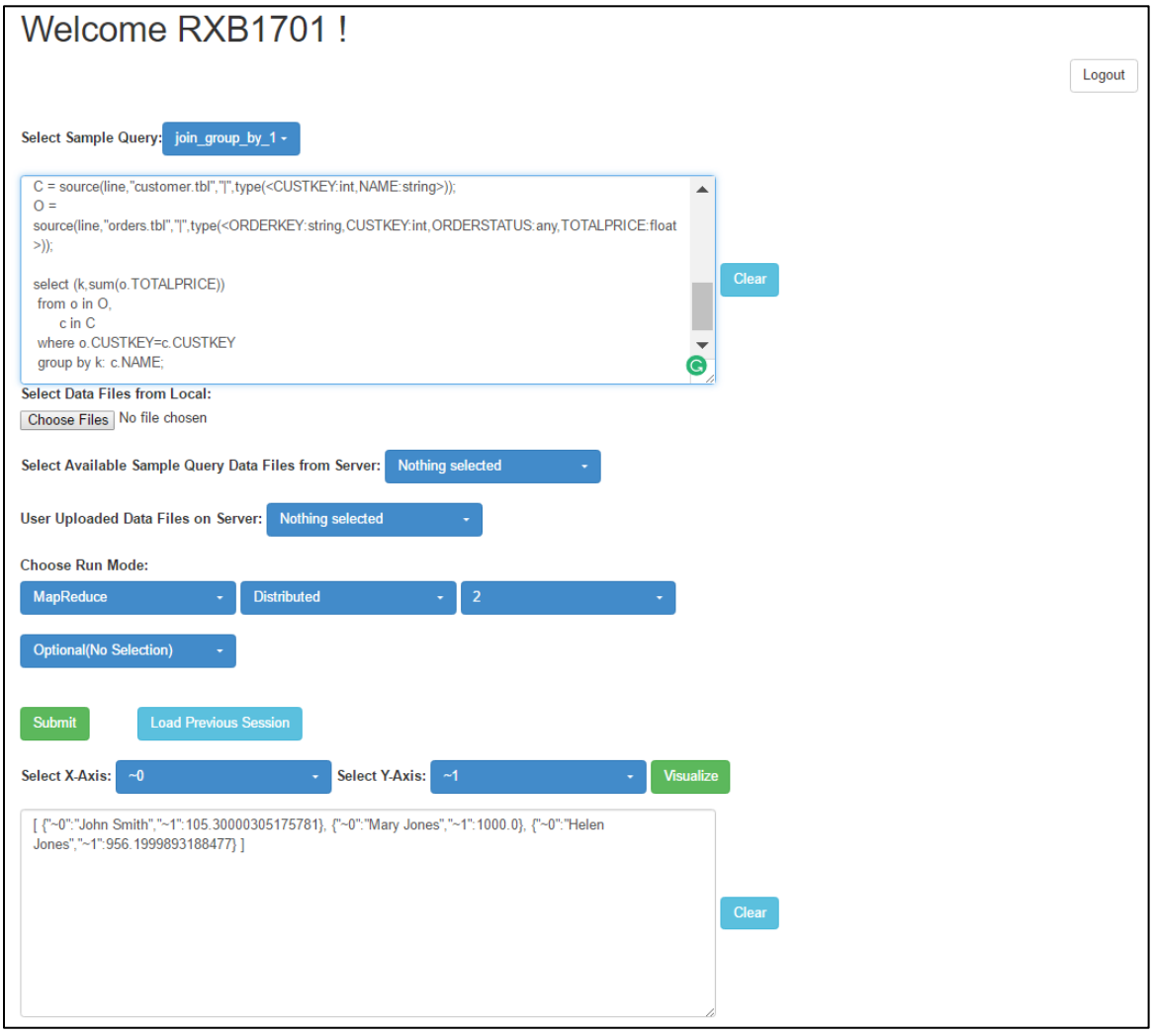

Figure 5-3 Query and Result

On click of dropdown to any of the axis, the parsed keys from json output is displayed for user to choose from it as shown below

