EAST WEST UNIVERSITY

Developing a User-friendly Tool for Executing Queries over Hadoop Framework

Submitted by:

Tahira Bishwas Anny

ID: 2012-1-60-006

MD. Minhazur Rahman

ID: 2012-1-60-017

Supervised by:

Dr. Mohammad Rezwanul Huq

Assistant Professor

Department of Computer Science and Engineering

East West University

A project submitted in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science and Engineering to the Department of Computer Science and Engineering

August 2016

Abstract

In Relational Database Management System, we always use query for retrieving any information or storing information. But now-a-days, as the data volumes are increasing in every second, it's difficult to keep regular update of this data as a form of relational database. So we present a tool which can deal with huge volume of unstructured data. There is no need to write any traditional query for this tool. It works over any numbers of flat files for executing queries. It does not directly ask for query from the user, rather it gives the user different menu base options and conditions. Any user can access the tool very easily. It does not require any previous knowledge about RDBMS or any other programming knowledge. The main goal of our project is to develop that user friendly tool using which without having any knowledge of the used framework of the developed tool, a particular user can have the benefit of executing queries.

Declaration

We hereby declare that, this project was done under CSE497 and has not been submitted elsewhere for requirement of any degree or for any reason except for publication.

Tahira Bishwas Anny

ID: **- 2012-1-60-006**

Department of Computer Science and Engineering

East West University

MD. Minhazur Rahman

ID: **- 2012-1-60-017**

Department of Computer Science and Engineering

East West University

Letter of Acceptance

This thesis is submitted by Tahira Bishwas Anny, Id: 2012-1-60-006 &MD. Minhazur Rahman, Id: 2012-1-60-017 to the Department of Computer Science and Engineering, East West University, Dhaka Bangladesh is accepted as satisfactory for the partial fulfillment of the requirement for the degree of Bachelor of Science in Computer Science and Engineering on August10th, 2016.

1. ___________________

Dr. Mohammad Rezwanul Huq

Assistant Professor (Supervisor)

Department of Computer Science and Engineering

East West University, Dhaka, Bangladesh

2. _______________________

Dr. Md. Mozammel Huq Azad Khan

Professor & Chairperson

Department of Computer Science and Engineering

East West University, Dhaka, Bangladesh

Acknowledgement

First of all, we would like to thank Almighty Allah for giving us the strength and patience.

We would not have been able to successfully complete this thesis without the support of supervisor during past four months. My sincere thanks to Dr. Mohammad Rezwanul Huq. He has been a source of inspiration for our throughout the process of the research and writing. His feedback and insights were always valuable, and never went unused.

Our deep gratefulness to all of our teachers, who have trained us at East West University. Their wonderful teaching methods improved our knowledge of the individual subject and enabled us to complete our studies in time.

In this acknowledgment would not complete without thanking our parents. They are supported us throughout our university study, we can’t express our appreciation enough. We hope this achievement will cheer them up during these demanding times.

Table of Contents

Contents

List of Figures

Chapter 1

1.1 Introduction

We live in a world increasingly driven by data. From the very beginning of software development, we have been using Relational Databases Management System, in form of SQL. Now RDBMS is good for updating small portion of a big database. In RDBMS, data stores in a specific or static structure, in other words data arranges in form of row and column.

Before starting with any explanation, we need to answer three question. The questions are-

1. What is the problem?
2. Why it's a problem?
3. How to solve it?

By answering all these three question here, you will be able to understand the motive of our work.

So, first what is the problem? Relational database is defined at the basic level by a series of table entities which contain columns and rows, linked to other table entities by shared attributes. So, for example, a small online business might have a SQL database behind website with a table recording the name and email address of customers. Another table might record product names and their prices. A third table might link the two, recording which customers bought which products, with additional information such as the date of purchase and whether or not any discount was applied.

So one can quickly see how this information could be useful, some analysis will give the average spend per customer; a list of regular customers, and a list of inactive ones; a list of the most popular products. From this simple data a small business holder can make good business decisions. However, in the recent years with the tremendous rise in use of internet in every hour of every day, the Databases grown into thousands and thousands of terabytes. And also large data comes in a variety of forms. Organizations are facing more and more big data challenge. They have to access a wealth of information, but they don't know how to get value out of it because it is sitting in its most raw form or in a semi-structured or unstructured format. When we deal with these data, if we use traditional relational database concepts which requires to know in-depth knowledge about Database system, programming knowledge such as- SQL query, relation database, database design, database implementation, basic data structures, computer organization, high level programming language such as- Java, C or Pascal etc.

Now, why it’s a problem? As an example, consider we need to work with huge amount of data of past 40 years. With a typical RDBMS implementation, first we need to create a database, then create some tables and then insert those data into the tables. Firstly, only an expert who has programming as well as database knowledge can access and work with these data. While it’s very difficult for a normal person to understand all these mechanisms without having any previous knowledge. Secondly and most importantly, even if someone manages to gather knowledge about all those things, thus the process of storing these huge amount of data, updating them, is very time consuming.

How to solve it? For reducing all these difficulties, here we are representing our tool, which works over any numbers of flat files for executing queries. The primary goal of our project is to develop a user friendly tool, using which any user who have no knowledge about relational database management system, structured query language, which can support any numbers of flat files. We have developed a tool for them so that they could execute queries over the flat files and we are executing our tool over the Hadoop framework, inside the program we have used MapReduce programming paradigm. It does not directly ask for query from the user, rather it gives different options, conditions to the user. Any user can access the tool very easily. It does not require any previous knowledge about RDBMS or any other programming knowledge. The only thing a user need, having flat files. It’s very easy to use and fulfill all the queries of a user within a very short time.

1.2 Related Works

Structured Query Language:

Structured Query Language are commonly known as SQL. SQL is used to communicate with database. According to ANSI (American National Standards Institute), it’s the standard language for relational database management system. SQL statements are used for storing, manipulating and retrieving data stored in a relational database. Some common relational database system that use SQL are: Oracle, Sybase, Microsoft SQL server, Access etc. Although most database systems use SQL, most of them have their own additional extensions that are only used on their system. However, the standard SQL commands to interact with relation databases are CREATE, INSERT, DELETE, UPDATE and DROP.

When Relational Database Management System gets a command for executing SQL, the system determines the best way to full fill the request and SQL engines figures out how to interpret the task. There are various components include in the process such as: Query Dispatcher, Optimization Engine, Classic Query Engine and SQL Query Engine etc.

Following diagram is showing SQL architecture:

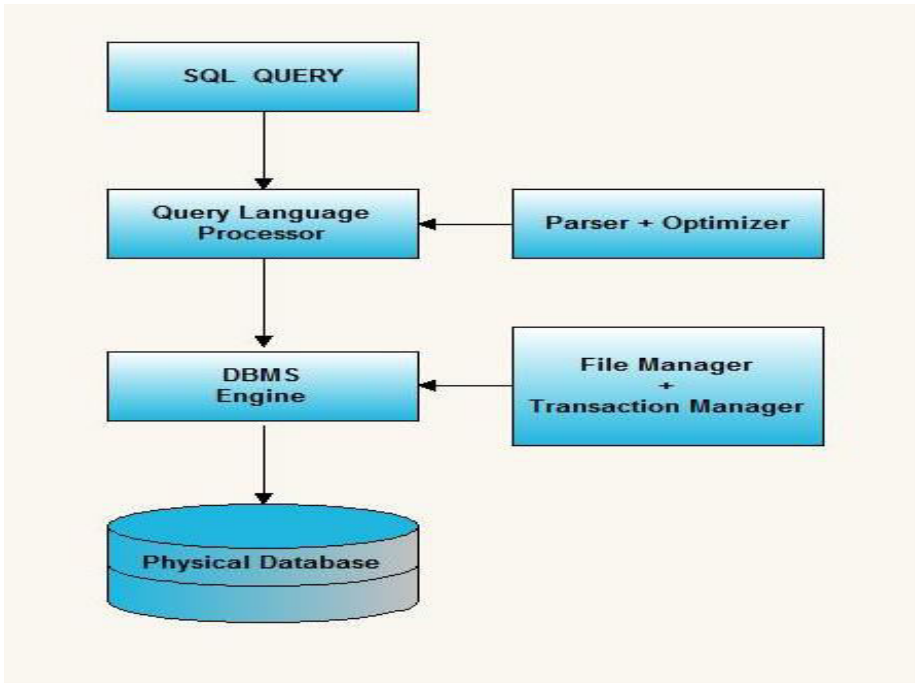


Figure1.1: Simple SQL architecture

NoSQL:

NoSQL, originally referring to “non SQL” or “non-relational”, database provides a mechanism for storing and retrieving data in relational databases (RDBMS). Generally, NoSQL databases are structured in a key-value pain, graph database, document-oriented or column-oriented structure.

Over decades and decades of software development, we have been using database in form of SQL where our data store in a relational table. In NoSQL scenario, it’s schema less, it does not follow any strict data structure. NoSQL database introduce for mainly dealing with huge and huge amount of data.

Big Data:

Big Data is a collection of a large dataset that cannot be processed using traditional computing techniques. Mainly three characteristics define Big Data: *volume, variety* and *velocity*.

- **Volume**

  Volume refers to the vast amount of data generated in every moment. Big data requires processing high volume of low-density, unstructured Hadoop data-that is data of unknown value, such as Twitter data feeds, click streams on web page and a mobile app, network traffic and many more. The main task of Big Data is to convert those Hadoop data into valuable information.

- **Variety**

  Variety refers to the range of data types both structured and unstructured, domains and sources. Usually we used to store data from sources like spreadsheets and database, but nowadays, data generated by different web pages, web log files, search index, social media forums, e-mail, documents, sensor data from active and passive systems and so on.

  Traditional data was structured data, neatly fitted in columns and rows but those days are over. These days, over 90% of data, generated by organizations, are unstructured. Data today comes in many different formats: structured data, semi-structured data, unstructured data and even complex structured data. These wide *variety* of data requires different techniques to store them and as well as access them. Traditional analytical data can't handle *variety*. However, an organization's success will rely on its ability to draw insights from the various kinds of data available to it, which includes both traditional and non-traditional data.

- **Velocity**

  Velocity refers to, how fast the sheer *volume* and *variety* of data are generated, stored, analyzed and visualized. In the past, when batch processing was common practice, it was easy to receive an update from database in every night or even every week. Substantial time was required for computers and servers to process the data and update the database. In the big data era, data is created in real time or near real time.

  The speed at which data is flowing is unimaginable. Every minute we upload 100 hours of video on YouTube. In addition, every minute over 200 million e-mails are sent, around 20 million photos are viewed and 30,000 uploaded on Flicks, almost 300000 tweets are sent and 2.5 million queries on google are performed.

  Big data requires that these *volume* and *variety* of data should be analyzes while it is *still in motion*, not just after it is *at rest*.

