

<b>Ex No.</b>	<b>1</b>	<b>Downloading and installing Hadoop; Understanding different Hadoop modes. Startup scripts, Configuration files.</b>
<b>Date</b>		

**AIM:-**

i) Perform setting up and Installing Hadoop in its three operating modes:

- Standalone
- Pseudo Distributed
- Fully Distributed

**DESCRIPTION:**

Hadoop is written in Java, so you will need to have Java installed on your machine, version 6 or later. Sun's JDK is the one most widely used with Hadoop, although others have been reported to work.

Hadoop runs on Unix and on Windows. Linux is the only supported production platform, but other flavors of Unix (including Mac OS X) can be used to run Hadoop for development. Windows is only supported as a development platform, and additionally requires Cygwin to run. During the Cygwin installation process, you should include the openssh package if you plan to run Hadoop in pseudo- distributed mode

**ALGORITHM****STEPS INVOLVED IN INSTALLING HADOOP IN STANDALONE MODE:-**

1. Command for installing ssh is “**sudo apt-get install ssh**”.
2. Command for key generation is **ssh-keygen -t rsa -P “ ”**.
3. Store the key into rsa.pub by using the command **cat \$HOME/.ssh/id\_rsa.pub >> \$HOME/.ssh/authorized\_keys**
4. Extract the java by using the command **tar xvfz jdk-8u60-linux-i586.tar.gz**.
5. Extract the eclipse by using the command **tar xvfz eclipse-jee-mars-R-linux-gtk.tar.gz**
6. Extract the hadoop by using the command **tar xvfz hadoop-2.7.1.tar.gz**
7. Extract the hadoop by using the command **tar xvfz hadoop-2.7.1.tar.gz**
8. Move the java to **/usr/lib/jvm/** and eclipse to **/opt/** paths. Configure the java path in the eclipse.ini file
9. Export java path and hadoop path in **./bashrc**
10. Check the installation successful or not by checking the java version and hadoop version
11. Check the hadoop instance in standalone mode working correctly or not by using an implicit hadoop jar file named as word count.
12. If the word count is displayed correctly in part-r-00000 file it means that standalone mode is installed successfully.

## ALGORITHM

### STEPS INVOLVED IN INSTALLING HADOOP IN PSEUDO DISTRIBUTED MODE:-

1. In order to install pseudo distributed mode we need to configure the hadoop configuration files which reside in the directory **/home/lendi/hadoop-2.7.1/etc/hadoop**.
2. First configure the **hadoop-env.sh** file by changing the java path.
3. Configure the **core-site.xml** which contains a property tag, it contains name and value. Name as **fs.defaultFS** and value as **hdfs://localhost:9000**

4. Configure hdfs-site.xml.
5. Configure yarn-site.xml.
6. Configure mapred-site.xml before configure the copy mapred-site.xml.template to mapred-site.xml.
7. Now format the name node by using command `hdfs namenode -format`.
8. Type the command `start-dfs.sh`, `start-yarn.sh` means that starts the daemons like NameNode, DataNode, SecondaryNameNode, ResourceManager, NodeManager.
9. Run JPS which views all daemons. Create a directory in the hadoop by using command `hdfs dfs -mkdir /csedir` and enter some data into `lendi.txt` using command `nano lendi.txt` and copy from local directory to hadoop using command `hdfs dfs -copyFromLocal lendi.txt /csedir/` and run sample jar file `wordcount` to check whether pseudo distributed mode is working or not.
10. Display the contents of file by using command `hdfs dfs -cat /newdir/part-r-00000`.

## FULLY DISTRIBUTED MODE INSTALLATION:

### ALGORITHM

1. Stop all single node clusters

```
$stop-all.sh
```

2. Decide one as NameNode (Master) and remaining as DataNodes (Slaves).
3. Copy public key to all three hosts to get a password less SSH access

```
$ssh-copy-id -I $HOME/.ssh/id_rsa.pub lendi@15sys24
```

4. Configure all Configuration files, to name Master and Slave Nodes.

```
$cd $HADOOP_HOME/etc/hadoop
```

\$ nano core-site.xml

\$ nano hdfs-site.xml

5. Add hostnames to file slaves and save it.

\$ nano slaves

6. Configure \$ nano yarn-site.xml

7. Do in Master Node

\$ hdfs namenode –format

\$ start-dfs.sh

\$start-yarn.sh

8. Format NameNode

9. Daemons Starting in Master and Slave Nodes

10. End

## **INPUT**

ubuntu @localhost> jps

## **OUTPUT:**

Data node, name node Secondary name node,  
NodeManager, Resource Manager

**RESULT:**

Hence Installation of Hadoop in different modes were studied.

<b>Ex No.</b>	<b>2</b>	<b>Hadoop Implementation of file management tasks, such as Adding files and directories, Retrieving files and Deleting files</b>
<b>Date</b>		

**AIM:-**

Implement the following file management tasks in Hadoop:

- Adding files and directories
- Retrieving files
- Deleting Files

**DESCRIPTION:-**

HDFS is a scalable distributed filesystem designed to scale to petabytes of data while running on top of the underlying filesystem of the operating system. HDFS keeps track of where the data resides in a network by associating the name of its rack (or network switch) with the dataset. This allows Hadoop to efficiently schedule tasks to those nodes that contain data, or which are nearest to it, optimizing bandwidth utilization. Hadoop provides a set of command line utilities that work similarly to the Linux file commands, and serve as your primary interface with HDFS. We're going to have a look into HDFS by interacting with it from the command line. We will take a look at the most common file management tasks in Hadoop, which include:

- Adding files and directories to HDFS
- Retrieving files from HDFS to local filesystem
- Deleting files from HDFS

**ALGORITHM:-****SYNTAX AND COMMANDS TO ADD, RETRIEVE AND DELETE DATA FROM HDFS****Step-1****Adding Files and Directories to HDFS**

Before you can run Hadoop programs on data stored in HDFS, you'll need to put the data into HDFS first. Let's create a directory and put a file in it. HDFS has a default working directory of /user/\$USER, where \$USER is your login user name. This directory isn't automatically created for you, though, so let's create it with the mkdir command. For the purpose of illustration, we use chuck. You should substitute your user name in the example commands.

```
hadoop fs -mkdir /user/chuck
```

```
hadoop fs -put example.txt
```

```
hadoop fs -put example.txt /user/chuck
```

**Step-2****Retrieving Files from HDFS**

The Hadoop command get copies files from HDFS back to the local filesystem. To retrieve example.txt, we can run the following command:

```
hadoop fs -cat example.txt
```

**Step-3****Deleting Files from HDFS**

```
hadoop fs -rm example.txt
```

- Command for creating a directory in hdfs is “**hdfs dfs –mkdir /lendicse**”. Adding
- directory is done through the command “**hdfs dfs –put lendi\_english /**”.

#### Step-4

#### Copying Data from NFS to HDFS

Copying from directory command is “**hdfs dfs –copyFromLocal**

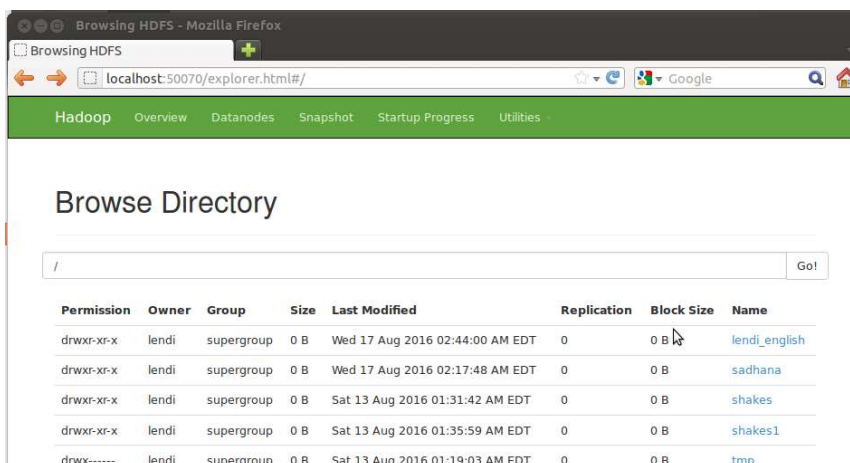
**/home/lendi/Desktop/shakes/glossary /lendicse/**”

- View the file by using the command “**hdfs dfs –cat /lendi\_english/glossary**” Command
- for listing of items in Hadoop is “**hdfs dfs –ls hdfs://localhost:9000/**”. Command for
- Deleting files is “**hdfs dfs –rm r /karthee**

#### SAMPLE INPUT:

Input as any data format of type structured, Unstructured or Semi Structured

#### EXPECTED OUTPUT:



Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxr-xr-x	lendi	supergroup	0 B	Wed 17 Aug 2016 02:44:00 AM EDT	0	0 B	<a href="#">lendi_english</a>
drwxr-xr-x	lendi	supergroup	0 B	Wed 17 Aug 2016 02:17:48 AM EDT	0	0 B	<a href="#">sadhana</a>
drwxr-xr-x	lendi	supergroup	0 B	Sat 13 Aug 2016 01:31:42 AM EDT	0	0 B	<a href="#">shakes</a>
drwxr-xr-x	lendi	supergroup	0 B	Sat 13 Aug 2016 01:35:59 AM EDT	0	0 B	<a href="#">shakes1</a>
drwx-----	lendi	supergroup	0 B	Sat 13 Aug 2016 01:19:03 AM EDT	0	0 B	<a href="#">tmp</a>



**RESULT :**

Thus the various file management tasks in Hadoop were implemented.

Ex No.	3	Implementation of matrix Multiplication with Hadoop map reduce
Date		

**AIM:-**

Write a Map Reduce Program that implements Matrix Multiplication.