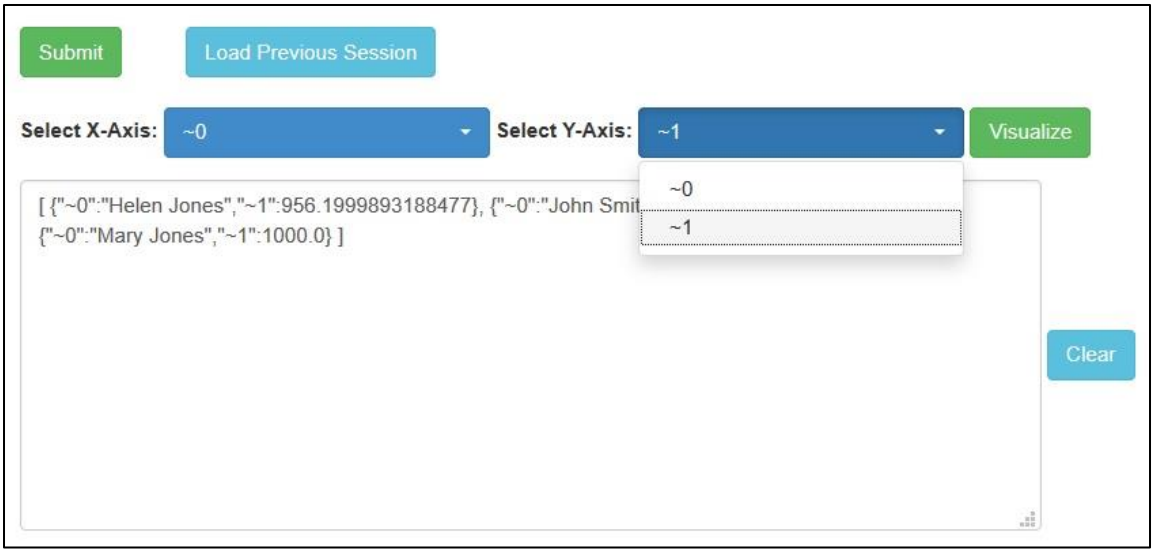

Figure 5-4 Json Keys Selection

5.5. Phase Two: D3JS for Visualization

D3JS is used for visualization of json output coming from MRQL. Based on what key the user selects as X-Axis and Y-Axis, the web application parses the json output to fetch data based on selected keys. The data is formed by function *getData* and *parseJSONData* in server side which checks the key values to handle nested keys. The function generates comma separated two columns data which is used by D3JS. So, D3JS calls a get request to the web server, the web server builds the data and sends as response to the get request then D3JS processes the data to form the desired graph thus acting like a framework which accepts two column data to produce visualization. The code for D3JS is below

```
// The base endpoint to receive data from
var URL_BASE = getURLBASE();

var xScale_Ordinal = false;
var yScale_Ordinal = false;
var ticksCount = 10;

var margin = {top: 20, right: 30, bottom: 100, left: 80};
var width = 800 - margin.left - margin.right;
var height = 600 - margin.top - margin.bottom;

// Whitespace on either side of the bars in units of minutes
var binMargin = 0.01;
```

```
 // x scale
var x = d3.scale.linear()
    .range([0,  width]);
//    .domain([0, 25]);

var xAxis = d3.svg.axis()
    .scale(x)
    .orient("bottom")
    .ticks(10);

//y scale
var y = d3.scale.linear()
    .range([height, 0]);
var yAxis = d3.svg.axis()
    .scale(y)
    .orient("left")
    .ticks(10);

var svg = d3.select("body").append("svg")
    .attr("width", width + margin.left + margin.right)
    .attr("height", height + margin.top + margin.bottom)
  .append("g")
    .attr("transform",
      "translate(" + margin.left + "," + margin.top + ")");

//For Tooltip
var div = d3.select("body").append("div")
    .attr("class", "tooltip")
    .style("opacity", 0);

// x axis
svg.append("g")
      .attr("class", "x axis")
      .attr("transform", "translate(0," + height + ")")
      .call(xAxis)
    .append("text")
      .attr("class", "xaxis_label")
      .text("X-Axis")
      .attr("dy", "3em")
      .attr("text-align", "center")
      .attr("x", width / 2 - margin.right - margin.left);

// y axis
svg.append("g")
    .attr("class", "y axis")
    .call(yAxis)
  .append("text")
    .attr("class", "yaxis_label")
    .attr("transform", "rotate(-90)")
```

```
    .attr("x", -height / 2)
   .attr("dy", "-3em")
   .text("Y-Axis");

function getURLBASE(){
            var urlName = window.location.origin + window.location.pathname;
            urlName = urlName.replace("graph", "graphdata");
            return urlName;
}


// Return url to recieve data
function update_url() {
  var x_axis = document.getElementById("x_axis").value;
  var y_axis = document.getElementById("y_axis").value;

  return URL_BASE +
       "?xaxis=" + x_axis +
       "&yaxis=" + y_axis;
}

// Convert csv data to correct datatypes
function type(d) {
  //Check if xAxis data is Numerical
  if(!isNaN(d.xAxis)){
       d.xAxis = +d.xAxis;
            xScale_Ordinal = false;
  }
  else{
    d.xAxis = d.xAxis;
    xScale_Ordinal = true;
  }

  //Check if yAxis data is Numerical
  if(!isNaN(d.yAxis)){
       d.yAxis = +d.yAxis;
            yScale_Ordinal = false;
  }
  else{
    d.yAxis = d.yAxis;
    yScale_Ordinal = true;
  }
  return d;
}


function build_scale(){
//For String datatype, Ordinal scale needs to be used
   if (xScale_Ordinal){
```

```
      x = d3.scale.ordinal().rangeRoundBands([0, width], .05);
   }
   else{
     x = d3.scale.linear().range([0,  width]);