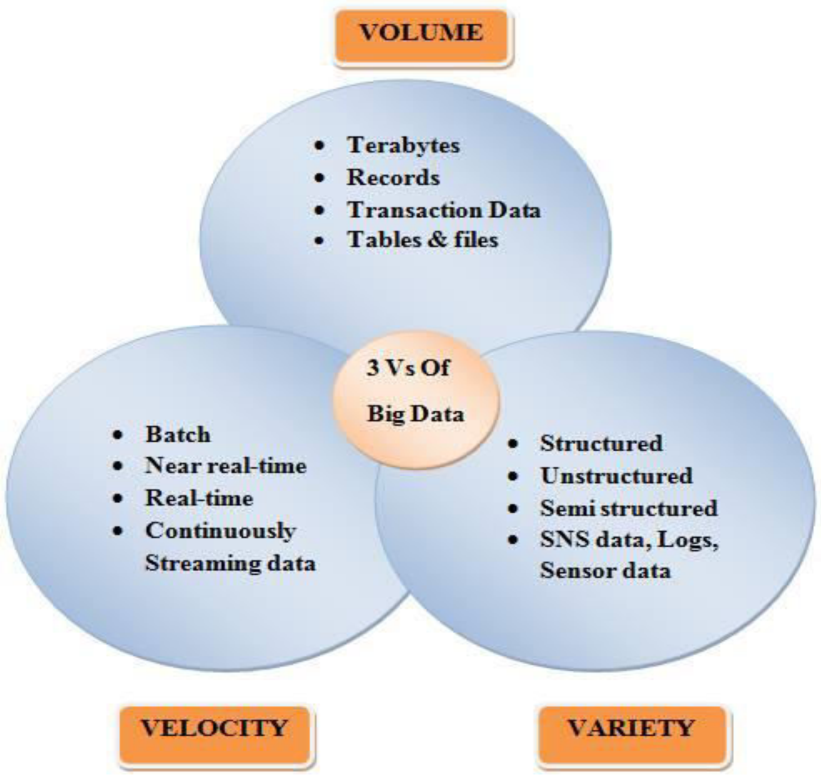


Figure1.2: Big Data characteristics

Hadoop:

Hadoop is an open-source framework, where we feel that Hadoop based platform is well suited to deal with structured and unstructured data. It allows to store and process huge amount of data in a distributed environment across cluster of computers using simple programming models. We will discuss more detailed things of Hadoop framework in Chapter 2.

1.3 Objective

The main objective of the thesis is to develop a user configurable Query Processing tool, using which without having any knowledge of the used framework of the developed tool a particular user can have the benefit of executing queries. Its user configurable in the sense that using this query tool user does not have to know anything about Hadoop, anything about relational queries. Both expert and inexpert users can use this tool, the only thing the user need to have any numbers of data files.

1.4 Related Issues

i. **Which types of Quires the Tool can handle?**

- ✓ SELECT column1, column2....columnN
  FROM table_name;

- ✓ SELECT column1, column2....columnN
  FROM table_name
  WHERE CONDITION;

- ✓ SELECT column1, column2....columnN
  FROM table_name
  WHERE CONDITION-1 {AND|OR} CONDITION-2;

ii. **Which types of data files the Tool can handle?**

- ✓ Unstructured and semi-structured data with 'N' number of attributes.
  For example,
  <attribute1><attribute2><attribute3>……………<attributeN>

iii. **What types of framework and programming language is used?**

- ✓ Framework Used
  - o Hadoop
  - o MapReduce

- ✓ Programming Language Used
  - o JAVA

1.5 Outline

This report is organized as follows:

i. Chapter 2, which contains a brief overview on Hadoop's architecture, Hadoop Distributed File System and Hadoop MapReduce.
ii. The architecture of the tool, discussed in Chapter 3 and will give a clear view about the tool architectural behavior to the user. In this chapter, we are using block diagram, activity diagram and sequence diagram to make things easier to the user.
iii. Chapter 4 covers the brief description of each component related with the tool.
iv. Chapter 5 is all about conclusion and future work.

Chapter 2

Hadoop Overview

2.1 Hadoop

Hadoop provides a distributed file system and a framework for storing and processing huge *volume* and *variety* of data sets using MapReduce paradigm. It's mainly a Java based framework. With data *volumes* and *variety* constantly increasing, a technology was needed that can handle and process data in faster way. Hadoop has the ability to store and process huge amounts of any kind data. It can work with numerous numbers of flat files. Hadoop's distributed computing model can process big data fast. The more nodes will be used, the more computing power it will have. Data need not to process before storing it, any amount of data can be stored.

Hadoop framework includes following four models:

- **Hadoop Common:** Hadoop common is Java libraries and utilities, which are required by other Hadoop modules. They provide filesystem and OS level abstractions and contains the necessary Java files and scripts required to start Hadoop.
- **Hadoop YARN:** Hadoop YARN is a framework which is used for job scheduling and cluster resource management.
- **Hadoop Distributed File System (HDFS):** A distributed file system that provides high-throughput access to application data.
- **Hadoop MapReduce:** Hadoop MapReduce is a YARN-based system for parallel processing of large data sets.

2.2 Hadoop Distributed File Systems

HDFS is mainly file system component of Hadoop. Hadoop Distributed File System is designed to run on large clusters of small machines in a reliable, fault-tolerant manner. HDFS basically follow Google File System (GFS) and provides a distributed file system. HDFS separately stores file system metadata and application data.

HDFS uses a master/slave architecture where master consists of a single NameNode which manages the file system metadata and one or more slave DataNodes which store the actual/application data.

HDFS namespace is a hierarchy of multiple files and directories, where files and directories are represented on NameNode. Files are split into several large blocks and each block of the files are independently replicated and stored in at multiple DataNodes. The NameNode determine the mapping of blocks. The DataNodes takes care of read and write operation with the file system. The DataNode represented by two files in the local host. First file contains the data itself and the second file contains block's metadata including checksum.

When a client wants to read a file, it first contact with the NameNode for the location of data file. Then NameNode compares the data files and reads block contents from the DataNode closet to the client. The NameNode does not directly call DataNodes. It uses heartbeats to send instruction to the DataNodes. They also perform block creation, deletion and replication based on given instruction by the NameNode.

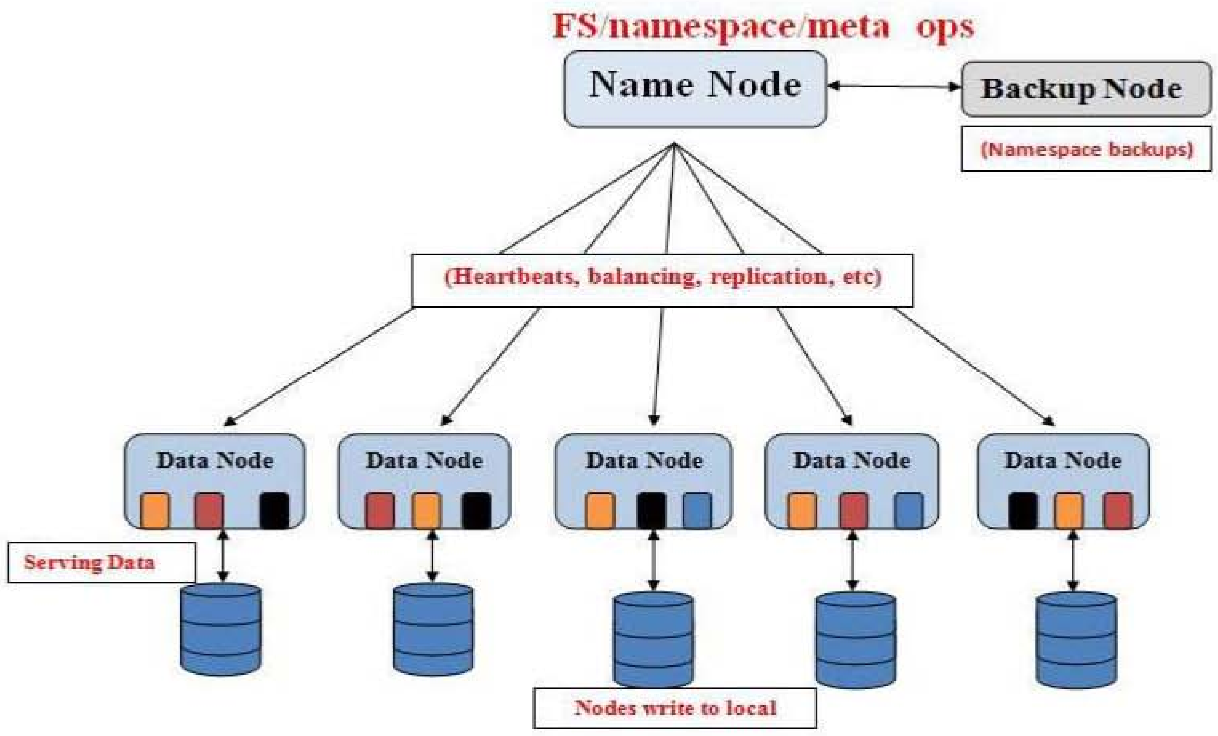


Figure 2.1: HDFS Architecture

2.3 Hadoop MapReduce

MapReduce is the original framework for writing applications that process large amounts of structured and unstructured data, stored in the Hadoop Distributed File System (HDFS). Apache Hadoop YARN opened Hadoop to other data processing engines that can now run alongside existing MapReduce jobs to process data in many different ways at the same time.

MapReduce implementation is combined with several systems, including Google's internal implementation and the popular open implementation of Hadoop which can be obtained along with the HDFS file system from Apache Foundation. MapReduce can manage large scale computation, with the help of only two function: Map and Reduce.

A MapReduce job splits a large data set into independent chunks and organizes them into key, value pairs for parallel processing. This parallel processing improves the speed and reliability of the cluster, returning solutions more quickly and with greater reliability.

- **The Map Task:** Files consisting of elements, are input for Map task, which can be any type, structured or unstructured. A chunk is a collection of elements and no elements is stored across two chunks. The inputs of Map tasks and output from Reduce tasks are the key-value-pair form.
  The Map function takes an input element as its argument and produces zero or more key-value pairs. The types of keys and values are arbitrary. A Map tasks can produce several key-value pairs with the same key, even from the same element.
  Such as: (k,v1), (k,v2),………., (k,vn). Where k denotes for keys and v denotes for values.

- **Grouping by Key:** After the completion of successful Map task, the key-value pairs are grouped by key and the values associated with each key are formed into a list of values. Grouping is performed by the system. The user tells the MapReduce system that how many Reduce task there will be and inform it to the master controller process. Then the master controller picks a hash function that applies to keys and produces a bucket number from 0 to (Reduce tasks - 1). Each key of Map task is hashed and it's key-value pair is put in one of r local files and each files is send for Reduce tasks.
  The master controller merges the files before sending to the Reduce task and process as a sequence of key-list-of-value pairs such as: (k,[v1,v2,……,vn]), where (k,v1), (k,v2), ……., (k,vn) are all the key-value pairs of Map tasks output.