**DESCRIPTION:**

We can represent a matrix as a relation (table) in RDBMS where each cell in the matrix can be represented as a record (i,j,value). As an example let us consider the following matrix and its representation. It is important to understand that this relation is a very inefficient relation if the matrix is dense. Let us say we have 5 Rows and 6 Columns , then we need to store only 30 values. But if you consider above relation we are storing 30 rowid, 30 col\_id and 30 values in other sense we are tripling the data. So a natural question arises why we need to store in this format ? In practice most of the matrices are sparse matrices . In sparse matrices not all cells used to have any values , so we don't have to store those cells in DB. So this turns out to be very efficient in storing such matrices.

**MapReduceLogic**

Logic is to send the calculation part of each output cell of the result matrix to a reducer.

So in matrix multiplication the first cell of output elements from row 0 of the matrix A and elements from col 0 of matrix B. To do the computation of value in the output cell (0,0) of resultant matrix in a separate reducer we need to use (0,0) as output key of mapphase and value should have array of values from row 0 of matrix A and column 0 of matrix B.

Hopefully this picture will explain the point. So in this algorithm output from map phase should be having a  $\langle \text{key}, \text{value} \rangle$ , where key represents the output cell location (0,0), (0,1) etc.. and value will be list of all values required for reducer to do computation.

Let us take an example for calculating value at output cell (00). Here we need to collect values from row 0 of matrix A and col 0 of matrix B in the map phase and pass (0,0) as key. So a single reducer can do the calculation.

## ALGORITHM

We assume that the input files for A and B are streams of (key,value) pairs in sparse matrix format, where each key is a pair of indices (i,j) and each value is the corresponding matrix element value. The output files for matrix  $C=A*B$  are in the same format.

We have the following input parameters:

The path of the input file or directory for matrix A.

The path of the input file or directory for matrix B.

The path of the directory for the output files for matrix C.

strategy = 1, 2, 3 or 4.

R = the number of reducers.

I = the number of rows in A and C.

K = the number of columns in A and rows in B.

J = the number of columns in B and C.

IB = the number of rows per A block and C block.

KB = the number of columns per A block and rows per B block.

JB = the number of columns per B block and C block.

Note that in all the strategies the memory footprint of both the mappers and the reducers is flat at

scale.

Note that the strategies all work reasonably well with both dense and sparse matrices. For sparse matrices we do not emit zero elements. That said, the simple pseudo-code for multiplying the individual blocks shown here is certainly not optimal for sparse matrices. As a learning exercise, our focus here is on mastering the MapReduce complexities, not on optimizing the sequential matrix multiplication algorithm for the individual blocks.

### STEPS

1. setup ()
2. var NIB = (I-1)/IB+1
3. var NKB = (K-1)/KB+1
4. var NJB = (J-1)/JB+1
5. map (key, value)
6. if from matrix A with key=(i,k) and value=a(i,k)
7. for  $0 \leq j_b < NJB$
8. emit (i/IB, k/KB, j\_b, 0), (i mod IB, k mod KB, a(i,k))
9. if from matrix B with key=(k,j) and value=b(k,j)
10. for  $0 \leq i_b < NIB$   
emit (i\_b, k/KB, j/JB, 1), (k mod KB, j mod JB, b(k,j))

Intermediate keys (i\_b, k\_b, j\_b, m) sort in increasing order first by i\_b, then by k\_b, then by j\_b, then by m. Note that m = 0 for A data and m = 1 for B data.

**The partitioner maps intermediate key (i\_b, k\_b, j\_b, m) to a reducer r as follows:**

11.  $r = ((ib * JB + jb) * KB + kb) \bmod R$

12. These definitions for the sorting order and partitioner guarantee that each reducer  $R[ib, kb, jb]$  receives the data it needs for blocks  $A[ib, kb]$  and  $B[kb, jb]$ , with the data for the A block

immediately preceding the data for the B block.

13. var A = new matrix of dimension  $IB \times KB$

14. var B = new matrix of dimension  $KB \times JB$

15. var siv = -1

16. var skb = -1

**Reduce (key, valueList)**

17. if key is (ib, kb, jb, 0)

18. // Save the A block.

19. sib = ib

20. skb = kb

21. Zero matrix A

22. for each value = (i, k, v) in valueList  $A(i, k) = v$

23. if key is (ib, kb, jb, 1)

24. if  $ib \neq sib$  or  $kb \neq skb$  return //  $A[ib, kb]$  must be zero!

25. // Build the B block.

26. Zero matrix B

27. for each value = (k, j, v) in valueList  $B(k, j) = v$

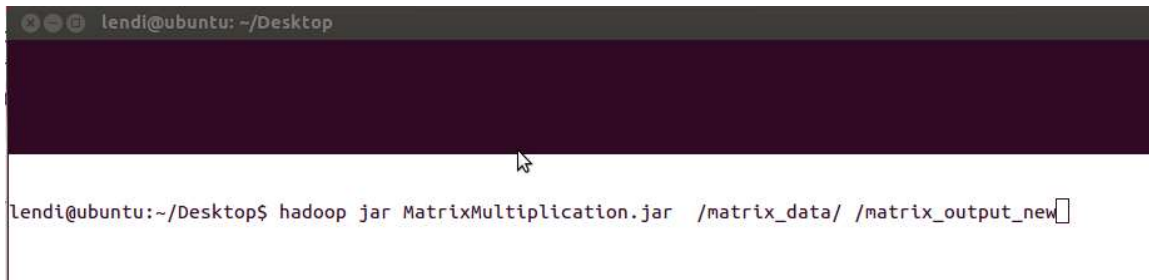
28. // Multiply the blocks and emit the result.

29. ibase =  $ib * IB$

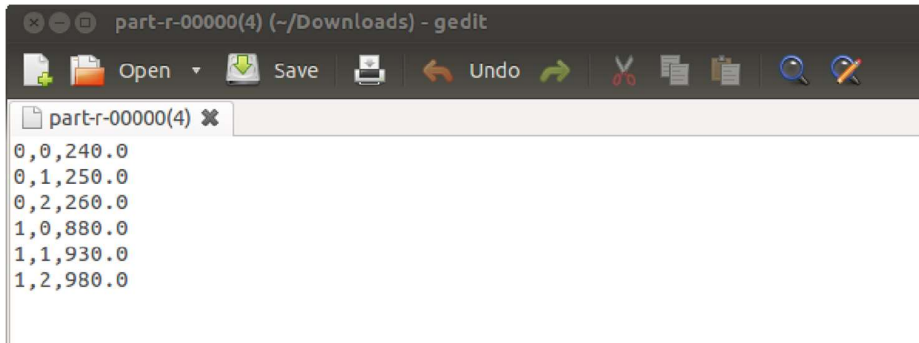
30.  $jbase = j_b * JB$
31. for  $0 \leq i < \text{row dimension of A}$
32. for  $0 \leq j < \text{column dimension of B}$
33.  $sum = 0$
34. for  $0 \leq k < \text{column dimension of A} = \text{row dimension of B}$ 
  - a.  $sum += A(i,k) * B(k,j)$
35. if  $sum \neq 0$  emit  $(ibase+i, jbase+j)$ ,  $sum$

**INPUT:-**

Set of Data sets over different Clusters are taken as Rows and Columns

**OUTPUT:-**

```
lendi@ubuntu: ~/Desktop
lendi@ubuntu:~/Desktop$ hadoop jar MatrixMultiplication.jar /matrix_data/ /matrix_output_new
```



The screenshot shows a gedit text editor window titled "part-r-00000(4) (~/.Downloads) - gedit". The window contains a list of floating-point numbers, each formatted with commas as thousands separators. The numbers are: 0,0,240.0, 0,1,250.0, 0,2,260.0, 1,0,880.0, 1,1,930.0, and 1,2,980.0. The editor's toolbar includes icons for Open, Save, Print, Undo, Redo, Cut, Copy, Paste, Find, and Replace.

```
part-r-00000(4) ✕  
0,0,240.0  
0,1,250.0  
0,2,260.0  
1,0,880.0  
1,1,930.0  
1,2,980.0
```

**RESULT:**

Thus the Matrix Multiplication using Map Reduce has been implemented.

<b>Ex No.</b>	<b>4</b>	<b>Run a basic Word Count Map Reduce program to understand Map Reduce Paradigm</b>
<b>Date</b>		

**AIM:-**

Run a basic Word Count Map Reduce Program to understand Map Reduce Paradigm

**DESCRIPTION:--**

MapReduce is the heart of Hadoop. It is this programming paradigm that allows for massive scalability across hundreds or thousands of servers in a Hadoop cluster. The MapReduce concept is fairly simple to understand for those who are familiar with clustered scale-out data processing solutions. The term MapReduce actually refers to two separate and distinct tasks that Hadoop programs perform. The first is the map job, which takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key/value pairs). The reduce job takes the output from a map as input and combines those data tuples into a smaller set of tuples. As the sequence of the name MapReduce implies, the reduce job is always performed after the map job.

**ALGORITHM****MAPREDUCE PROGRAM**

WordCount is a simple program which counts the number of occurrences of each word in a given text input data set. WordCount fits very well with the MapReduce programming model making it a great example to understand the Hadoop Map/Reduce programming style. Our implementation consists of three main parts:

1. Mapper
2. Reducer
3. Driver



**Step-1. Write a Mapper**

A Mapper overrides the `map()` function from the Class `"org.apache.hadoop.mapreduce.Mapper"` which provides `<key, value>` pairs as the input. A Mapper implementation may output `<key,value>` pairs using the provided Context .

Input value of the WordCount Map task will be a line of text from the input data file and the key would be the line number `<line_number, line_of_text>` . Map task outputs `<word, one>` for each word in the line of text.

**Pseudo-code**

```
void Map (key, value){  
  
    for each word x in value:  
        output.collect(x, 1);  
}
```

**Step-2. Write a Reducer**

A Reducer collects the intermediate `<key,value>` output from multiple map tasks and assemble a single result. Here, the WordCount program will sum up the occurrence of each word to pairs as `<word, occurrence>`.