   }

   if (yScale_Ordinal){
      y = d3.scale.ordinal().rangeRoundBands([height, 0], .05);
   }
   else{
     y = d3.scale.linear().range([height, 0]);
   }
}

function make_graph() {
  var graphType = document.getElementById("graph_type").value;
  if (graphType == "line"){
              make_scatterPlots();
  }
  else{
              make_barGraph();
  }
}

function make_barGraph() {
  url = update_url();
  var xaxisLabel = document.getElementById("x_axis").value;
  var yaxisLabel = document.getElementById("y_axis").value;
  d3.csv(url, type, function(error, data) {

     build_scale();

     xAxis = d3.svg.axis()
    .scale(x)
    .orient("bottom")
    .ticks(ticksCount);

     yAxis = d3.svg.axis()
    .scale(y)
    .orient("left")
    .ticks(ticksCount);

     var xMax = d3.max(data, function(d) { return d.xAxis; });

     if (xScale_Ordinal){
       x.domain(data.map(function(d) { return d.xAxis; }));
      }
      else{
            x.domain([0, d3.max(data, function(d) { return d.xAxis; })*1.2 ]);
```

```
  }

  if (yScale_Ordinal){
    y.domain(data.map(function(d) { return d.yAxis; }));
  }
  else{
        y.domain([0, d3.max(data, function(d) { return d.yAxis; })*1.2]);
  }


//Redraw X-Axis Scale
svg.selectAll("g.x.axis")
  .transition()
  .call(xAxis);

//Redraw X-Axis Label
svg.selectAll("g.x.axis")
   .select("text.xaxis_label")
   .transition()
   .text(xaxisLabel)
   .attr("dy", "3em")
   .attr("text-align", "center")
   .attr("x", width / 2 - margin.right - margin.left);

//Redraw Y-Axis Scale
svg.selectAll("g.y.axis")
  .transition()
  .call(yAxis);

//Redraw Y-Axis Label
svg.selectAll("g.y.axis")
   .select("text.yaxis_label")
   .transition()
   .attr("class", "yaxis_label")
   .attr("transform", "rotate(-90)")
   .attr("x", -height / 2)
   .attr("dy", "-3em")
   .text(yaxisLabel);

svg.selectAll(".mline").remove();
svg.selectAll("circle").remove();

var bars = svg.selectAll(".bar")
  .data(data, function(d) { return d.xAxis; });

bars.transition(1000)
  .attr("y", function(d) { return  y(d.yAxis); } )
  .attr("height", function(d) { return height - y(d.yAxis); } );
```

```
  bars.enter().append("rect")
    .attr("class", "bar")
    .attr("x", function(d) { return x(d.xAxis); })  //-0.25

    if (xScale_Ordinal){
    bars.attr("width", x.rangeBand())
    }
    else{
      var calcWidth = xMax/(ticksCount*2);
          if (calcWidth < 2 * binMargin){
            bars.attr("width", x(1 - 2 * binMargin))
          }
          else{
            bars.attr("width", x((xMax/(ticksCount*2)) - 2 * binMargin))
          }
    }

    bars.attr("y", height)
    .attr("height", 0)
    .attr("y", function(d) { return y(d.yAxis); })

    if (yScale_Ordinal){
           bars.attr("height", function(d) { return y.rangeBand(); })
    }
    else{
           bars.attr("height", function(d) { return height - y(d.yAxis); })
    }

    bars.on("mouseover", function(d) {
         div.transition()
            .duration(200)
            .style("opacity", .9);
         div.html(xaxisLabel+":"+d.xAxis + "<br/>"  + yaxisLabel+":"+d.yAxis)
            .style("left", (d3.event.pageX) + "px")
            .style("top", (d3.event.pageY - 28) + "px");
         })
      .on("mouseout", function(d) {
         div.transition()
            .duration(500)
            .style("opacity", 0);
      });


    bars.exit()
      .transition(1000)
       .attr("y", height)
       .attr("height", 0)
      .remove();
});
```

```
}

function make_scatterPlots(){
   url = update_url();
   var xaxisLabel = document.getElementById("x_axis").value;