- **The Reduce Tasks:** Reduce task receives one or more keys and their associated value lists. Reduce task can execute one or more reducers. The output from all the Reduce tasks are merged into a single file. Reducers may be partitioned among a smaller number of

Reduce tasks is by hashing the keys and associating each Reduce task with one of the buckets of the hash function.

- **Combiners:** Reduce function is associative and commutative. That means the value can be combined in any order with the same result. It does not matter how we group a list of number, the sum will always be the same.

With the help of Hadoop, the user program forks a Master Controller process and some number of Worker processes. Worker process can handle either Map tasks or Reduce tasks, but not both at the same time. The Master has responsibilities to create some number of Map tasks and some number of Reduce tasks, moreover, this tasks are selected by the user program. Master will assign these tasks to the Worker processes.

Master keeps track of the status of each Map and Reduce tasks, Worker process reports to the master. When it finishes a task, new task is scheduled for the Worker by the Master.

Each Map task is assigned one or more chunks of the input files and execute on it the code written by the user. The Map task creates a file for each Reduce task on the local disk of the worker that execute the Map task. The Master is informed of the location and sizes of each of these files. The Reduce task executes code written by the user and writes the output to a file that is part of the surrounding distributed file system.

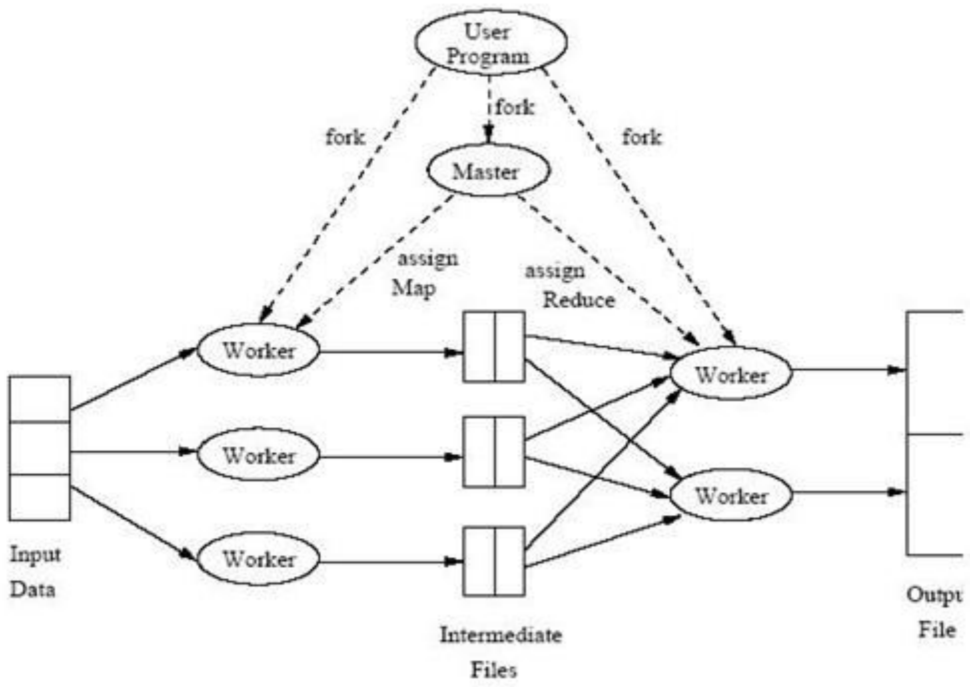


Figure 2.2: MapReduce Architecture

Chapter 3

Architecture of Proposed Solution

In this chapter we will discuss about our tool's architecture. The brief discussion of all these steps in at Chapter 4.

3.1 Overview

The steps involved in our tool appear in Figure 3.1. The basic steps are:

i. Input File Manager
ii. File Splitting Manager
iii. Query Processing Manager
iv. File Handling Manager
v. MapReduce
vi. Output Processor

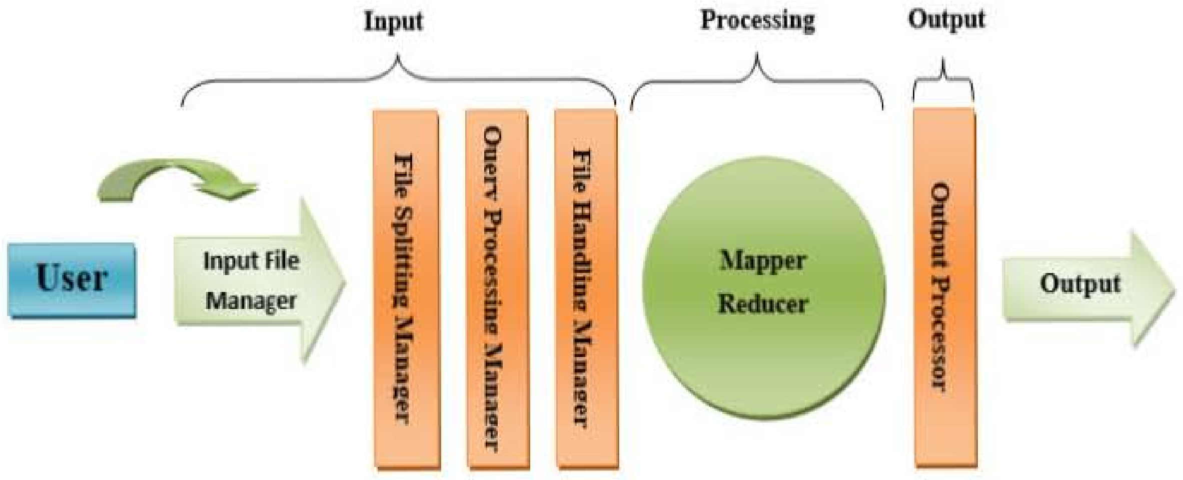


Figure 3.1 Architecture of our Tool

For using this tool, a user must give all the information to the input part. By getting input, MapReduce will start processing the job and then generate a desire output that a user wants to see.

3.2 Input File Manager

Input file manager will ask the user about the previously configured input file. If the user has any previous input file, user will give the configuration file name to the system otherwise user will continue with new configuration, which will be handled by file splitting manager.

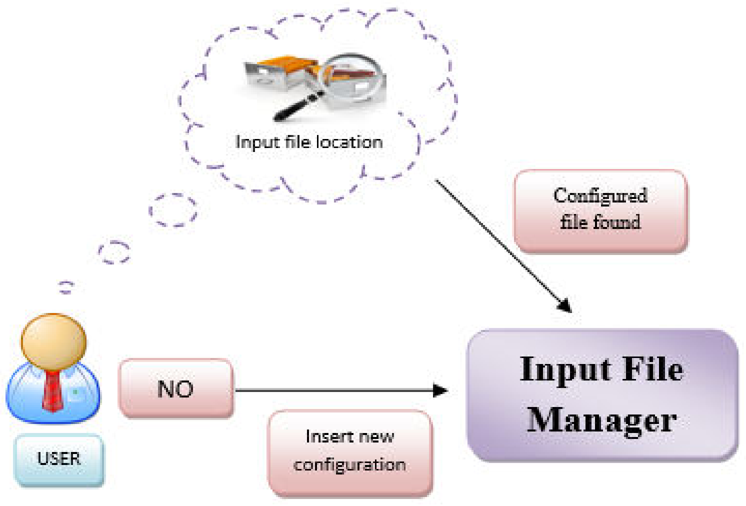


Figure 3.2: Input File Manager Architecture

3.3 File Splitting Manager

File splitting manager, first will take the new configuration name from user and then will take the split information. Then it will split each lines of the file according to user configuration.

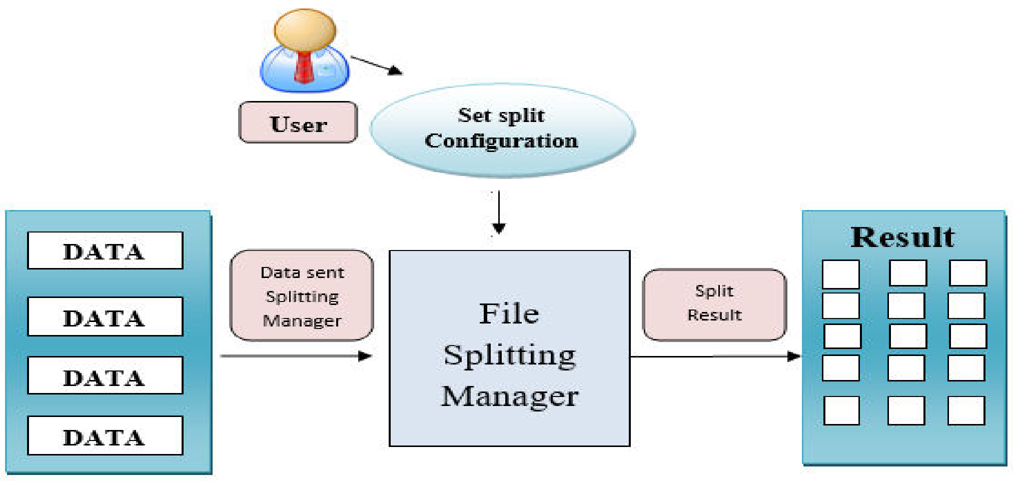


Figure 3. 3: File Splitting Manager

3.4 Query Processing Manager

Query processing manager will ask the user about different information that what type of queries he/she wants to form and then the user information will be given to the file handling manager.

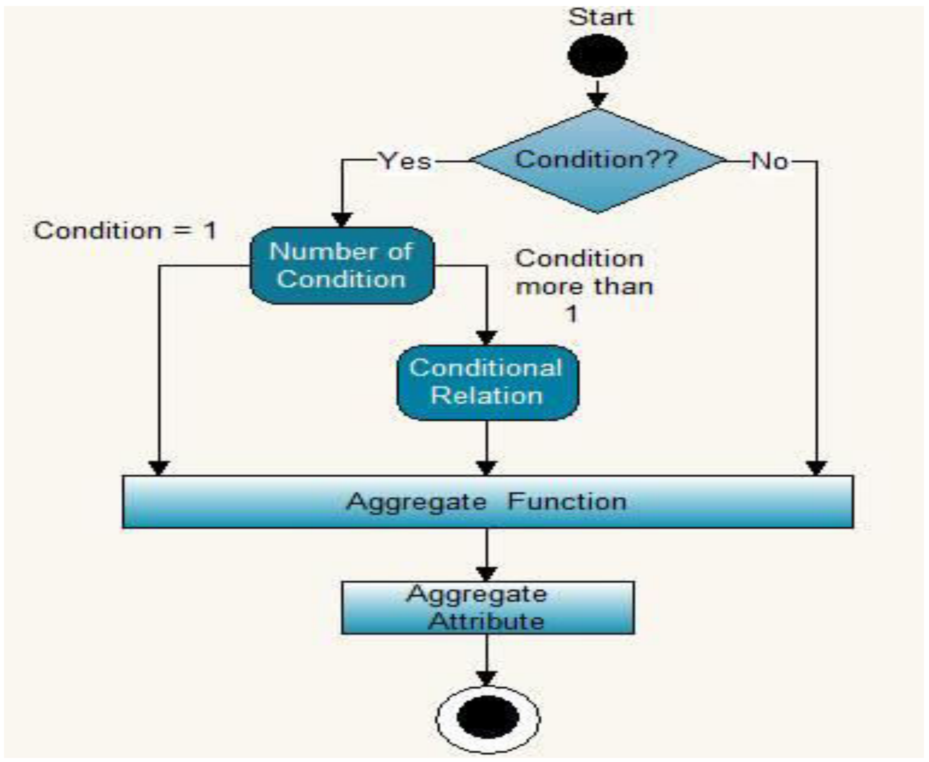


Figure 3.4: Query Processing Manager

3.5 File Handling Manager

File handling manager will ask the user that which input file the user would like to use and where the output will be generated.