**Pseudo-code**

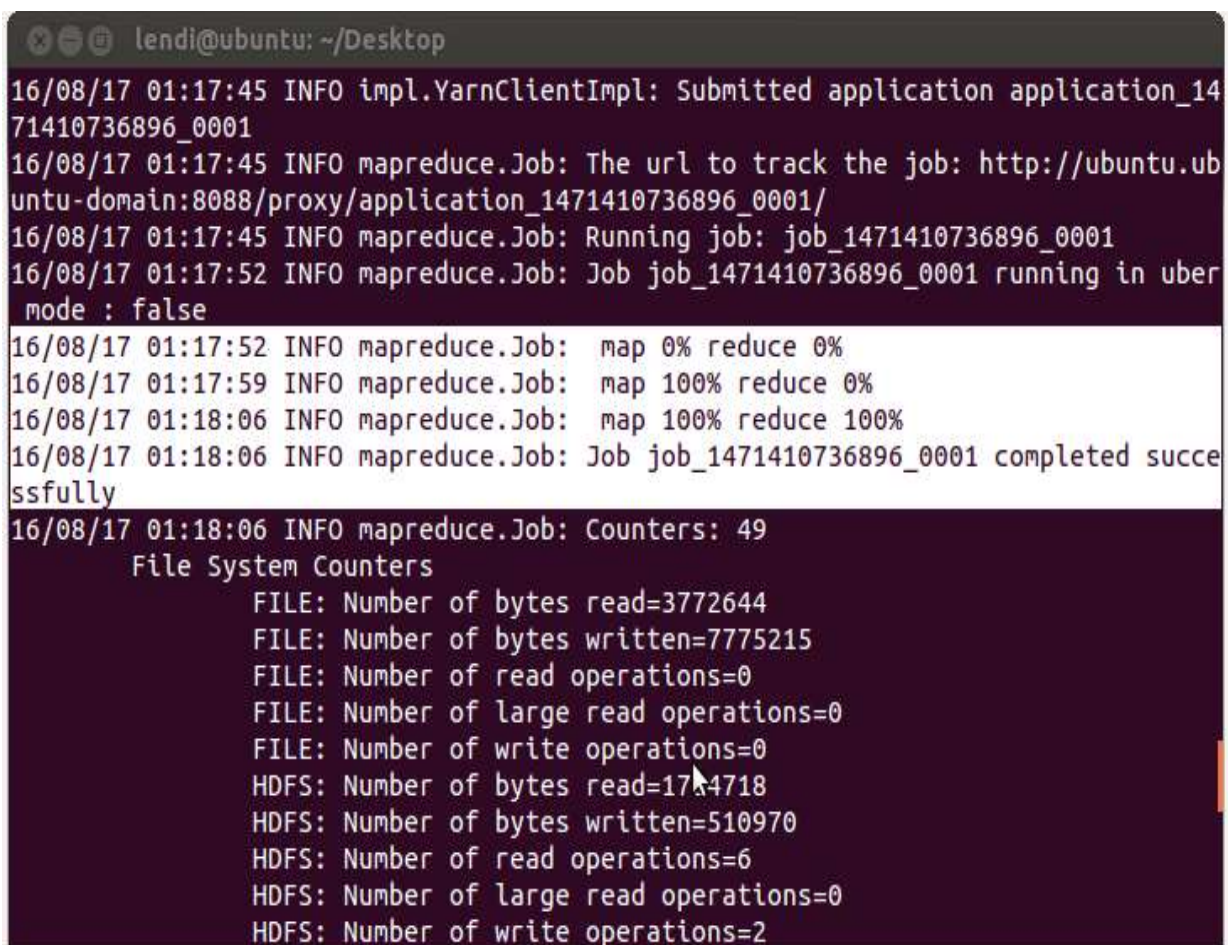
```
void Reduce (keyword, <list of value>){  
    for each x in <list of value>:  
        sum+=x;  
    final_output.collect(keyword, sum);
```

**Step-3. Write Driver**

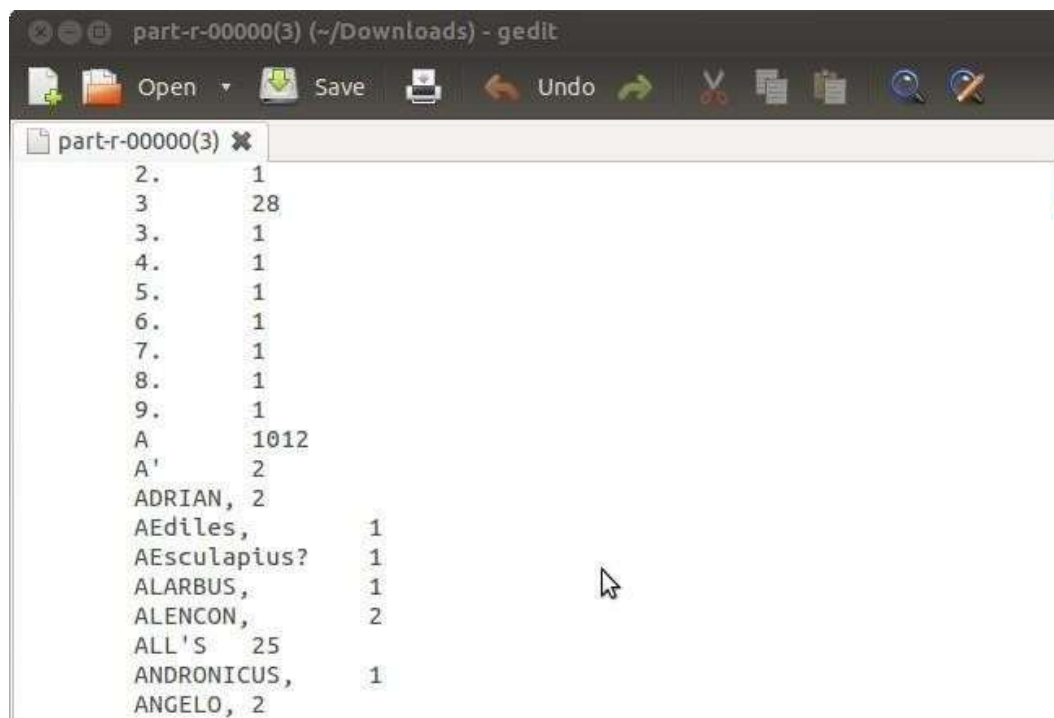
The Driver program configures and run the MapReduce job. We use the main program to perform basic configurations such as:

**INPUT:-**

Set of Data Related Shakespeare Comedies, Glossary, Poems

**OUTPUT:-**A terminal window screenshot showing the execution of a MapReduce job. The window title is 'lendi@ubuntu: ~/Desktop'. The logs show the job submission, progress updates, and completion status. The job is identified as 'application\_1471410736896\_0001'. The progress shows 100% map and reduce completion. The final output shows counters for the job, including file system and HDFS statistics.

```
lendi@ubuntu: ~/Desktop
16/08/17 01:17:45 INFO impl.YarnClientImpl: Submitted application application_1471410736896_0001
16/08/17 01:17:45 INFO mapreduce.Job: The url to track the job: http://ubuntu.ubuntu-domain:8088/proxy/application_1471410736896_0001/
16/08/17 01:17:45 INFO mapreduce.Job: Running job: job_1471410736896_0001
16/08/17 01:17:52 INFO mapreduce.Job: Job job_1471410736896_0001 running in uber mode : false
16/08/17 01:17:52 INFO mapreduce.Job:  map 0% reduce 0%
16/08/17 01:17:59 INFO mapreduce.Job:  map 100% reduce 0%
16/08/17 01:18:06 INFO mapreduce.Job:  map 100% reduce 100%
16/08/17 01:18:06 INFO mapreduce.Job: Job job_1471410736896_0001 completed successfully
16/08/17 01:18:06 INFO mapreduce.Job: Counters: 49
    File System Counters
        FILE: Number of bytes read=3772644
        FILE: Number of bytes written=7775215
        FILE: Number of read operations=0
        FILE: Number of large read operations=0
        FILE: Number of write operations=0
        HDFS: Number of bytes read=174718
        HDFS: Number of bytes written=510970
        HDFS: Number of read operations=6
        HDFS: Number of large read operations=0
        HDFS: Number of write operations=2
```



The screenshot shows a gedit text editor window titled "part-r-00000(3) (~/.Downloads) - gedit". The window contains a list of words and their corresponding counts, formatted as follows:

2.	1
3	28
3.	1
4.	1
5.	1
6.	1
7.	1
8.	1
9.	1
A	1012
A'	2
ADRIAN,	2
AEdiles,	1
AEsculapius?	1
ALARBUS,	1
ALENCON,	2
ALL'S	25
ANDRONICUS,	1
ANGELO,	2

**RESULT:**

Thus Word Count Map Map Reduce program has been implemented.

<b>Ex No.</b>	<b>5</b>	<b>Implementation of K-means clustering using Map Reduce</b>
<b>Date</b>		

**AIM :**

To Implement K –means clustering using Map Reduce

**DESCRIPTION:**

MapReduce is a shared-memory model, the centroids can be shared among iterations.

To share the centroids, a file can be created on HDFS to include the initial K centroids (in iteration 0) and the updated centroids in each iteration.