   var yaxisLabel = document.getElementById("y_axis").value;

   // Get the data
   d3.csv(url, type, function(error, data) {

    build_scale();

    xAxis = d3.svg.axis()
   .scale(x)
   .orient("bottom")
   .ticks(ticksCount);

    yAxis = d3.svg.axis()
   .scale(y)
   .orient("left")
   .ticks(ticksCount);

   // Scale the range of the data
   if (xScale_Ordinal){
      x.domain(data.map(function(d) { return d.xAxis; }));
    }
   else{
            x.domain([0, d3.max(data, function(d) { return d.xAxis; })*1.2 ]);
    }

   if (yScale_Ordinal){
      y.domain(data.map(function(d) { return d.yAxis; }));
    }
   else{
      y.domain([0, d3.max(data, function(d) { return d.yAxis; })*1.2 ]);
    }

   //Redraw X-Axis Scale
   svg.selectAll("g.x.axis")
     .transition()
     .call(xAxis);

   //Redraw X-Axis Label
   svg.selectAll("g.x.axis")
      .select("text.xaxis_label")
      .transition()
      .text(xaxisLabel)
      .attr("dy", "3em")
      .attr("text-align", "center")
```

```
   .attr("x", width / 2 - margin.right - margin.left);

//Redraw Y-Axis Scale
svg.selectAll("g.y.axis")
  .transition()
  .call(yAxis);

//Redraw Y-Axis Label
svg.selectAll("g.y.axis")
   .select("text.yaxis_label")
   .transition()
   .attr("class", "yaxis_label")
   .attr("transform", "rotate(-90)")
   .attr("x", -height / 2)
   .attr("dy", "-3em")
   .text(yaxisLabel);

//Calculate X-Axis Offset for String Value Ploting
var xoffset = 0;
if (xScale_Ordinal){
        var maxVal = d3.max(data, function(d){return x(d.xAxis)});
        var minVal = d3.min(data, function(d){return x(d.xAxis)});
        var valArray = d3.values(data, function(d){return x(d.xAxis)});
        varArray = valArray.length - 1;
        xoffset = (maxVal - minVal)/(varArray*2); //42.5
}
//Calculate Y-Axis Offset for String Value Ploting
var yoffset = 0;
if (yScale_Ordinal){
    var maxVal = d3.max(data, function(d){return y(d.yAxis)});
    var minVal = d3.min(data, function(d){return y(d.yAxis)});
    var valSize = d3.values(data, function(d){return y(d.yAxis)});
    valSize = valSize.length - 1;
    yoffset = (maxVal - minVal)/(valSize*2);
}


// Define the line
var valueline = d3.svg.line()
                        .x(function(d) { return x(d.xAxis)+xoffset; })
                        .y(function(d) { return y(d.yAxis)+yoffset; });

        svg.selectAll(".bar").remove();
        svg.selectAll(".mline").remove();
        svg.selectAll("circle").remove();

var chart = svg.selectAll(".mline")
                    .data("1");
```

```
    chart.enter()
            .append("path")
        .attr("class", "mline");

    if (!xScale_Ordinal){
         chart.attr("d", valueline(data.sort(function(a,b) { return d3.ascending(a.xAxis,b.xAxis) })));
    }
    else{
             chart.attr("d", valueline(data));
    }

    // Add the scatterplot
    svg.selectAll("circle")
       .data(data)
      .enter().append("circle")
       .attr("r", 3.5)
       .attr("cx", function(d) { return x(d.xAxis)+xoffset; })
       .attr("cy", function(d) { return y(d.yAxis)+yoffset; })
             .on("mouseover", function(d) {
           div.transition()
              .duration(200)
              .style("opacity", .9);
           div.html(xaxisLabel+":"+d.xAxis + "<br/>"  + yaxisLabel+":"+d.yAxis)
              .style("left", (d3.event.pageX) + "px")
              .style("top", (d3.event.pageY - 28) + "px");
           })
       .on("mouseout", function(d) {
           div.transition()
              .duration(500)
              .style("opacity", 0);
       });
    });
}

make_graph();
```