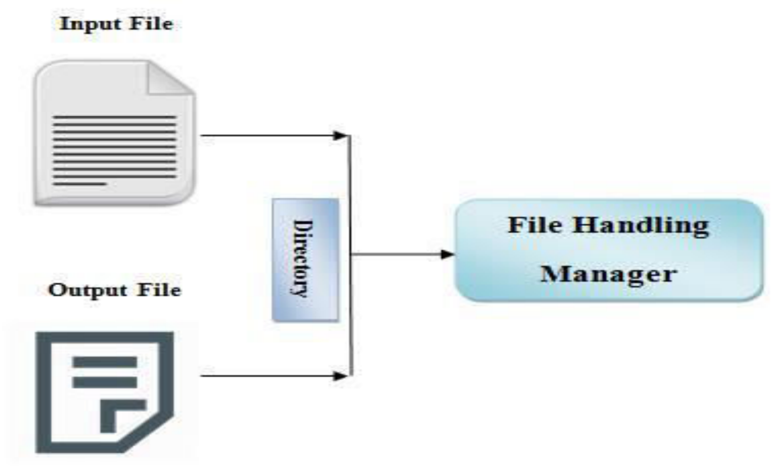


Figure 3.5: File Handling Manager

3.6 MapReduce

MapReduce will work just like traditional mapper and reducer but on top of that some extra condition checking will be done by doth mapper and reducer on the basis of user information. and will produce a result

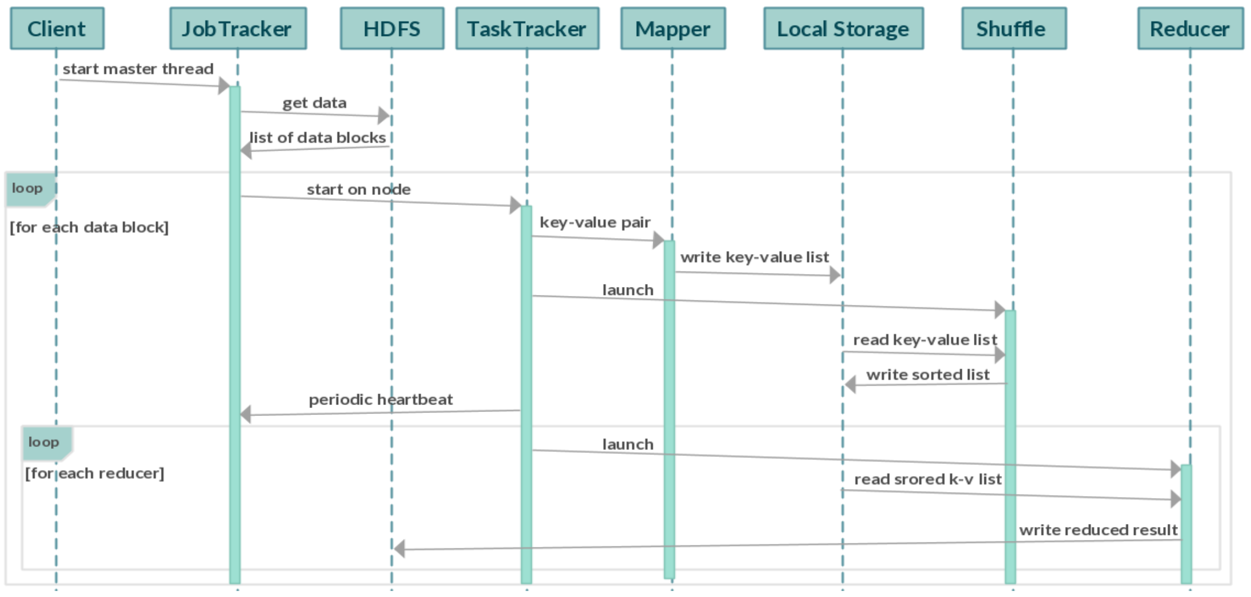


Figure 3.6: MapReduce Architecture

3.7 Output Processor

The generated result after MapReduce task will be stored in a text file and output processor will access the file and show output to the user.

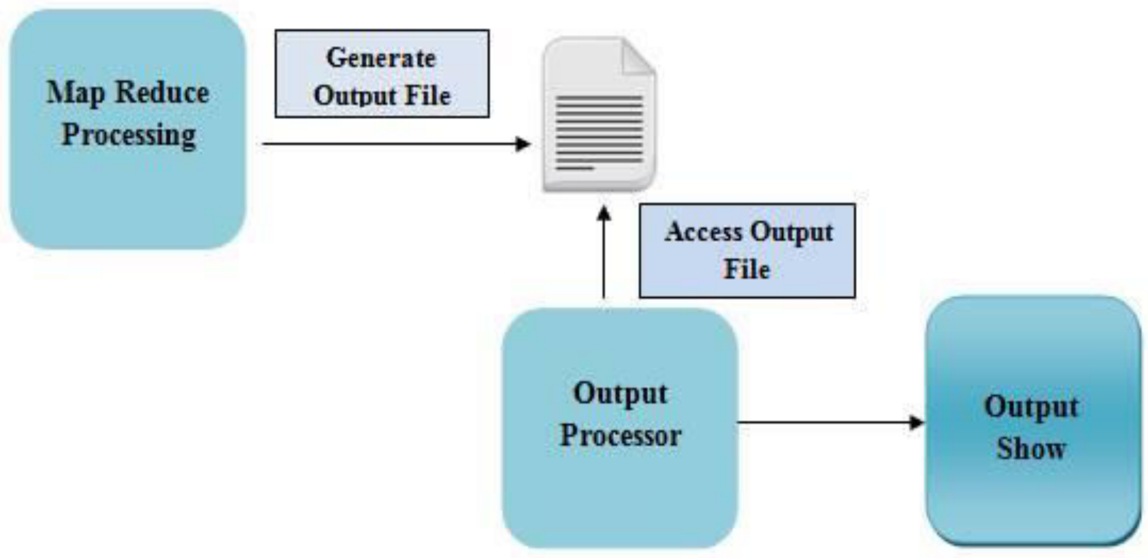


Figure 3.7: Output Processor

Chapter 4

Development of Tool

4.1 Background

- **Environment Used**
    - ✓ Hadoop. Because Hadoop has the ability to run application on systems with thousands of nodes involving thousands of terabyte with faster computing process.

- **Hadoop Version**
    - ✓ Hadoop-1.2.1

- **Operating System Used**
    - ✓ Linux (Ubuntu, Version 12.04)

4.2 Component Wise Discussion

For using this tool, a user must have some flat files. All the component of the files should be in same length, if needed a user can do some adjustments, which is known as file pre-processing. That mans every field length of each attribute must be equal.

4.2.1 Input File Manager

As the data are unstructured, so a user needs to configure the input data files. Input file manager is responsible for this step. In this step, the system will check whether he/she is a previous user. It means that whether the user has already given some input which has been executed on this system.

If a new user starts the process, then input file manager will take the configuration name from the user and give the rest of the responsibility to the file splitting manager. The configuration file name will be saved in a text file for further use.

For a previous user, if he wants to continue with the old configuration, he/she can do so. In that case the user must tell the existing configuration name to the input file manager.

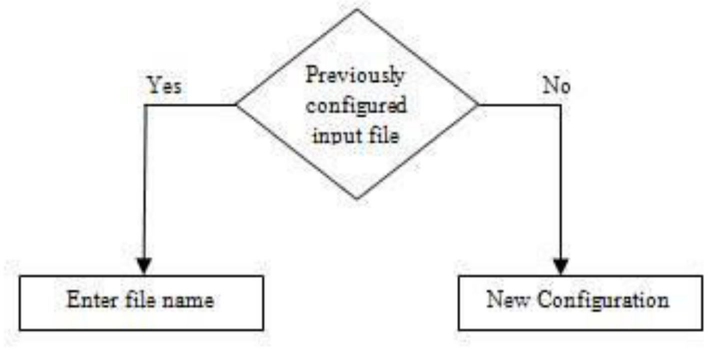


Figure 4. 1 Flow chart for Input File Manager

4.2.2 File Splitting Manager

The file splitting manager starts working after the input file manager successfully complete its work.

First file splitting manager will take the configuration file name from the user. Then the system will save all the information given by the user related to the text file, so that if a user wants to use the same configuration again and again, he/she can do so. In this case, a user should keep in mind that the files name should be unique, user will not be able to use existing file to save the configuration, he/she must create a new one.

Then the user will give input to the file splitting manager that in how many segment he/she wants to split the text files, that means how many attributes will be there and the other information like the name of the attribute, data type, range of the splitting of the index.

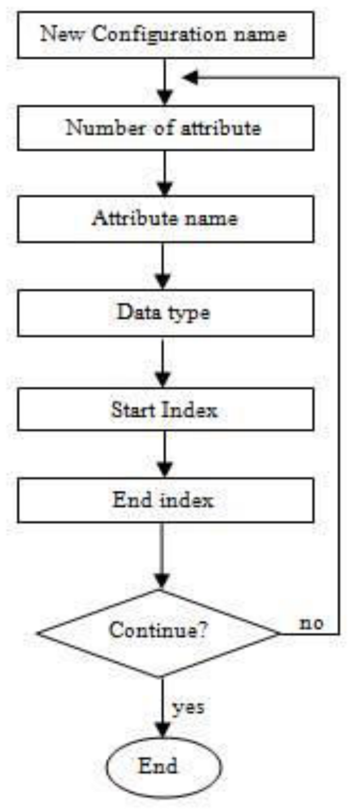


Figure 4. 2 Flow chart for File Splitting Manager

4.2.3 Query Processing Manager

Query processing manager is the most important part for our tool. The users will be able to know all their query from here. But the most interesting part of the tool is, user doesn't have to write any traditional query here. For example, our traditional query structure is,

```
SELECT column1, column2....columnN
FROM   table_name
WHERE  CONDITION;
```

Nothing like this has to be written by the user. Now the question is, How will the processing tool work? It's simple. In this step the system will ask the user for the conditions which we use in the traditional query as "WHERE". For now, the system is capable of handling up to two conditions.