You can create a FileSystem in your program's Configuration()

**Mapper class (i.e., WordCount mapper)**

- ☐ Extends MapReduceBase
- ☐ Implements Mapper <LongWritable, Text, Text, IntWritable>
- ☐

Should implement the map function:

```
public void map (LongWritable key, Text value, OutputCollector<Text,IntWritable>
output,Reporter report)
```

The map function takes a key, value, output collector and a reporter:

- Key: represents the offset in the file (in Figure 1, key1)
- Value: The value of the offset (In Figure 1, "Tamim is" for offset 0)
- Output Collector: collects the output from the map function and feeds it into the Reduce phase. The type of the output collector depends on the <key,value> pair type of the map output (in Figure 1, the type is <Text, IntWritable>)
- Reporter: reports any failure on the mapper

**Reducer class (i.e., WordCountReducer)**

Extends MapReduceBase

Implements Reducer Reducer <Text, IntWritable, Text, LongWritable>

Implements the reducefunction with the following parameters:

- Key: input key where the data is combined together (in case of word count, the key is the word).
- Iterator for the values: iterates over the values assigned for a given key.
- Output Collector: we output a word and its count, with types of output <key, value> pair as  
  
<Text, LongWritable>
- Reporter: reports any failure on the reduce

**Main configurations**

There is a set of configurations that should be considered in the main function, before running the job:

1. Defining a new job configuration: new JobConf(class instance)
2. Set the mapper and the reducer classes
3. Define the types of the map and reduce <key, value> output types:

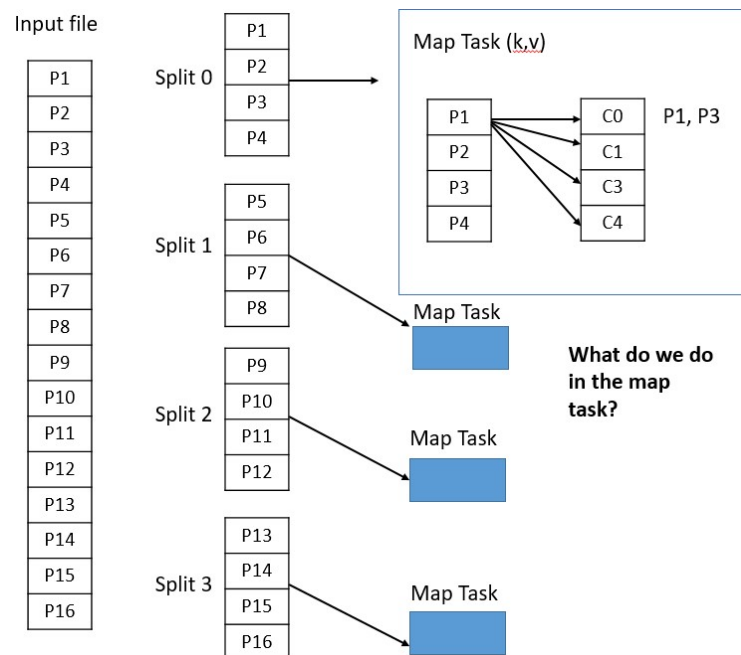
```
conf.setMapOutputKeyClass(Text.class);  
conf.setMapOutputValueClass(IntWritable.class);  
conf.setOutputKeyClass(Text.class);  
conf.setOutputValueClass(LongWritable.class);
```

4. Run the job

```
JobClient.runJob(conf);
```

**KMeans on MapReduce****Map Phase**

- 1) The mapper reads the data input file and gets the centroids from last iteration (or initial iteration)
- 2) The file is chunked and fed into the mapfunction.
- 3) You should have predefined the mapper <key, value> pairs with the key being the offset and the value is a point read from the file.
- 4) For each find the nearest centroid and assign the point to it (Figure 2 illustrates the process)



*Figure 2 Kmeans Map Task*

### Combiner

The output of the map phase is huge (= total number of points) and we shall need to use a combiner to minimize the size of the data before sending it to the reducer. Let's see an example of a combiner in WordCount. Figure 3 shows a WordCount example without a combiner as opposed to Figure 3 with a combiner. The number of keys processed by the Reduce task is reduced from 9 to 4. This will be helpful in our KMeans implementation so that we minimize the number of points to be processed by the Reduce phase. The combiner calculates the average of the data instances for each cluster id, along with the number of the instances. It outputs (cluster id, (intermediate cluster centroid, number of instances)).

**To define a combiner, you set it in your configuration**

**as:**`job.setCombinerClass(IntSumReducer.class);`

**where IntSumReducer is a Reducer class.**

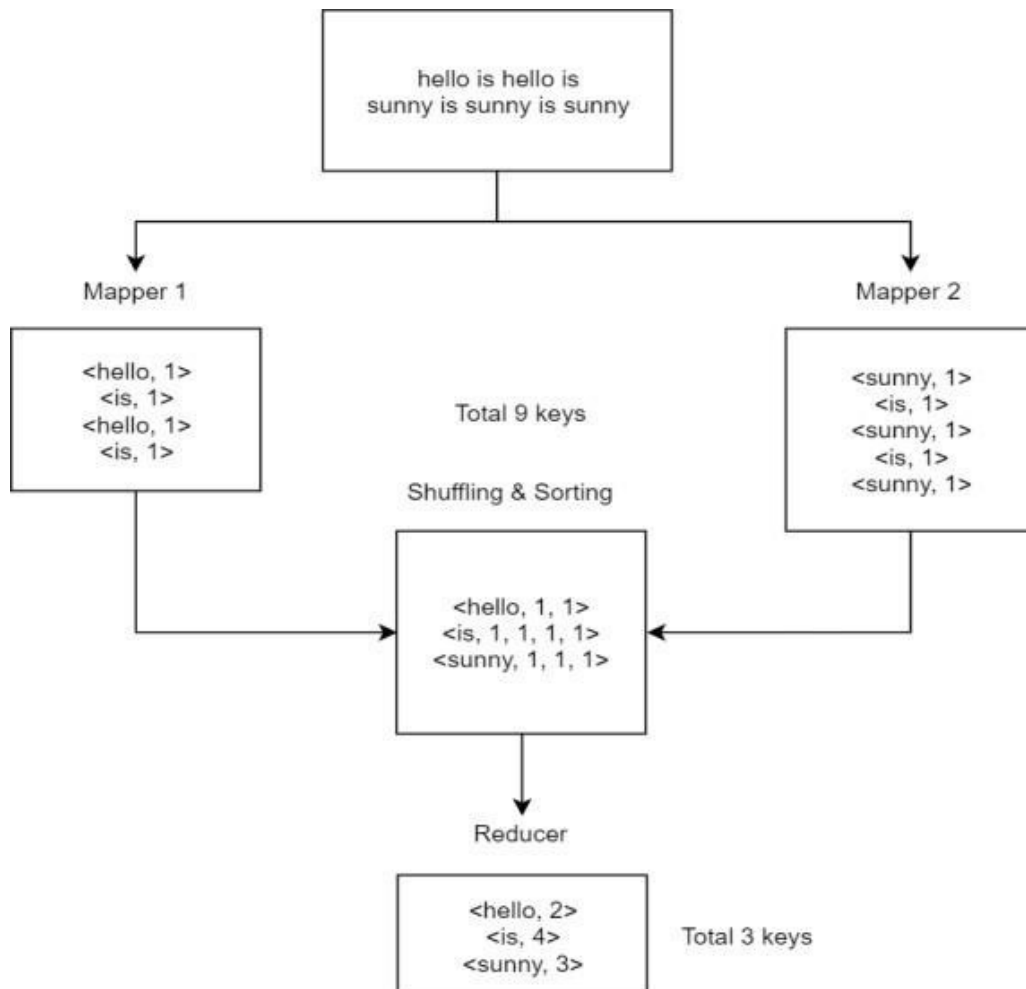


Figure 3 WordCount without a combiner

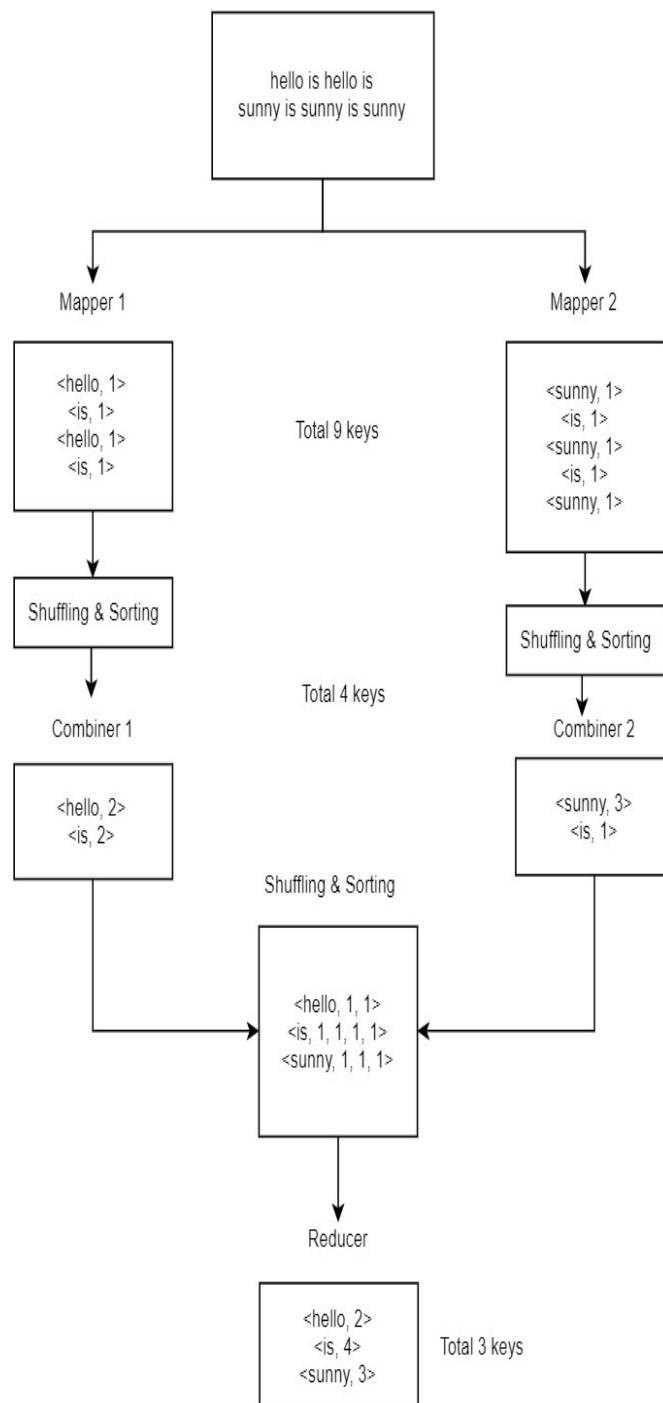


Figure 4 WordCount with a combiner



## Reduce Phase

In the reduce phase, and based on the output of the combiner, you need to recalculate the centroids by iterating over the values and output the intermediate centroids. Since we are sharing the centroids among iterations, the centroid values have to be updated using the configuration file as stated in the previous section.