Based on what graph type is selected by the user, D3JS is used to produce bar graph as well as scatter plot graph. Bar Graph is shown below

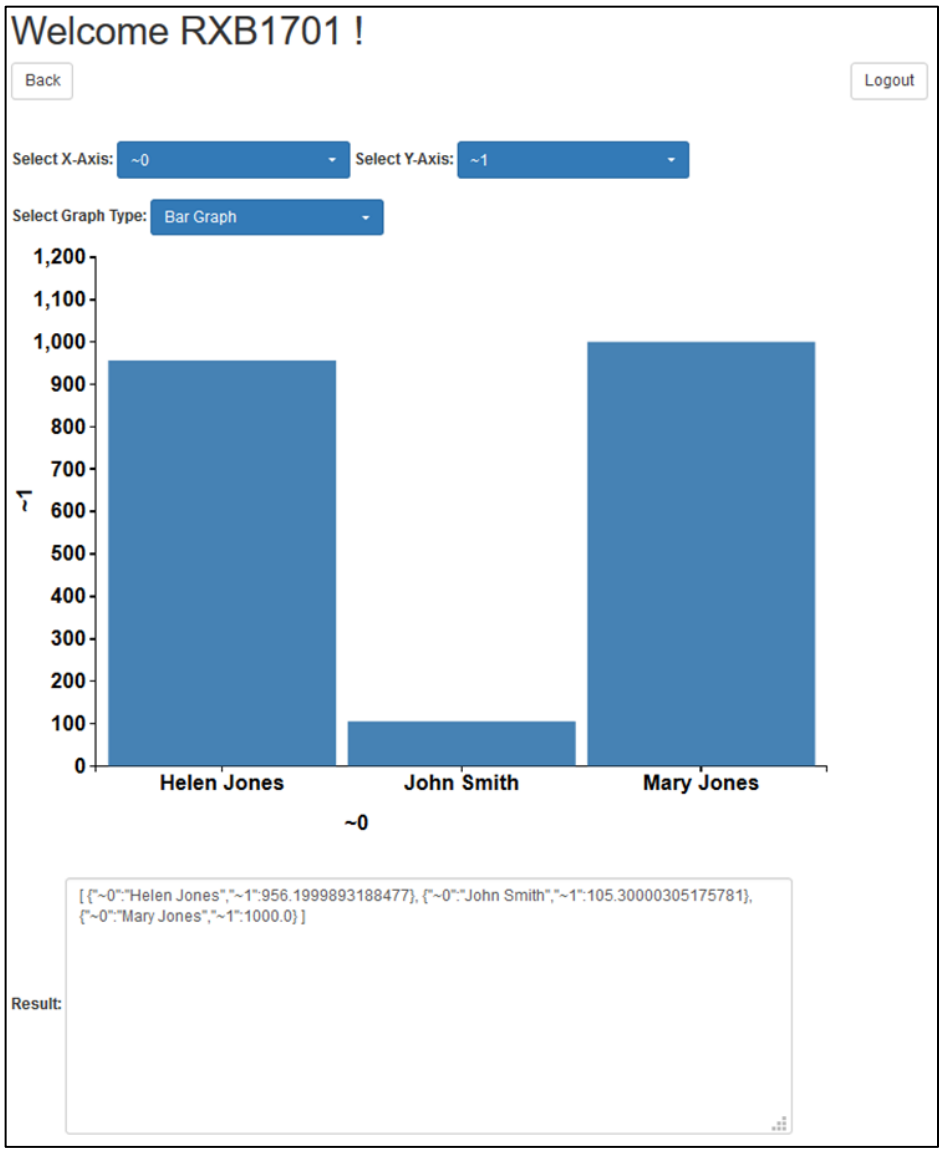

Figure 5-5 Bar Graph

Scatter Plot Graph below

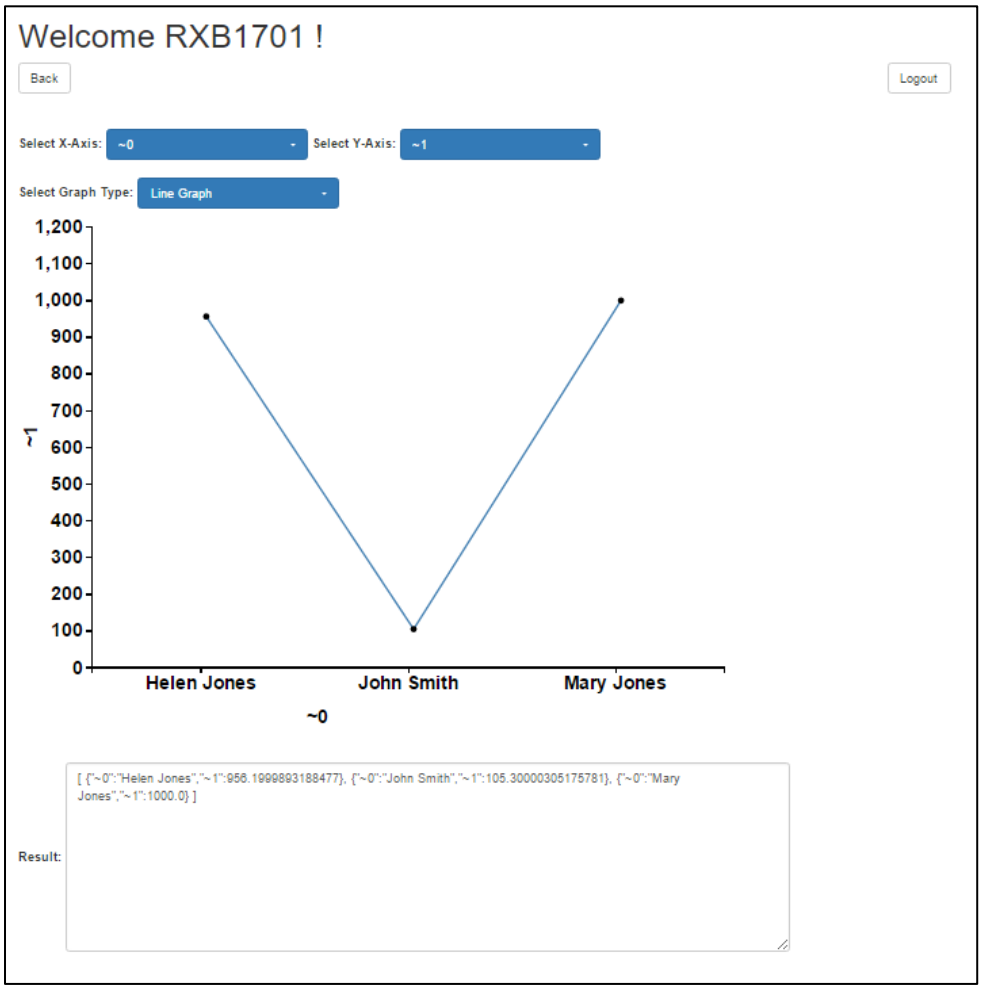

Figure 5-6 Scatter Plot

5.6. Phase Two: Authentication

Web application allows local and centralized authentication. Centralized authentication is achieved using Kerberos network authentication protocol. It works on the basis of 'tickets' to allow nodes communicating over a non-secure network to prove their identity to one another in a secure manner. It is aimed primarily for a client–server model and it provides mutual authentication—both the user and the server verify each other's identity. Kerberos protocol messages are protected against eavesdropping and replay attacks. It can be used by configuring the krb5.conf file with active directory information.

5.7. Phase Two: Features

The web application has following features

- The web application allows users to write MRQL queries in the query text box.
- The query should include only data file name and not the entire file path if the query is using any file for analysis.
- User has option of choosing any of predefine queries by selecting any query list from drop down menu.
- It allows users to either upload multiple files from local or selecting available data files in server.
- User has the option of executing his query in ‘MapReduce’or ‘Spark’ query processing mode.
- After selecting the query processing mode, user can choose to run it either in ‘memory’, ‘local’ or ‘distributed’ mode.
- For distributed mode user has option of specifying number of nodes from drop down menu.
- There is optional drop down menu for giving additional commands like ‘trace’ or ‘info’.
- The result json output is displayed in the result text box.
- User get option to choose keys of result json to be X-Axis or Y-Axis based on their requirement for visualization.
- User has option to choose from bar graph or line scatter plot graph.