Now if the user selects that he/she has one condition then the system will go to the next step and will ask the user for aggregate function. But if the user selects that he/she has two conditions, then the system will ask the user that in which way the conditions are related to each other.

For example-

condition1 "&&" condition2

or

condition1 "||" condition2

Then the user will select the aggregation function. Here the system will ask the user that which aggregation function of what attribute he/she wants to see and then user information will be given to the file handling manager.

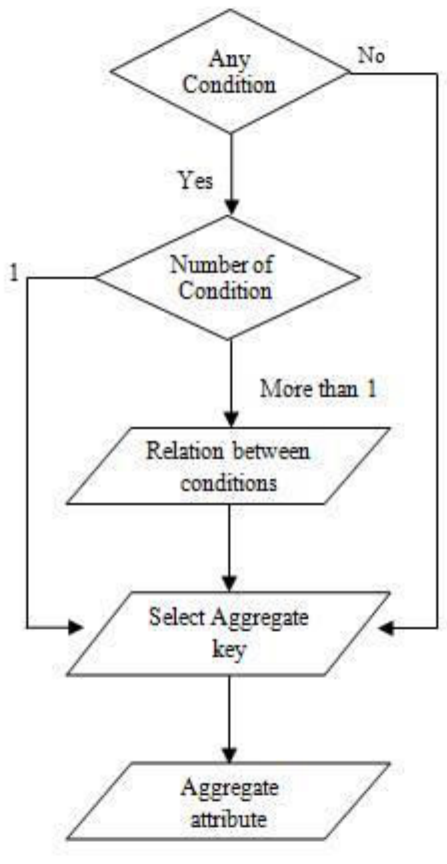


Figure 4. 3 Flow chart for Query Processing Manager

4.2.4 File Handling Manager

File handling manager deals with the input and output file. Input file is the one where user have the all kinds of flat files. Our tool can work with any number of flat flies. The task we do by using "FROM" clause in the traditional RDBMS, we do the same work here only by selecting the input file. In traditional RDBMS we use table name in "FROM" because the data is structured and designed as rows and columns. But here the data is not structured, so the users don't have to input data in the database rather they can directly use the flat files as input the data through file handling manager.

Output file is the place where MapReduce generates the output. In this case the user must be careful about the fact that not any "existing folder" is used as output file. The user must give a now folder name for generating output file. The users don't have to create the folder in the home page rather he/she just need to insert the file name in the file handling manager.

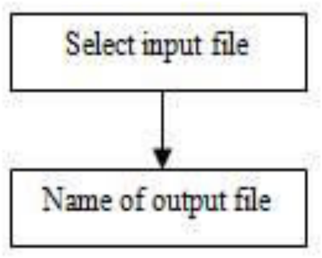


Figure 4. 4 Flow chart for File Handling Manager

4.2.5 MapReduce

System will take all the user information and after that the actual program will start working, which will be executing under Hadoop framework, without any user intervention. Mapper and Reducer will work as traditional Mapper and Reducer. But some extra feature will be added both Mapper and Reducer task.

First Mapper will read all the text file where the user inputs are saved. It will read all the attribute and put all the attributes to different hash map according to data type. If an integer type attribute is found, then it will put in the integer type hash map. If it's a string type data, then it will put in the string type hash map and so on. Then mapper will check all the conditions and according to the conditions it will generate key, value pairs and send them to the reducer.

Reducer will check what aggregate key operation the user wants to perform. According to that choice reducer will perform its task and generate a result.

4.2.6 Output Processor

The result that MapReduce will generate, will be kept in a text file which will be selected by the user. Output processor will read the text file by using given file location and will display the final result to the user.

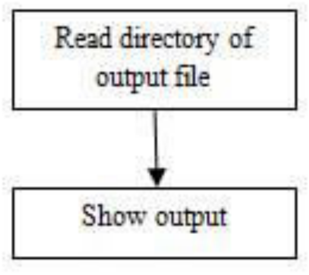


Figure 4. 5 Flow chart for Output Processor

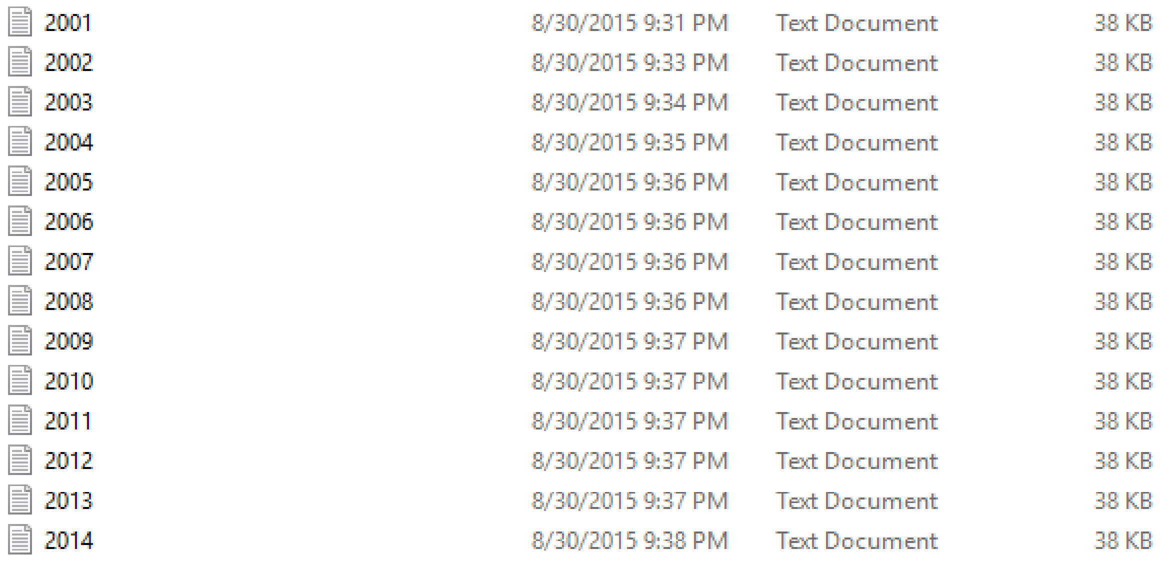
4.3 Working Procedure

To work with this project, first we have learnt about Java, Hadoop, MapReduce, HDFS. As the domain is new, so before start working a developer should have in-depth knowledge about all these related frameworks.

Our main motivation was to develop a user-friendly tool, using which a user who has no knowledge about SQL, Relational Database, Hadoop, MapReduce and all, can have the benefit of executing queries using some menu based system over Hadoop framework. We chose Hadoop framework because Hadoop is an open source framework, which can handle huge numbers of data and it stores data much more cheaply.

Our used operation system, version and environment, all these things we are discussed at the beginning of this chapter. Now we are directly discussing the whole working procedure below:

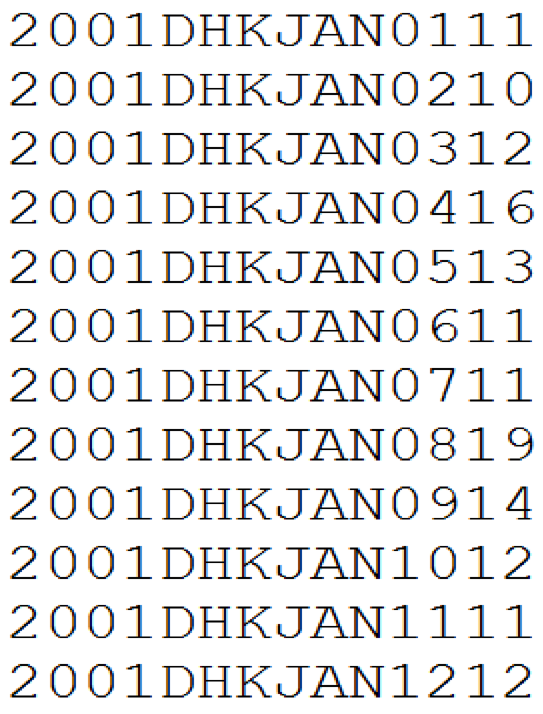
1. For our tool we are using some flat files as input, shown in Figure 4.6, which are the weather data of Bangladesh from 2001 to 2014. There are 14 files for 14 years and each files contains the same amount of data. These are the synthetically generated data.



2001	8/30/2015 9:31 PM	Text Document	38 KB
2002	8/30/2015 9:33 PM	Text Document	38 KB
2003	8/30/2015 9:34 PM	Text Document	38 KB
2004	8/30/2015 9:35 PM	Text Document	38 KB
2005	8/30/2015 9:36 PM	Text Document	38 KB
2006	8/30/2015 9:36 PM	Text Document	38 KB
2007	8/30/2015 9:36 PM	Text Document	38 KB
2008	8/30/2015 9:36 PM	Text Document	38 KB
2009	8/30/2015 9:37 PM	Text Document	38 KB
2010	8/30/2015 9:37 PM	Text Document	38 KB
2011	8/30/2015 9:37 PM	Text Document	38 KB
2012	8/30/2015 9:37 PM	Text Document	38 KB
2013	8/30/2015 9:37 PM	Text Document	38 KB
2014	8/30/2015 9:38 PM	Text Document	38 KB

Figure 4.6 Input files used in our project

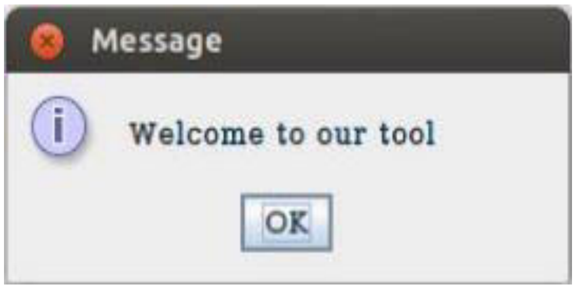
2. Figure 4.7 shows the structure of the data and all the files contain same structure. First we have year, then city, month, date and finally temperature. We will split these data according to this.



```
2001DHKJAN0111
2001DHKJAN0210
2001DHKJAN0312
2001DHKJAN0416
2001DHKJAN0513
2001DHKJAN0611
2001DHKJAN0711
2001DHKJAN0819
2001DHKJAN0914
2001DHKJAN1012
2001DHKJAN1111
2001DHKJAN1212
```

Figure 4. 7 Structure of Data

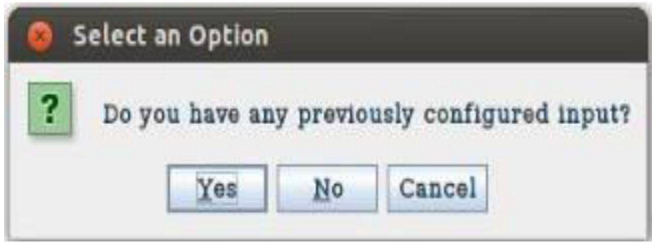
3. When the user will execute our tool, then a welcome message will be appeared just like the figure 4.7 and the user will be continuing by pressing ok.