## Running the Job

The main function involves two parts:

1. Configurations
2. Running multiple iteration jobs using the above Mapper + Combiner + Reducer.

```
int iteration = 0;
// counter from the previous running import job
long counter = job.getCounters().findCounter(Reducer.Counter.CONVERGED).getValue();
iteration++;
while (counter > 0) {
    conf = new Configuration();
    conf.set("recursion.iter", iteration + "");
    job = new Job(conf);
    job.setJobName("KMeans " + iteration);
    // ...job.set Mapper, Combiner, Reducer... //
    // always take the output from last iteration as the input
    in = new Path("files/kmeans/iter_" + (iteration - 1) + "/" );
    out = new Path("files/kmeans/iter_" + iteration);
    //... job.set Input, Output... //
    // wait for completion and update the counter
    job.waitForCompletion(true);
    iteration++;
    counter = job.getCounters().findCounter(Reducer.Counter.CONVERGED).getValue();
}
```

You can define the counter in the reducer class and update it as necessary:

```
public enum Counter{  
    CONVERGED //name of the counter  
}  
  
context.getCounter(Counter.CONVERGED).increment(1);
```

### Command Line Instructions

#### How to run your code?

1. Create a folder for the .class files of your application:  
`$ mkdir KMeans_Classes`
2. Compile your KMeans program (where Kmeans.java is where you have your implementation)  
`$ javac -classpath $(hadoop classpath) -d KMeans_Classes Kmeans.java`
3. Create the jar file required by Hadoop to run your application using the following command:  
`$ jar -cvf Kmeans.jar -C KMeans_Classes/ .`
4. Create an input directory in HDFS using the following command:  
`hadoop dfs -mkdir ./KmeansInput` (for new versions)
5. Copy your points file to HDFS input directory using the following command:  
`hadoop dfs - copyFromLocal points.txt ./KmeansInput`
6. You can check if the points file was copied correctly by:  
`hadoop dfs -ls ./ KmeansInput`
7. Run your Kmeans application (where KMeans is the classname, if Kmeans Output is not created, please create it as you created the KmeansInput)  
`hadoop jar Kmeans.jar KMeans ./KmeansInput ./KmeansOutput 100 3 5`

8. You can check your output file: `hadoop`  
`dfs -ls ./KmeansOutput`

**RESULT:**

Thus K-means clustering has been implemented using Map Reduce.

<b>Ex No.</b>	<b>6</b>	<b>Installation of Hive along with practice examples.</b>
<b>Date</b>		

**AIM:**

Install and Run Hive then use Hive to Create, alter and drop databases, tables, views, functions and Indexes.

**DESCRIPTION :**

Hive, allows SQL developers to write Hive Query Language (HQL) statements that are similar to standard SQL statements; now you should be aware that HQL is limited in the commands it understands, but it is still pretty useful. HQL statements are broken down by the jobs and executed across a Hadoop cluster. Hive looks very much like Hive service into MapReduce like traditional database code with SQL access. However, because Hive is based on Hadoop and MapReduce operations, there are several key differences. The first is that Hadoop is intended for long sequential scans, and because Hive is based on Hadoop, you can expect queries to have a very high latency (many minutes). This means that Hive would not be appropriate for applications that need very fast response times, as you would expect with a database such as DB2. Finally, Hive is read-based and therefore not appropriate for transaction processing that typically involves a high percentage of write operations.

**ALGORITHM:****Apache HIVE INSTALLATION STEPS**

- 1) Install MySQL-Server  
  
Sudo apt-get install mysql-server
- 2) Configuring MySQL UserName and Password
- 3) Creating User and granting all Privileges

Mysql –uroot –proot

Create user <USER\_NAME> identified by <PASSWORD>

- 4) Extract and Configure Apache Hive

tar xvfz apache-hive-1.0.1.bin.tar.gz

- 5) Move Apache Hive from Local directory to Home directory

- 6) Set CLASSPATH in bashrc

Export HIVE\_HOME = /home/apache-hive

Export PATH = \$PATH:\$HIVE\_HOME/bin

- 7) Configuring hive-default.xml by adding My SQL Server Credentials

<property>

<name>javax.jdo.option.ConnectionURL</name>

<value>

jdbc:mysql://localhost:3306/hive?createDatabaseIfNotExist=true

</value>

</property>

<property>

<name>javax.jdo.option.ConnectionDriverName</name>

<value>com.mysql.jdbc.Driver</value>

</property>

<property>

<name>javax.jdo.option.ConnectionUserName</name>

<value>hadoop</value>

</property>

<property>

```
<name>javax.jdo.option.ConnectionPassword</name>
```

```
<value>hadoop</value>
```

```
</property>
```

- 8) Copying mysql-java-connector.jar to hive/lib directory.

## SYNTAX for HIVE Database

### Operations

#### DATABASE Creation

*CREATE DATABASE|SCHEMA [IF NOT EXISTS] <database name>*

#### DropDatabaseStatement

*DROP DATABASE Statement DROP (DATABASE|SCHEMA) [IF EXISTS]*

*database\_name [RESTRICT|CASCADE];*

#### Creating and Dropping Table in HIVE

*CREATE [TEMPORARY] [EXTERNAL] TABLE [IF NOT EXISTS] [db\_name.] table\_name*

*[(col\_name data\_type [COMMENT col\_comment], ...)]*

*[COMMENT table\_comment] [ROW FORMAT row\_format] [STORED AS file\_format]*

#### Loading Data into table log\_data

#### Syntax:

*LOAD DATA LOCAL INPATH '<path>/u.data' OVERWRITE INTO TABLE u\_data;*

**Alter Table in HIVE****Syntax**

*ALTER TABLE name RENAME TO new\_name*

*ALTER TABLE name ADD COLUMNS (col\_spec[, col\_spec ...])*

*ALTER TABLE name DROP [COLUMN] column\_name*

*ALTER TABLE name CHANGE column\_name new\_name new\_type*

*ALTER TABLE name REPLACE COLUMNS (col\_spec[, col\_spec ...])*

**Creating and Dropping View**

*CREATE VIEW [IF NOT EXISTS] view\_name [(column\_name [COMMENT  
column\_comment], ...)] [COMMENT table\_comment] AS SELECT ..*

**Dropping****ViewSyntax:**

*DROP VIEW view\_name*

**Functions in HIVE**

*String Functions:- round(), ceil(), substr(), upper(), reg\_exp() etc*

*Date and Time Functions:- year(), month(), day(), to\_date() etc*

*Aggregate Functions :- sum(), min(), max(), count(), avg() etc*

**INDEXES**

*CREATE INDEX index\_name ON TABLE base\_table\_name (col\_name, ...)*

*AS 'index.handler.class.name'*

*[WITH DEFERRED REBUILD]*

*[IDXPROPERTIES (property\_name=property\_value, ...)]*

*[IN TABLE index\_table\_name]*

*[PARTITIONED BY (col\_name, ...)]*

```
[  
[ ROW FORMAT ...] STORED AS ...  
| STORED BY ...  
]  
  
[LOCATION hdfs_path]  
[TBLPROPERTIES (...)]
```

### Creating Index

```
CREATE INDEX index_ip ON TABLE log_data(ip_address) AS  
'org.apache.hadoop.hive.ql.index.compact.CompactIndexHandler' WITH DEFERRED  
REBUILD;
```

### Altering and Inserting Index

```
ALTER INDEX index_ip_address ON log_data REBUILD;
```

### Storing Index Data in Metastore

```
SET
```

```
hive.index.compact.file=/home/administrator/Desktop/big/metastore_db/tmp/index_ipaddress_re  
sult;
```

```
SET
```

```
hive.input.format=org.apache.hadoop.hive.ql.index.compact.HiveCompactIndexInputFormat;
```

### Dropping Index

```
DROP INDEX INDEX_NAME on TABLE_NAME;
```



## INPUT

Input as Web Server Log Data

## OUTPUT

```

administrator@ubuntu: ~
d yet. Please use TIMESTAMP instead
hive> create table log_data(l_date string,l_time string,s_sitename string,s_compu
tername string,l_uri string,uri_query string,ip_address string,user_agent strin
g,status1 int,status2 int,s_bytes int,c_bytes int,time_taken int);
OK
Time taken: 0.331 seconds
hive> show tables;
OK
log_data
Time taken: 0.074 seconds, Fetched: 1 row(s)
hive> desc log_data;
OK
l_date                string                None
l_time                string                None
s_sitename             string                None
s_computername         string                None
l_uri                 string                None
uri_query              string                None
ip_address             string                None
user_agent             string                None
status1                int                   None
status2                int                   None
s_bytes                int                   None
c_bytes                int                   None