The web application provides following benefits:

- Allows execution on MapReduce, Spark, Hama, Flink distributed platforms
- No need to write the Mapper and Reducer to perform Big Data Analysis
- MRQL is independent of underlying distributed platform
- Eliminates the need to learn the individual language syntax of MapReduce, Spark, Hama, Flink
- Visual representation of the query output
- Centralized and localized authentication

Chapter 6

Setup And Installation

Here setup to run Web Application and MRQL is covered. Hadoop installation is assumed to be present and hence will not be covered. This setup and installation is done on Ubuntu 14.04 LTS operating system on Amazon Web Service EC2 instance.

6.1. Required libraries or essentials

Make sure the OS as essential libraries installed along with Java.

```
sudo apt-get update
sudo apt-get install build-essential
sudo apt-get install zlib1g-dev
sudo apt-get install libssl-dev
sudo apt-get install default-jdk
```

6.2. Apache Web Server

Apache is the most commonly used Web server on Linux systems. Web servers are used to serve Web pages requested by client computers.

```
sudo apt-get install apache2
```

6.3. Python

Local installation of python is shown here so that it does not disrupts any pre-installed version of python used by existing applications. Python 2.7.12 version is used for deploying the server script.

```
mkdir ~/Python_2.17.12
mkdir ~/Setups
cd ~/Setups
wget https://www.python.org/ftp/python/2.7.12/Python-2.7.12.tgz
tar -zxvf Python-2.7.12.tgz
cd Python-2.7.12
```

Python setup needs to configured to specify the exact installation location

```
./configure --enable-shared --prefix=/home/${USER}/Python_2.7.12 LDFLAGS="-Wl,--rpath=/usr/local/lib"
```

Use *altinstall* instead of install to avoid changing symbolic links of system Python and man pages.

```
clean make
make altinstall
```

Once installation of python completes, we need to install python pip which allow easy installation of other python libraries.

```
cd ~/Setups
wget https://bootstrap.pypa.io/get-pip.py
```

```
~/Python_2.7.12/bin/python2.7 get-pip.py
```

6.4. Mod Wsgi

The mod_wsgi package provides an Apache module that implements a WSGI compliant interface for hosting Python based web applications on top of the Apache web server. Installation of mod_wsgi can be performed in two ways. First way is to install it as apache module

```
sudo apt-get install libapache2-mod-wsgi
```

Second way is to install as python package

```
~/Python_2.7.12/bin/pip install mod_wsgi
```

This will get the library *mod_wsgi-py27.so* required for mod_wsgi.

6.5. Flask

Flask is a micro web framework written in Python and based on the Werkzeug toolkit and Jinja2 template engine. Flask is called a micro framework because it does not require particular tools or libraries. Flask supports extensions that can add application features as if they were implemented in Flask itself. For using sessions Flask-Session extension is used.

```
~/Python_2.7.12/bin/pip install flask
~/Python_2.7.12/bin/pip install Flask-Session
```

6.6. MRQL

MRQL is a query processing and optimization system for large-scale, distributed data analysis, built on top of Apache Hadoop, Hama, Spark, and Flink. The MRQL query language is powerful enough to express most common data analysis tasks over many forms of raw in-situ data, such as XML and JSON documents, binary files, and CSV documents.

```
cd ~/Setups
wget http://www-us.apache.org/dist/incubator/mrql/apache-mrql-0.9.6-incubating/apache-mrql-0.9.6-incubating-bin.tar.gz
tar -zxvf apache-mrql-0.9.6-incubating-bin.tar.gz
mv apache-mrql-0.9.6-incubating ~/
cd ~/apache-mrql-0.9.6-incubating
mvn clean install
```

Once installation is done, you to configure the Hadoop environments in *~/apache-mrql-0.9.6-incubating/conf/mrql-env.sh* file.

6.7. Kerberos

Kerberos is a computer network authentication protocol that works on the basis of 'tickets' to allow nodes communicating over a non-secure network to prove their identity to one another in a secure manner. It primarily aims at a client–server model and provides mutual authentication i.e. both the user and the server verify each other's identity. Kerberos protocol messages are protected against eavesdropping and replay attacks. It can be used by configuring the krb5.conf file with active directory information.

```
sudo apt-get install krb5-user
```