Welcome to our tool

Figure 4. 8 Welcome Message

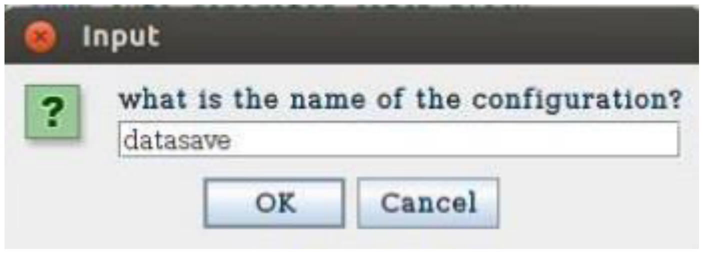
4. Then the system will ask the user if he/she has any previously configured input file or not. It means that weather the user has already given some input which has been already executed on this system. If the user has any, then he/she will press yes. If not, then user will have to give new configuration.



Do you have any previously configured input?

Figure 4. 9 Asking for previous configuration

5. If the user is new or user wants to continue with a new configuration, then user should input configuration name first.

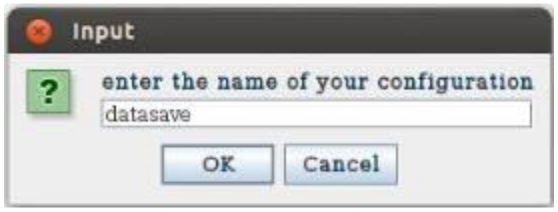


what is the name of the configuration?

datasave

Figure 4. 10 Enter new configuration name

6. if the user wants to continue with the old configuration and press no, a new window will be appeared and will ask for previous configuration name. user should give the configuration name in the input box.

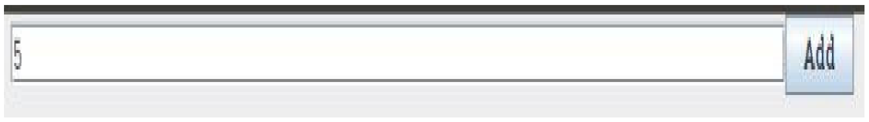


enter the name of your configuration

datasave

Figure 4. 11 Insert old configuration name

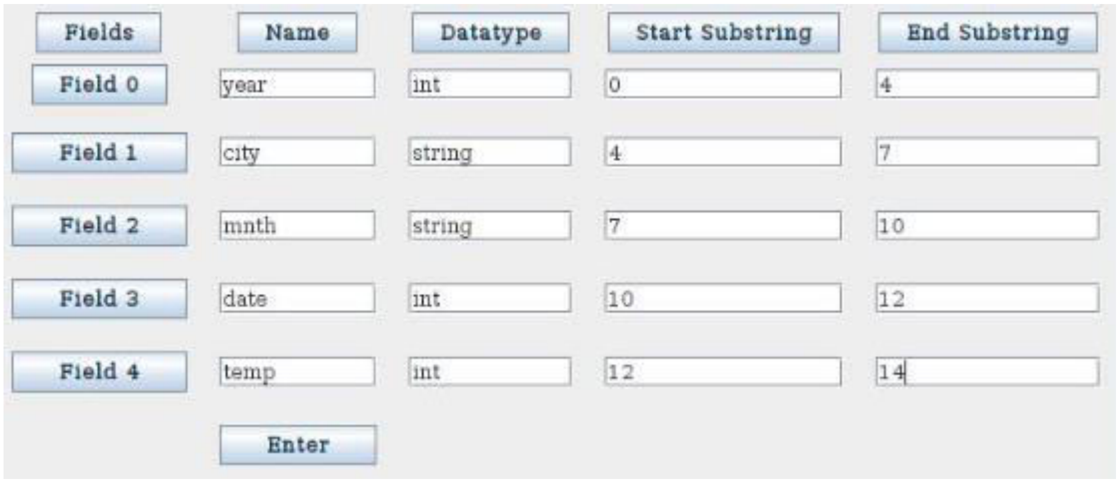
7. After giving the configuration name, a new frame will be appeared. In this frame user has to input the number of attribute he/she has and then will press add.



5

Figure 4. 12 Insert number of attribute

8. When the user will press "add", a new field will be open in the same frame. It will contain field number, attribute name, data type, start index and end index just like figure 4.12. User must fill all the field and press enter.



Fields	Name	Datatype	Start Substring	End Substring
Field 0	year	int	0	4
Field 1	city	string	4	7
Field 2	mnth	string	7	10
Field 3	date	int	10	12
Field 4	temp	int	12	14

Figure 4. 13 All splitting information

9. After pressing enter button, the query processing manager's work will start. First it will ask the user if he/she has any conditions.

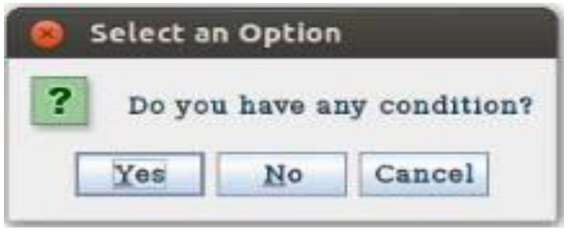


Figure 4. 14 Checking for conditions

10. If the user press yes, then system will ask the user about the number of conditions and continue with pressing ok.

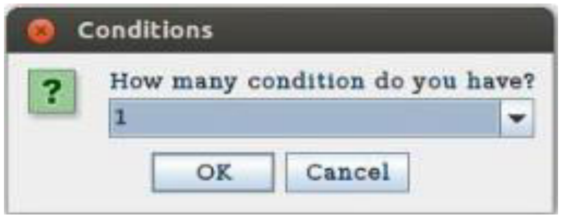
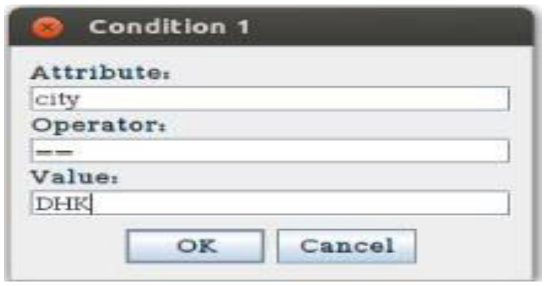


Figure 4. 15 Number of condition

11. If the number of condition is 1, then one window for condition will be appear.


Condition 1

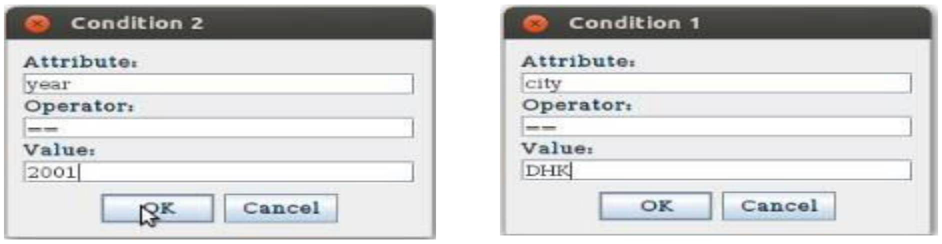
Attribute: city

Operator: ==

Value: DHK

Figure 4. 16 Condition for one

12. If the number of condition is 2, then two windows will be appeared.


Condition 2

Attribute: year

Operator: ==

Value: 2001

Condition 1

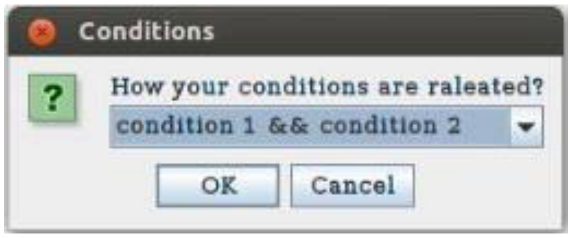
Attribute: city

Operator: ==

Value: DHK

Figure 4. 17 Conditions for two

13. If the number of conditions is 2, then the user will have to select the relation between two conditions. Weather the conditions are connected with "&" operator or "||" operator.


Conditions

How your conditions are raleated?

condition 1 && condition 2

Figure 4. 18 Relation between conditions

14. When the condition part will be over or if the user select that he/she has no conditions, then then system will ask the user for selecting aggregate key.

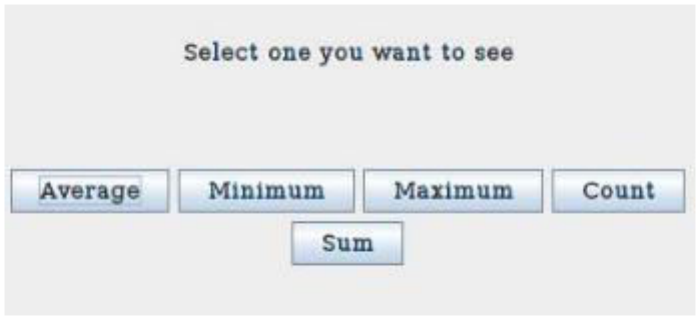
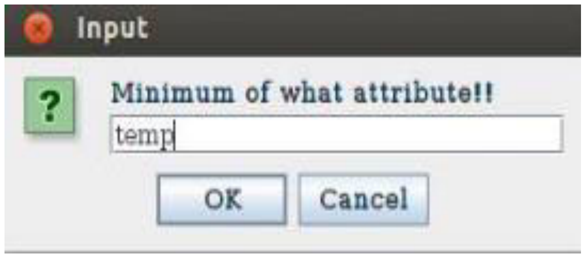

Select one you want to see

Figure 4. 19 Select aggregate key

15. The user must select anyone form this box and then system will ask the user that on what attribute he/she wants to perform aggregate operation. For example, if the user selects minimum, then system will ask minimum of what attribute. Then the user must insert the attribute name.


Input

Minimum of what attribute!!

temp

Figure 4. 20 Select attribute

16. After successfully completing all these steps, finally the system will ask the user about his/her input and output file directory. User will have to select the folder that contains the input files and write the name of the file where he/she wish the store the output.