```

```

administrator@ubuntu: ~
0.6.20.6 Mozilla/4.0+(compatible;+MSIE+7.0;+Windows+NT+6.1;+Trident/4.0;+GTB7.5;+SLC
R+2.0.50727;+.NET+CLR+3.5.30729;+.NET+CLR+3.0.30729;+Media+Center+PC+6.0;+InfoPath.2) 304
11 498 0
2014-12-23 23:08:38 W3SVC1 NEWINTSERV2 /trf/elastic/images/small/pic3.jpg
0.6.20.6 Mozilla/4.0+(compatible;+MSIE+7.0;+Windows+NT+6.1;+Trident/4.0;+GTB7.5;+SLC
R+2.0.50727;+.NET+CLR+3.5.30729;+.NET+CLR+3.0.30729;+Media+Center+PC+6.0;+InfoPath.2) 304
10 497 0
2014-12-23 23:16:07 W3SVC1 NEWINTSERV2 /trf/elastic/css/demo.css - 10.
ozilla/4.0+(compatible;+MSIE+7.0;+Windows+NT+6.0;+SLCC1;+.NET+CLR+2.0.50727;+.NET+
CLR+1.1.4322;+InfoPath.2) 304 0 210 458 0
2014-12-23 23:16:07 W3SVC1 NEWINTSERV2 /trf/elastic/css/elastislide.css -
0.22 Mozilla/4.0+(compatible;+MSIE+7.0;+Windows+NT+6.0;+SLCC1;+.NET+CLR+2.0.50727;+.NET+
06;+.NET+CLR+1.1.4322;+InfoPath.2) 304 0 210 465 0
2014-12-23 23:16:07 W3SVC1 NEWINTSERV2 /trf/elastic/images/small/pic11.jpg
0.3.20.22 Mozilla/4.0+(compatible;+MSIE+7.0;+Windows+NT+6.0;+SLCC1;+.NET+CLR+2.0.5072
+3.0.04506;+.NET+CLR+1.1.4322;+InfoPath.2) 304 0 211 469 0
2014-12-23 23:16:07 W3SVC1 NEWINTSERV2 /trf/elastic/images/small/pic12.jpg
0.3.20.22 Mozilla/4.0+(compatible;+MSIE+7.0;+Windows+NT+6.0;+SLCC1;+.NET+CLR+2.0.5072
+3.0.04506;+.NET+CLR+1.1.4322;+InfoPath.2) 304 0 211 469 0
2014-12-23 23:16:07 W3SVC1 NEWINTSERV2 /trf/elastic/images/small/pic10.jpg
0.3.20.22 Mozilla/4.0+(compatible;+MSIE+7.0;+Windows+NT+6.0;+SLCC1;+.NET+CLR+2.0.5072
+3.0.04506;+.NET+CLR+1.1.4322;+InfoPath.2) 304 0 211 469 0
2014-12-23 23:16:07 W3SVC1 NEWINTSERV2 /trf/elastic/images/small/pic9.jpg
0.3.20.22 Mozilla/4.0+(compatible;+MSIE+7.0;+Windows+NT+6.0;+SLCC1;+.NET+CLR+2.0.5072
+3.0.04506;+.NET+CLR+1.1.4322;+InfoPath.2) 304 0 210 467 0
2014-12-23 23:16:07 W3SVC1 NEWINTSERV2 /trf/elastic/images/small/pica.jpg

```

```

administrator@ubuntu: ~
hive> select * from index_ip;
FAILED: SemanticException [Error 10001]: Line 1:14 Table not found 'index_ip'
hive> INSERT OVERWRITE DIRECTORY '/home/administrator/Desktop/hive_data/index_test_result' SELECT '
bucketname', '_offsets' FROM lendi_db.lendi_db__log_data_index_ip__ where ip_address='141.0.11.19
9';
Total MapReduce jobs = 3
Launching Job 1 out of 3
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1476764326039_0014, Tracking URL = http://ubuntu.ubuntu-domain:8088/proxy/applica
tion_1476764326039_0014/
Kill Command = /home/administrator/hadoop-2.7.1/bin/hadoop job -kill job_1476764326039_0014
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 0
2016-10-18 02:16:23,240 Stage-1 map = 0%, reduce = 0%
2016-10-18 02:16:27,406 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 1.32 sec
2016-10-18 02:16:28,442 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 1.32 sec
2016-10-18 02:16:29,472 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 1.32 sec
MapReduce Total cumulative CPU time: 1 seconds 320 msec
Ended Job = job_1476764326039_0014
Stage-3 is selected by condition resolver.
Stage-2 is filtered out by condition resolver.
Stage-4 is filtered out by condition resolver.
Moving data to: hdfs://localhost:9000/tmp/hive-administrator/hive_2016-10-18_02-16-17_425_5894975364
0454830/-ext-10000
Moving data to: /home/administrator/Desktop/hive_data/index test result

```

Browsing HDFS - Mozilla Firefox

log file - ukcp.lend... LanguageManual... Hive - View and Ind... Hadoop Tutorial: ... Browsing HDFS

localhost:50070/explorer.html#/user/hive/warehouse/lendi\_db.db/lendi\_db\_log\_data\_i

Hadoop Overview Datanodes Snapshot Startup Progress Utilities

### Browse Directory

/user/hive/warehouse/lendi\_db.db/lendi\_db\_log\_data\_index\_ip\_\_ Go!

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	administrator	supergroup	106.78 KB	Tue 18 Oct 2016 02:09:07 AM EDT	1	128 MB	000000_0

## RESULT:

Thus Installation of Hive along with Database creation were studied and implemented.

Ex No.	7	Installation of HBase, Installing thrift along with Practice examples
Date		

**AIM:**

To study the installation of HBase and Thrift.

**DESCRIPTION :****Installing HBase**

We can install HBase in any of the three modes: Standalone mode, Pseudo Distributed mode, and Fully Distributed mode.

**Installing HBase in Standalone Mode**

Download the latest stable version of HBase form <http://www.interior-dsgn.com/apache/hbase/stable/> using “wget” command, and extract it using the tar “zxvf” command. See the following command.

```
$cd usr/local/
```

```
$wget http://www.interior-dsgn.com/apache/hbase/stable/hbase-0.98.8-hadoop2-bin.tar.gz
```

```
$tar -zxvf hbase-0.98.8-hadoop2-bin.tar.gz
```

Shift to super user mode and move the HBase folder to /usr/local as shown below.

```
$su
```

```
$password: enter your password heremv
```

```
hbase-0.99.1/* Hbase/
```

**Configuring HBase in Standalone Mode**

Before proceeding with HBase, you have to edit the following files and configure HBase.

```
hbase-env.sh
```

Set the java Home for HBase and open **hbase-env.sh** file from the conf folder. Edit JAVA\_HOME environment variable and change the existing path to your current JAVA\_HOME variable as shown below.

```
cd /usr/local/Hbase/conf
gedit hbase-env.sh
```

This will open the env.sh file of HBase. Now replace the existing **JAVA\_HOME** value with your current value as shown below.

```
export JAVA_HOME=/usr/lib/jvm/java-1.7.0
hbase-site.xml
```

```
#cd /usr/local/HBase/#cd
# gedit hbase-site.xml
```

Inside the **hbase-site.xml** file, you will find the <configuration> and </configuration> tags. Within them, set the HBase directory under the property key with the name “hbase.rootdir” as shown below.

```
<configuration>
//Here you have to set the path where you want HBase to store its files.
<property>
  <name>hbase.rootdir</name>
  <value>file:/home/hadoop/HBase/HFiles</value>
</property>

//Here you have to set the path where you want HBase to store its built in zookeeper files.
<property>
  <name>hbase.zookeeper.property.dataDir</name>
  <value>/home/hadoop/zookeeper</value>
</property>
</configuration>
```

With this, the HBase installation and configuration part is successfully complete. We can start HBase by using **start-hbase.sh** script provided in the bin folder of HBase. For that, open HBase Home Folder and run HBase start script as shown below.

```
$cd /usr/local/HBase/bin
$./start-hbase.sh
```

If everything goes well, when you try to run HBase start script, it will prompt you a message saying that HBase has started.

starting master, logging to /usr/local/HBase/bin/../logs/hbase-tpmaster-localhost.localdomain.out

Installing HBase in Pseudo-Distributed Mode

Let us now check how HBase is installed in pseudo-distributed mode.

### *Configuring HBase*

Before proceeding with HBase, configure Hadoop and HDFS on your local system or on a remote system and make sure they are running. Stop HBase if it is running.

#### **hbase-site.xml**

```
<property>
  <name>hbase.cluster.distributed</name>
  <value>true</value>
</property>
```

Edit hbase-site.xml file to add the following properties.

It will mention in which mode HBase should be run. In the same file from the local file system, change the hbase.rootdir, your HDFS instance address, using the hdfs:/// URI syntax. We are running HDFS on the localhost at port 8030.

#### **Starting HBase**

After configuration is over, browse to HBase home folder and start HBase using the following command.

```
$cd /usr/local/HBase
```

```
$bin/start-hbase.sh
```

**Note:** Before starting HBase, make sure Hadoop is running.

Checking the HBase Directory in HDFS

HBase creates its directory in HDFS. To see the created directory, browse to Hadoop bin and type the following command.

```
$ ./bin/hadoop fs -ls /hbase
```

If everything goes well, it will give you the following output.

Found 7 items

```
drwxr-xr-x - hbase users 0 2014-06-25 18:58 /hbase/.tmp
drwxr-xr-x - hbase users 0 2014-06-25 21:49 /hbase/WALs
drwxr-xr-x - hbase users 0 2014-06-25 18:48 /hbase/corrupt
drwxr-xr-x - hbase users 0 2014-06-25 18:58 /hbase/data
-rw-r--r-- 3 hbase users 42 2014-06-25 18:41 /hbase/hbase.id
-rw-r--r-- 3 hbase users 7 2014-06-25 18:41 /hbase/hbase.version
drwxr-xr-x - hbase users 0 2014-06-25 21:49 /hbase/oldWALs
```

#### Starting and Stopping a Master

Using the “local-master-backup.sh” you can start up to 10 servers. Open the home folder of HBase, master and execute the following command to start it.

```
$ ./bin/local-master-backup.sh 2 4
```

To kill a backup master, you need its process id, which will be stored in a file named “**/tmp/hbase-USER-X-master.pid.**” you can kill the backup master using the following command.

```
$ cat /tmp/hbase-user-1-master.pid |xargs kill -9
```

#### Starting and Stopping RegionServers

You can run multiple region servers from a single system using the following command.

```
$ ./bin/local-regionervers.sh start 2 3
```

To stop a region server, use the following command.

```
$ ./bin/local-regionervers.sh stop 3
```

### Starting HBaseShell

After Installing HBase successfully, you can start HBase Shell. Below given are the sequence of steps that are to be followed to start the HBase shell. Open the terminal, and login as super user.