6.8. Hosting Configuration

Create mrql folder in hosting location like */var/www/html/mrql*. Move the wb application files into the mrql folder. Create a mrql.conf which will contain configuration required for hosting the web application. Following should be in mrql.conf file. Place the conf file in the apache hosting configuration directory.

```
vim mrql.conf

# Conf file for MRQL Web Application


LoadModule wsgi_module /export/home/rohitbhawal/Python_2.7.12/lib/python2.7/site-packages/
mod_wsgi/server/mod_wsgi-py27.so


WSGIPythonHome /export/home/rohitbhawal/Python_2.7.12


WSGIDaemonProcess mrqlserver user=rohitbhawal group=rohitbhawal threads=5 home=/var/w
ww/html/mrql

WSGIScriptAlias /mrql /var/www/html/mrql/mrql.wsgi


WSGISocketPrefix /etc/httpd/run/wsgi


<Directory /var/www/html/mrql>
        WSGIProcessGroup mrqlserver
        WSGIApplicationGroup %{GLOBAL}
                Options All
        Order deny,allow
        Allow from all
</Directory>
```

Chapter 7

Future Work

Some of the future work that can be done are

- **XML Output**: Enable MRQL to generate standard XML output which follows W3C's XML 1.0 Specification thus allowing more external systems to be integrated.

- **Result Comparison**: Currently the web application only shows single query result and does visualization it. This can be enhanced so that the application holds more result and then do a comparison visualization between the result.

- **ML Module**: With the power of big data analysis, support of Machine Learning algorithms on the result to determine which models is suitable for the data can be added to web application.

- **Independent D3 Module**: Currently using D3JS a framework is created which accepts two-column data to perform visualization but it now integrated to just display json result. This can be made independent so that it able to accept any two-column data given b user.

- **Support More Graph Types**: Currently only scatter plot and bar graph is supported but more graphs can be added to perform different kind of visualization.

Chapter 8

Summary And Conclusion

Apache MRQL is powerful SQL-like language which translates the given queries into mapper and reducer code for Hadoop framework to do required data analysis. It helps user to define any data analysis to be done on data set in the form of query which can be easily understood. It removes the overhead from user of writing separate mapper and reducer code for doing data analysis. Since it is running on top of Hadoop it gets all the benefits of parallel and distributed computing on large cluster of commodity hardware. Allowing MRQL to be used as web application makes it use as Platform as a Service (PaaS) thus proving big data analysis. This also highlights, how powerful the tool can be.

In this thesis, I got to learn about, how actually a programing language works and gave me a good insight on the architecture and working of MRQL. Gained overview of Map Reduce programing model with Hadoop, web application development and use of D3JS library to create your own visualization framework.

References

[1] http://petapylon.com/the-big-data-paradigm-shift/

[2] https://en.wikipedia.org/wiki/Big_data

[3] https://www.sas.com/en_us/insights/big-data/what-is-big-data.html#

[4] https://en.wikipedia.org/wiki/Apache_Hadoop

[5] http://www.glennklockwood.com/data-intensive/hadoop/overview.html

[6] https://mrql.incubator.apache.org/

[7] https://en.wikipedia.org/wiki/Apache_Hive

[8] http://events.linuxfoundation.org/sites/events/files/slides/mrql-apachecon15.pdf

[9] https://wiki.apache.org/mrql/LanguageDescription

[10] http://blog.cloud-elements.com/json-better-xml

[11] https://d3js.org/

[12] http://www.bogotobogo.com/Hadoop/BigData_hadoop_Install_on_ubuntu_single_node_cluster.php

[13] https://en.wikipedia.org/wiki/Kerberos_(protocol)

[14] https://en.wikipedia.org/wiki/Lexical_analysis

[15] https://modwsgi.readthedocs.io/en/develop/

[16] https://pypi.python.org/pypi/mod_wsgi

[17] https://pythonprogramming.net/

[18] http://flask.pocoo.org/

[19] https://en.wikipedia.org/wiki/XML

[20] https://hadoop.apache.org/docs/current/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html

[21] http://www.itshared.org/2015/03/hadoop-and-mapreduce.html

Biographical Information

Rohit Bhawal joined the University of Texas at Arlington in fall 2015. He received his Bachelor's Degree in Computer Engineering from University of Mumbai, Mumbai in 2012. He worked as a Software Engineer with e-Emphasys Systems Pvt. Ltd., Mumbai in India until July 2015 where he was responsible for development related to business process automation, integration and reporting in ERP domain. After this he decided to pursue his Master's Degree in Computer Science at University of Texas at Arlington. He works as a System Administrator for Computer Science Engineering (CSE) department of University of Texas at Arlington since September 2016.

His current research interests are in the areas of software development, big data technologies, cloud computing, data mining, machine learning and DevOps.