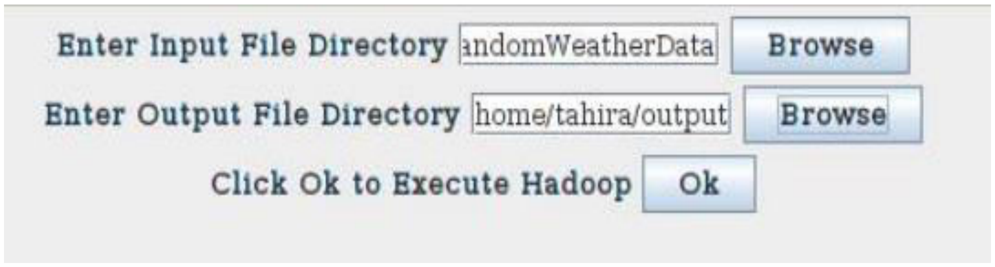
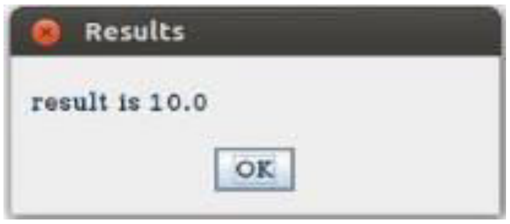

Enter Input File Directory andomWeatherData
Enter Output File Directory home/tahira/output
Click Ok to Execute Hadoop

Figure 4. 21 Select directory

17. Finally, MapReduce will start working and generate an output in the selected output folder. Output processor will read the output and show it to the user.


Results

result is 10.0

Figure 4. 22 Final Output

And thus, the execution of our developed tool will be completed. A user can use any numbers of flat files and can execute the tool as many times as he/she wants.

Chapter 5

Conclusion & Future Work

5.1 Conclusion

Data *volume* are increasing every second. For this increasing data, a tool is required which can process user query without taking long time and any difficulties, as well as anyone can deal with these data without having any knowledge about any types of accessing method.

For example, in organizations they do not hire people to do just a specific task or it's not possible to hire only experts. Also the amount of data an organization generate every day, is unimaginable. Even if company hire experts, it will take huge time to manage all these data.

So our goal was to make a user configurable query tool that can manage huge amount of data easily and handle user queries. A tool that anyone can access. To operate this tool, user need not to do any hard work, need not to know about RDBMS, Hadoop, MapReduce, Java etc. So here is our tool. This tool can work with numerous number of flat files and also Hadoop, MapReduce framework works in background. We use Hadoop framework; inside it MapReduce programming paradigm is working. We use Hadoop because Hadoop is open source framework and it can work with huge amount of data files and MapReduce gives a privilege to work with faster way.

In this tool, it takes only input from users about their input file path, output file path, how they want to split their text file, how many attribute do they have, what are their types, what they want to see as output etc. Then process data without any user intervention.

Throughout this paper, we have discussed about tool. In Chapter 1, tried to give some knowledge to the reader that what was the problem, why it was a problem and to solve the problem what is our solution. Then we tried to give some little overview about SQL, NoSQL, Big Data, Hadoop. In chapter 2, we briefly discuss about Hadoop and different components of it such as- Hadoop Distributed File System, Hadoop MapReduce. Chapter 3 was the architectural part of our developed tool. In that chapter, we discuss about different parts of our tool with the help of different diagram. Finally, Chapter 4 was the brief discussion part of the developed tool, used environment and working procedure of our tool.

5.2 Future Work

i. We would like to see, whether we could extend this tool for multi-valued database.
ii. We want to extend our tool for addressing other types of dataset.
iii. Till now our tool can only handle simple and WHERE clause related query. But we would like to work with some other queries like- group by query, nested query etc.
iv. In our tool, the conditions are limited. That a user can user only up to two conditions. We would like to see, weather we could extend the tool for unlimited number of conditions.

References

[1] Big Data: A Revolution That Will Transform How We Live, Work, and Think Hardcover
Retrieved on February 27, 2016

[2] Running Hadoop on Ubuntu Linux
http://www.michael-noll.com/tutorials/running-hadoop-on-ubuntu-linux-single-node-
Retrieved on February 28, 2016

[3] Hadoop the Definitive Guide, 4th Edition by Tom Wbite. Retrieved on March 02, 2016

[4] MapReduce Design Patterns by Donald Miner & Adam Shook.

Retrieved on March 05, 2016

[5] Hadoop http://hortonworks.com/wp-Tutorial_Hadoop_HDFS_MapReduce.pdf

Retrieved on March 11, 2016

[6] http://www.snia.org/sites/default/education/tutorials/2013/fall/BigData/SergeBazhievsky

Retrieved on March 25, 2016

[7] https://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html

Retrieved on April 08, 2016

[8] http://www.tutorialspoint.com/hadoop/hadoop_mapreduce.htm

Retrieved on May 02, 2016

[9] Hadoop in Practice by Alex Holmes. Retrieved on May 28, 2016

[10] Professional Hadoop Solutions Kindle Edition. Retrieved on June 13, 2016

[11] MARP https://www.mapr.com/products/apache-hadoopRetrieved on June 25, 2016

[12] JBM https://www-01.ibm.com/software/data/infosphere/hadoop/mapreduce/ Retrieved on July 10, 2016

[13] Tutorials https://tutorials.techmytalk.com/category/big-data/ Retrieved on July 18, 2016

[14] Stack Overflow http://stackoverflow.com/questions/18629000/mapreduce-program- Retrieved on July 18, 2016

[15] Java Swing http://www.javatpoint.com/java-swing Retrieved on July 25, 2016

Appendix

1. MaxTemperature

```
import javax.swing.JFrame;
import javax.swing.JOptionPane;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.*;
import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.*;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;


public class MaxTemperature
{

        public void exec(String args0, String args1) throws Exception
                {
                                System.out.println(args0);
                                System.out.println(args1);


                       if (args0 == null && args1 == null)
                        {
                                System.err.println("Please Enter the
input and output parameters");
                                System.exit(-1);
                        }

                      Configuration conf = new Configuration();

                      Job job = new Job(conf, "min,max and average");

                      job.setJarByClass(MaxTemperature.class);
                      job.setJobName("Max temperature");

                      FileInputFormat.addInputPath(job,new
Path(args0));
                      FileOutputFormat.setOutputPath(job,new Path
(args1));

                      job.setMapperClass(MaxTemperatureMapper.class);

job.setReducerClass(MaxTemperatureReducer.class);
```

```
     job.setOutputKeyClass(Text.class);
     job.setOutputValueClass(DoubleWritable.class);


         job.waitForCompletion(true);
         finalOutput.main();
                JFrame frame = new JFrame();
           String message = "Do you want to continue !!!";
           int answer = JOptionPane.showConfirmDialog(frame, message);
               if (answer == JOptionPane.YES_OPTION) {
                    main1.main();
               } else if (answer == JOptionPane.NO_OPTION) {
                JOptionPane.showMessageDialog(null, "Thank You");
               }
               else{
                JOptionPane.showMessageDialog(null, "Bye Bye");
               }

            }

}
```