Start Hadoop File System

Browse through Hadoop home sbin folder and start Hadoop file system as shown below.

```
$cd $HADOOP_HOME/sbin
```

```
$start-all.sh
```

Browse through the HBase root directory bin folder and start HBase.

```
$cd /usr/local/HBase
```

```
$/bin/start-hbase.sh
```

Start HBase Master Server

This will be the same directory. Start it as shown below.

```
$/bin/local-master-backup.sh start 2 (number signifies specific server.)
```

Start Region

Start the region server as shown below.

```
$/bin/./local-regionserver.sh start 3
```

Start HBase Shell

You can start HBase shell using the following command.

```
$cd bin
```

```
$/hbase shell
```

This will give you the HBase Shell Prompt as shown below.

```
2014-12-09 14:24:27,526 INFO [main] Configuration.deprecation:
```

```
hadoop.native.lib is deprecated. Instead, use io.native.lib.available HBase
```

Shell; enter 'help<RETURN>' for list of supported commands.Type

"exit<RETURN>" to leave the HBase Shell

Version 0.98.8-hadoop2, r6cfc8d064754251365e070a10a82eb169956d5fe, Fri

Nov 14 18:26:29 PST 2014

hbase(main):001:0> HBase

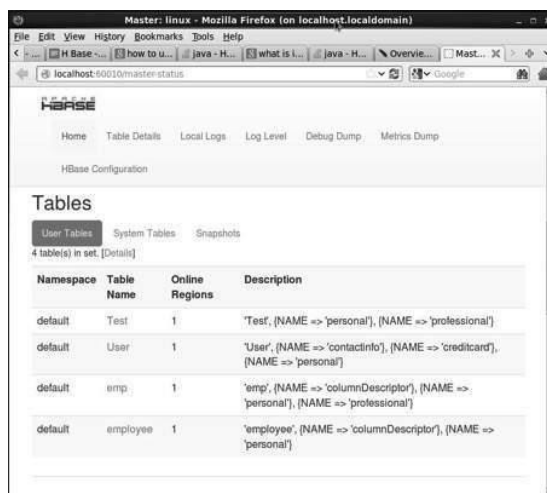
Web Interface

To access the web interface of HBase, type the following url in the browser.

<http://localhost:60010>

This interface lists your currently running Region servers, backup masters and HBase tables.HBase

Region servers and Backup Masters



The screenshot shows the HBase web interface in a Mozilla Firefox browser window. The page title is "Master: linux - Mozilla Firefox (on localhost.localdomain)". The address bar shows "localhost:60010/master/status". The page has a navigation bar with links: Home, Table Details, Local Logs, Log Level, Debug Dump, and Metrics Dump. Below the navigation bar is the "HBase Configuration" section. The main content area is titled "Tables" and has tabs for "User Tables", "System Tables", and "Snapshots". It indicates "4 table(s) in set. [Details]". Below this is a table with the following data:

Namespace	Table Name	Online Regions	Description
default	Test	1	'Test', (NAME => 'personal'), (NAME => 'professional')
default	User	1	'User', (NAME => 'contactinfo'), (NAME => 'creditcard'), (NAME => 'personal')
default	emp	1	'emp', (NAME => 'columnDescriptor'), (NAME => 'personal'), (NAME => 'professional')
default	employee	1	'employee', (NAME => 'columnDescriptor'), (NAME => 'personal')

We can also communicate with HBase using Java libraries, but before accessing HBase using JavaAPI you need to set classpath for those libraries.

Setting the Classpath

Before proceeding with programming, set the classpath to HBase libraries in **.bashrc** file.Open **.bashrc** in any of the editors as shown below.

```
$ gedit ~/.bashrc
```

Set classpath for HBase libraries (lib folder in HBase) in it as shown below.Export

```
CLASSPATH = $CLASSPATH://home/hadoop/hbase/lib/*
```



This is to prevent the “class not found” exception while accessing the HBase using java API.

### **Thrift installation**

Detailed information on how to install Thrift can be found here: <http://thrift.apache.org/docs/install/> On Ubuntu Linux for example you just need to first install the dependencies and then you are ready to install Thrift. Install the languages with which you plan to use thrift. To use with Java for example, install a Java JDK you prefer. In this demo I am using Oracle JDK 7 for Ubuntu, but you shouldn't have problem using the one you like.

To use with Java you will also need to install Apache Ant

**sudo apt-get install ant**

Installing required tools and libraries:

```
sudo apt-get install libboost-dev libboost-test-dev libboost-program-options-dev libboost-filesystem-dev  
libboost-thread-dev libevent-dev automake libtool flex bison pkg-config g++ libssl-dev
```

You can check for specific requirements for each language you wish to use here: <http://thrift.apache.org/docs/install/>

Download Thrift: <http://thrift.apache.org/download>

Copy the downloaded file into the desired directory and untar the file

**tar -xvf thrift-0.9.3.tar.gz**

For detailed instructions on how to build Apache Thrift on your specific system you can read here:

<http://thrift.apache.org/docs/BuildingFromSource>

For an Ubuntu linux distribution you just need to go to the thrift directory and type:

```
./bootstrap.sh
```

```
./configure
```

At the end of the output you should be able to see a list of all the libraries that are currently built in your system and ready to use with your desired programming languages. If a component is missing you should download the missing language and repeat the above step.

thrift 0.9.3

Building C++ Library ..... yes  
Building C (GLib) Library .....no  
Building Java Library ..... yes  
Building C# Library.....no  
Building Python Library ..... yes  
Building Ruby Library.....no  
Building Haskell Library ..... no  
Building Perl Library .....no  
Building PHP Library ..... no

Building D Library .....no

C++ Library:

Build TZlibTransport.....yes  
Build TNonblockingServer .. : yes  
Build QTcpServer (Qt).....no

Java Library:

Using javac ..... javac  
Using java ..... java  
Using ant ..... /usr/bin/ant

Python Library:

Using Python ...../usr/bin/python

Here <http://thrift.apache.org/docs/install/debian/> you can find all the packages you might need to support your desired language in case some of them are missing.

On the same directory run make to build Thrift

sudo make

(Optional) Run the test suite if you want

sudo make check

And finally you are ready to install Thrift by running

sudo make install

**Verify installation**

Now your Thrift installation is completed! To verify that you have successfully installed Thrift justtype **thrift –version** and you should be able to see something like the following:Thrift

version 0.9.0

Additionally you can go to the tutorial directory and follow the instructions located on the README files on each of the targeted languages directories. For example for JAVA you should be able to verify the following:

```
thrift/tutorial/java$ file ../../lib/java/build/libthrift-${version}-${release}.jar
```

```
../../lib/java/libthrift.jar: Zip archive data, at least v1.0 to extract
```

```
thrift/tutorial/java$ file ../../compiler/cpp/thrift
```

```
../../compiler/cpp/thrift: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked (uses shared libs), for GNU/Linux 2.6.15, not stripped
```

```
thrift/tutorial/java$ ls ../../lib/java/build/lib/
```

```
commons-lang-2.5.jar junit-4.4.jar servlet-api-2.5.jar slf4j-api-1.5.8.jar slf4j-simple-1.5.8.jar#
```

```
https://thrift-tutorial.readthedocs.io/en/latest/installation.htm
```

**RESULT: .**

Thus Hbase and Thrift installation were implemented.

<b>Ex No.</b>	<b>8</b>	<b>Practice importing and exporting data from various data bases</b>
<b>Date</b>		

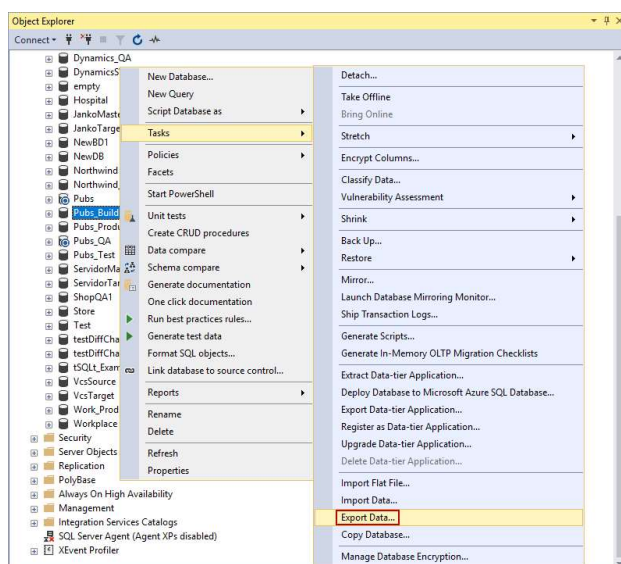
**AIM:**

To implement import and export data from various databases.

**DESCRIPTION :**

The SQL Server **Import and export wizard** provides a graphical user interface onto a SQL Server Integration Services (SSIS) package. Once created the package can be automated, to run on a schedule. It can be further configured and modified by using SQL Server Data Tools (SSDT)

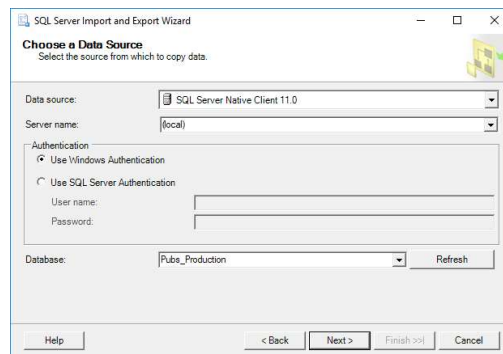
To begin, open the **Import and export wizard**, right-click a database and select the **Tasks** sub-menu  
-> **Export data** command:



1. Connect to a source database via the **Choose a data source** step.

Permissions: You will need the following permissions to for the source data source/instance

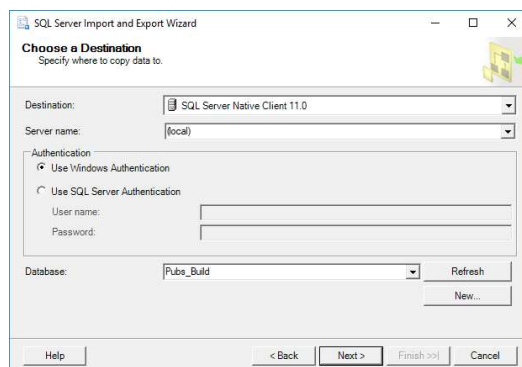
- a. read data from the database or file
- b. **Insert** permission on the msdb database to save the SSIS package