2. MaxTemperatureMapper

```
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.*;
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.HashMap;




public class MaxTemperatureMapper extends Mapper <LongWritable, Text,
Text, DoubleWritable>
{


     public String variableName;
     public String datatype;
     public Integer startstring;
     public Integer endstring;

     public String why;
     public String variableName2;
     public String datatype1;
```

```
      public String varia;
      public String variable1;
      public String newv;
      public Integer icnt;
      public String valu;
      public String operator;
      public String conname;
      public String convalu;
      public String conoperator;
      public String relation;


      public int[] I = new int[50000];
      public String[] S = new String[5000];
      public double[] D = new double[5000];
      //int[] F = new int[5000];

      public int cntI=0;
      public int cntS=0;
      public int cntD=0;


    public  HashMap<String, Integer> mapi = new HashMap<String,
Integer>();
    public  HashMap<String, String> maps = new HashMap<String,
String>();



    public  String counter;


    public int i=0;
    public int d= 0;
    public int j=0;
    public int e=0;
    public int f=0;


    public  String[] Var = new String[50000];
    public  String[] Type = new String[50000];
    public Integer[] Start = new Integer[50000];
    public  Integer[] End = new Integer[50000];

    public  String[] why1 = new String[500000];
    public  String[] Type1 = new String[500000];
    public  String[] Vr1 = new String[500000];
    public  String[] Vr2 = new String[500000];
    public  String[] Var2 = new String[500000];
    public  String[] Varo = new String[500000];
    public  Integer[] Varv = new Integer[500000];
public void map(LongWritable key, Text value, Context context) throws
IOException,  InterruptedException
{
           try
           {
                BufferedReader newrd = new BufferedReader(new
FileReader("config.txt"));
                String LineCur = null;
                String outputdir = null;
                while ((LineCur = newrd.readLine()) != null)
                {
                      outputdir = LineCur.toString();
                      System.out.println(outputdir);
                }
                newrd.close();



                BufferedReader br = null;
                String sCurrentLine;

                br = new BufferedReader(new
FileReader(outputdir+".txt"));

                int c=0; i=0;
               while ((sCurrentLine = br.readLine()) != null)
               {
                String line = sCurrentLine.toString();
                    String[] arr=line.split(" ");
                    variableName = arr[0];
                    datatype = arr[1];
                    startstring = Integer.parseInt(arr[2]);
                    endstring = Integer.parseInt(arr[3]);

                    Var[c] = variableName;
                    Type[c] = datatype;
                    Start[c] = startstring;
                    End[c] = endstring;
                    c++;
                  }

               cntI = 0; cntS = 0;
               String line = value.toString();

               for( i=0; i<c; i++){
                if(Type[i].contains("int")){
                      I[cntI] = 
Integer.parseInt(line.substring(Start[i], End[i]));
                          System.out.println("Value: " + I[cntI]);

                          varia = Var[i];
                          icnt = I[cntI];

                          System.out.println("new value "+ icnt);

                          mapi.put(Var[i], I[cntI]);
                          System.out.println("keys one " + Var[i] + " "
+ " "+ I[cntI] + " " + mapi.get(Var[i]) + " " + mapi);

                          cntI++;
                      }
                else if(Type[i].contains("string")){
                      S[cntS] = (line.substring(Start[i], End[i]));
                          System.out.println("Value: " + S[cntS]);

                          maps.put(Var[i], S[cntS]);
                          System.out.println("keys three " + Var[i] + "
" + " "+ S[cntS] + " " + maps.get(Var[i]) + " " + maps);

                          cntS++;
                      }
                     else{

                     }
               }
           }
           catch (IOException e)
           {
                e.printStackTrace();
           }


           BufferedReader br1 = new BufferedReader(new 
FileReader("attribute.txt"));
           String sCurrentLine1;

           while ((sCurrentLine1 = br1.readLine()) != null)
           {
                 String line1 = sCurrentLine1.toString();
                 String[] arr=line1.split(" ");
                 why = arr[0];
                 System.out.println("why contains "+mapi.get(why));
           }
           br1.close();

           BufferedReader reader = new BufferedReader(new 
FileReader("yes.txt"));
           String CurrentLine = reader.readLine();
```

```
            reader.close();


            if(CurrentLine.contains("yes")){

                  BufferedReader bufr = new BufferedReader(new
FileReader("counter.txt"));
                  counter = bufr.readLine();
                  System.out.println("counter value "+counter);
                  bufr.close();

                  BufferedReader br2 = new BufferedReader(new
FileReader("where.txt"));
                  String sCurrentLine2;
                  while ((sCurrentLine2 = br2.readLine()) != null)
                  {
                       String line2 = sCurrentLine2.toString();
                    String[] arr=line2.split(" ");
                    variableName2 = arr[0];
                    operator = arr[1];
                    valu = arr[2];
                    System.out.println("condition er variable
"+variableName2);
                 }
                  br2.close();

                  System.out.println("valu contains "+valu);

                  BufferedReader Read = new BufferedReader(new
FileReader("where1.txt"));
                  String readLine;
                  while((readLine = Read.readLine()) != null){
                       String line3 = readLine.toString();
                       String[] arr = line3.split(" ");

                       conname = arr[0];
                       conoperator= arr[1];
                       convalu = arr[1];
                       System.out.println(" 2nd condition er variable "+
conname);
                       System.out.println("condition er operator
"+conoperator);
                       System.out.println("condition er value
"+convalu);
                  }
                  Read.close();

                  BufferedReader bufr1 = new BufferedReader(new
FileReader("relation.txt"));
                  relation = bufr1.readLine();
                  System.out.println("relation type "+relation);
                  bufr1.close();
```

```
                if(counter.contains("1")){

                  if(operator.contains("==") ||
operator.contains("=")){
                if(mapi.containsKey(variableName2) &&
mapi.containsValue(Integer.parseInt(valu))){
                      context.write(new Text("result is "), new
DoubleWritable(mapi.get(why)));

                      }
                else if(maps.containsKey(variableName2) &&
maps.containsValue(valu)){
                      context.write(new Text("result is "), new
DoubleWritable(mapi.get(why)));
                      }
                }
                  else if(operator.contains(">")){
                if(mapi.containsKey(variableName2)){
                      System.out.println("map er value
"+mapi.get(variableName2));
                            System.out.println("map er value
"+Integer.parseInt(valu));

                            if(mapi.get(variableName2) >
Integer.parseInt(valu)){
                          context.write(new Text("result is "), new
DoubleWritable(mapi.get(why)));
                          }
                            }
                }
                  else if(operator.contains("<")){
                if(mapi.containsKey(variableName2)){
                         System.out.println("map er value
"+mapi.get(variableName2));
                         System.out.println("map er value
"+Integer.parseInt(valu));

                         if(mapi.get(variableName2) <
Integer.parseInt(valu)){
                          context.write(new Text("result is "), new
DoubleWritable(mapi.get(why)));
                          }
                         }
                  }
                  else if(operator.contains(">=")){
                if(mapi.containsKey(variableName2)){
                         System.out.println("map er value
"+mapi.get(variableName2));
                         System.out.println("map er value
"+Integer.parseInt(valu));
```

```
                          if(mapi.get(variableName2) >=
Integer.parseInt(valu)){
                           context.write(new Text("result is "), new
DoubleWritable(mapi.get(why)));
                           }
                          }
                   }
                   else if(operator.contains("<=")){
                 if(mapi.containsKey(variableName2)){
                          System.out.println("map er value
"+mapi.get(variableName2));
                          System.out.println("map er value
"+Integer.parseInt(valu));

                          if(mapi.get(variableName2) <=
Integer.parseInt(valu)){
                           context.write(new Text("result is "), new
DoubleWritable(mapi.get(why)));
                           }
                          }
                   }
                   else if(operator.contains("!=")){
                 if(mapi.containsKey(variableName2)){
                          System.out.println("map er value
"+mapi.get(variableName2));
                          System.out.println("map er value
"+Integer.parseInt(valu));

                          if(mapi.get(variableName2) !=
Integer.parseInt(valu)){
                           context.write(new Text("result is "), new
DoubleWritable(mapi.get(why)));
                           }
                          else if(maps.get(variableName2) != valu){
                           context.write(new Text("result is "), new
DoubleWritable(mapi.get(why)));
                          }
                          }
                   }
               }
                 else if(counter.contains("2")){
                        if(operator.contains("==") ||
operator.contains("=")){
                             if(relation.contains("&&")){
                      if((mapi.containsKey(variableName2) &&
mapi.containsValue(Integer.parseInt(valu))) &&
                           (mapi.containsKey(conname) &&
mapi.containsValue(Integer.parseInt(convalu)))){
                           context.write(new Text("result is "), new
DoubleWritable(mapi.get(why)));

                           }
```

```
                              else if((maps.containsKey(variableName2)
&& maps.containsValue(valu)) &&
                                  maps.containsKey(conname) &&
maps.containsValue(valu)){
                       context.write(new Text("result is "), new
DoubleWritable(mapi.get(why)));
                             }
                       }
                              else if(relation.contains("||")){
                                   if(operator.contains("==") ||
operator.contains("=")){

     if((mapi.containsKey(variableName2) &&
mapi.containsValue(Integer.parseInt(valu))) ||
                                      (mapi.containsKey(conname) &&
mapi.containsValue(Integer.parseInt(convalu)))){
                                      context.write(new Text("result
is "), new DoubleWritable(mapi.get(why)));

                                      }
                                       else
if((maps.containsKey(variableName2) && maps.containsValue(valu)) ||
                                           maps.containsKey(conname)
&& maps.containsValue(valu)){
                                 context.write(new Text("result is "),
new DoubleWritable(mapi.get(why)));
                                      }
                                        }
                                      }
                              }
                         else if((operator.contains("==") ||
operator.contains("=")) && (conoperator.contains("!="))){
                              if(relation.contains("&&")){
                                   if(mapi.containsKey(variableName2)
&& mapi.containsValue(Integer.parseInt(valu)) &&
                                            mapi.containsKey(conname)
!= mapi.containsValue(Integer.parseInt(convalu))){
                                        context.write(new Text("result
is "), new DoubleWritable(mapi.get(why)));

                                   }
                                   else
if((maps.containsKey(variableName2) && maps.containsValue(valu)) &&
                           maps.containsKey(conname) !=
maps.containsValue(valu)){
                                        context.write(new Text("result
is "), new DoubleWritable(mapi.get(why)));

                                   }
                              }
                              else if(relation.contains("||")){
                                   if(mapi.containsKey(variableName2)
&& mapi.containsValue(Integer.parseInt(valu)) ||
                                              mapi.containsKey(conname)
!= mapi.containsValue(Integer.parseInt(convalu))){
                                         context.write(new Text("result
is "), new DoubleWritable(mapi.get(why)));

                                   }
                                   else
if((maps.containsKey(variableName2) && maps.containsValue(valu)) ||
                           maps.containsKey(conname) !=
maps.containsValue(valu)){
                                         context.write(new Text("result
is "), new DoubleWritable(mapi.get(why)));

                                   }

                                   }

                              }


                        }
                  }


           else if(CurrentLine.contains("no")){

                context.write(new Text("result is "), new
DoubleWritable(mapi.get(why)));
           }


    }
}
```

3. MaxTemperatureReducer

```
import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.*;

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;

public class MaxTemperatureReducer extends Reducer <Text,
DoubleWritable, Text, DoubleWritable >{

     String variableName1;
```

```
      String[] Var1 = new String[500000];
      int d = 0;;

      public void reduce(Text key, Iterable<DoubleWritable> values,
Context context) throws IOException,InterruptedException
      {

            System.out.println("its reducer");

            /*Text maxCity = null;
            Text minCity = null;
            System.out.println(maxCity);

            for(LongWritable value : values){

                  String compositeString = value.toString();
                  System.out.println(compositeString);
                  String[] compositeStringArray =
compositeString.split("_");
                  System.out.println(compositeStringArray);

                  Text tempCity = new Text(compositeStringArray[0]);
                  System.out.println("Temperature of City "+tempCity);
                  long tempValue = new
Long(compositeStringArray[1]).longValue();
                  System.out.println("Temperature Value "+tempValue);


                  if(tempValue > max){
                        max = tempValue;
                        System.out.println("max is"+max);
                        maxCity = tempCity;
                        System.out.println("maxcity: "+maxCity);
                  }
                  if(min > tempValue){
                              min = tempValue;
                        System.out.println("min is "+min);
                        minCity = tempCity;
                        System.out.println("mincity: "+minCity);
                  }
            }

            System.out.println("max "+max);
            String keyText = new String("max" + "(" +
maxCity.toString() + "): " + max + "  //  " + "min" + "(" +
minCity.toString() + "):" + min);

            System.out.println("value is"+ keyText);*/


            //context.write(key,new Text(keyText));
            BufferedReader br = new BufferedReader(new
FileReader("check.txt"));
            String sCurrentLine;
            while ((sCurrentLine = br.readLine()) != null)
            {
                   String line = sCurrentLine.toString();
                   String[] arr=line.split(" ");
                   variableName1 = arr[0];
                   System.out.println("variable name of reducer
"+variableName1);
                   Var1[d] = variableName1;

                   System.out.println("reduce "+Var1[d]);

                   if(Var1[d].contains("Maximum")){
                         double maxValue = Integer.MIN_VALUE;
                              for (DoubleWritable value : values) {
                                    maxValue = Math.max(maxValue,
value.get());
                                    //System.out.println("max value
"+maxValue);
                              }
                              context.write(key, new
DoubleWritable(maxValue));
                   }
                   else if(Var1[d].contains("Minimum")){
                         double minValue = Integer.MAX_VALUE;
                         for (DoubleWritable value : values) {
                               minValue = Math.min(minValue,
value.get());
                               //System.out.println("min value
"+minValue);
                         }
                         context.write(key, new
DoubleWritable(minValue));
                   }
                   else if(Var1[d].contains("Average")){
                         double temp = 0;
                       double count = 0;
                       for (DoubleWritable value : values) {
                          temp += value.get();
                         System.out.println("each temp value "+temp);
                          count ++;
                          System.out.println("count value "+count);
                        }
                       double average = temp/count;
                       System.out.println("value of average temperature
"+average);
                        context.write(key, new DoubleWritable(average ));
                   }
            }
      }
```

```
}
```

4. FinalOutput

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import javax.swing.JOptionPane;

public class finalOutput
{
     private static String result;

       public static void main()
    {
            BufferedReader br = null;

           try
           {
                String sCurrentLine=null;
                String outputdir=null;

                br = new BufferedReader(new
FileReader("/home/tahira/Documents/Data/finalOutputDir.txt"));

                while ((sCurrentLine = br.readLine()) != null)
                {
                      outputdir = sCurrentLine.toString();
                      System.out.println(outputdir);
                }
                br.close();

                sCurrentLine=null;
                br = new BufferedReader(new
FileReader(outputdir+"/part-r-00000"));
                while ((sCurrentLine = br.readLine()) != null)
                {
                     result = sCurrentLine;
                     System.out.println(result);
                     JOptionPane.showMessageDialog (null, "Your result
is " + result, "Results",JOptionPane.PLAIN_MESSAGE);

                }
              }

           catch (IOException e)
           {
                e.printStackTrace();
           }
    }
}
```