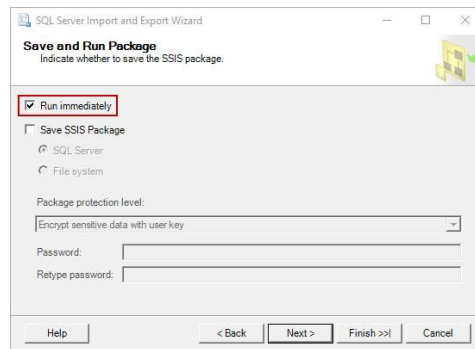


2. Connect to a destination SQL Server database in the **Choose a destination** step.

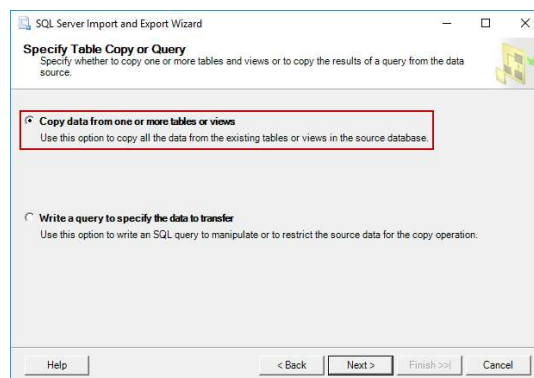
Permissions: The following permissions are required for the destination instance:

- a. write data to the database or file
- b. permissions to create a database
- c. if necessary, permission to create table

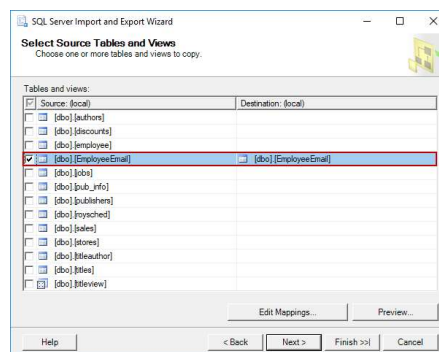


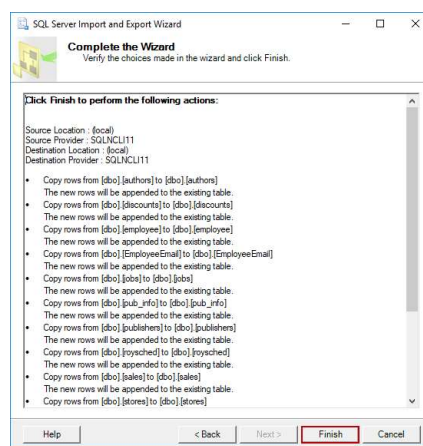
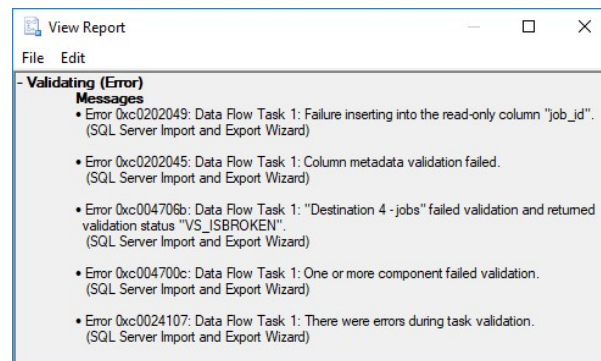


3. Choose the **Copy data from one or more tables or views** option, In the **Specify table copy or query** step:



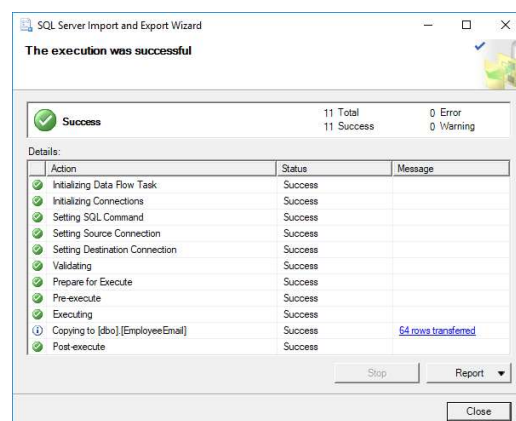
4. In the **Select source tables and views** step, choose the table(s) for which you want to export data from:
5. In the **Save and run package** step, choose the **Run immediately** option
6. in the **Complete the wizard** step, Click **Finish**





7. Once completed, the following dialog will be displayed:

Troubleshooting: The **SQL Server import and export wizard** will not distinguish identity columns from any other column type. This will lead to errors when inserting data into such columns



Troubleshooting: The **SQL Server import and export wizard** also doesn't process tables based on dependency order. An example might be loading a table, with a foreign key (the child), before the referencing table the parent, causing a foreign key constraint failure

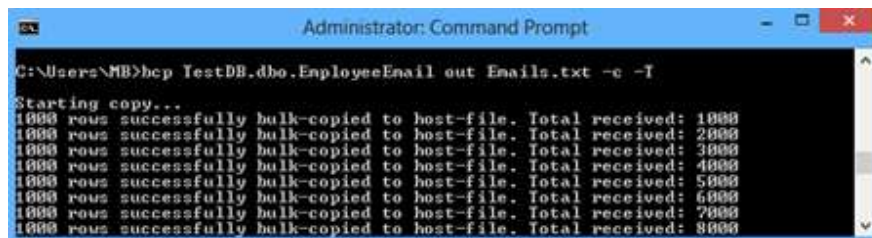
Bulk copy

### BCP utility

The BCP utility is a console application, managed via the command line, that can build import/export data from a database to a file and visa versa

For example, to export all *TeamMemberIDs* and *Emails* to the *TeamMembers.txt* file from the *TeamMemberEmail* table in the *QA* database, run the following bcp command:

**bcp QA.dbo.TeamMemberEmail out TeamMembers.txt -c -T**



```
Administrator: Command Prompt
C:\Users\MB>bcp TestDB.dbo.EmployeeEmail out Emails.txt -c -T
Starting copy...
1000 rows successfully bulk-copied to host-file. Total received: 1000
1000 rows successfully bulk-copied to host-file. Total received: 2000
1000 rows successfully bulk-copied to host-file. Total received: 3000
1000 rows successfully bulk-copied to host-file. Total received: 4000
1000 rows successfully bulk-copied to host-file. Total received: 5000
1000 rows successfully bulk-copied to host-file. Total received: 6000
1000 rows successfully bulk-copied to host-file. Total received: 7000
1000 rows successfully bulk-copied to host-file. Total received: 8000
```

The **-c** switch specifies that the utility is being used with character data and that the **-T** switch states that this process will use a trusted connection, the Windows login credentials of the user that is currently logged. If the **-T** option is not specified a username and password must be specified with the **-U** and **-P** options.

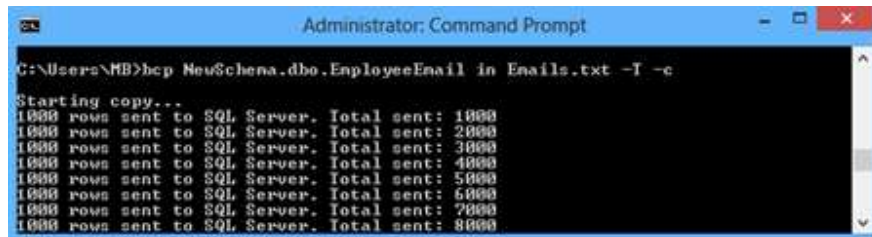
As would be expected, the destination table must exist prior to the import and the table must have the expected number and data types of columns to match the imported data.

To insert data from the *TeamMembers.txt* file into the *NewSchema.dbo.TeamMemberEmail* table use



the following bcp command:

**bcp NewSchema.dbo.TeamMemberEmail in TeamMembers.txt -T -c**



```
Administrator: Command Prompt
G:\Users\MB>bcp NewSchema.dbo.EmployeeEmail in Emails.txt -T -c
Starting copy...
1000 rows sent to SQL Server. Total sent: 1000
1000 rows sent to SQL Server. Total sent: 2000
1000 rows sent to SQL Server. Total sent: 3000
1000 rows sent to SQL Server. Total sent: 4000
1000 rows sent to SQL Server. Total sent: 5000
1000 rows sent to SQL Server. Total sent: 6000
1000 rows sent to SQL Server. Total sent: 7000
1000 rows sent to SQL Server. Total sent: 8000
```

Permissions: **Select** permissions are required on the source table

### Bulk insert statement

Another option for importing/exporting data between files and tables is the **Bulk insert** statement. The same restrictions and requirements that apply to BCP apply to Bulk insert as well including the requirement for a table to exist that matches the imported data

### Openrowset(Bulk) function

**Openrowset(Bulk)** is a T-SQL function that connects via an OLE DB data source to read data. It can access remoted data sources from a remote connection vs a linked server

INSERT INTO AllEmails(Email)

BULK 'C:\TeamMembers.txt ',

SINGLE\_BLOB) AS x;

The **Openrowset(Bulk)** function provides an alternative to accessing objects from a linked server and it is suitable for one-off entry of data from a remote source.

## SELECT INTO

The **Into** clause used, in combination with the **Select** statement, enables creating a new table based on the result set of the **Select** statement. For example, to copy the *TeamMemberEmail* table, on the same instance, in the default schema of the *QA* database, run the following query:

```
SELECT * INTO QA.dbo.TeamMemberEmail  
  
FROM Neptune.HumanResources.TeamMemberEmail;
```

**Select into** cannot be used to create a new table on a remote SQL Server instance, but a remote source can be included in the **Select** statement, if there is a link to the remote instance.

Any constraints, indexes, and triggers will not be transferred to the new table. Columns in the newly created table will not inherit the **Identity property** from the query output if the **Select** statement contains an aggregate function, a **Join** clause, or a **Group by** clause, and if an identity column is used in an expression, is used more than once, or is from a remote data source

## RESULT:

Thus import and export of data's from various databases are implemented.