2015

Hadoop Bigdata

Shalaj Shukla

Table of Contents

[HDFS System commands 1](#_Toc413244713)

[Hadoop Data Flow 6](#_Toc413244714)

[Partitioner 15](#_Toc413244715)

[HBase 21](#_Toc413244716)

[HBase Data Model Overview 21](#_Toc413244717)

[Hbase prompt 24](#_Toc413244718)

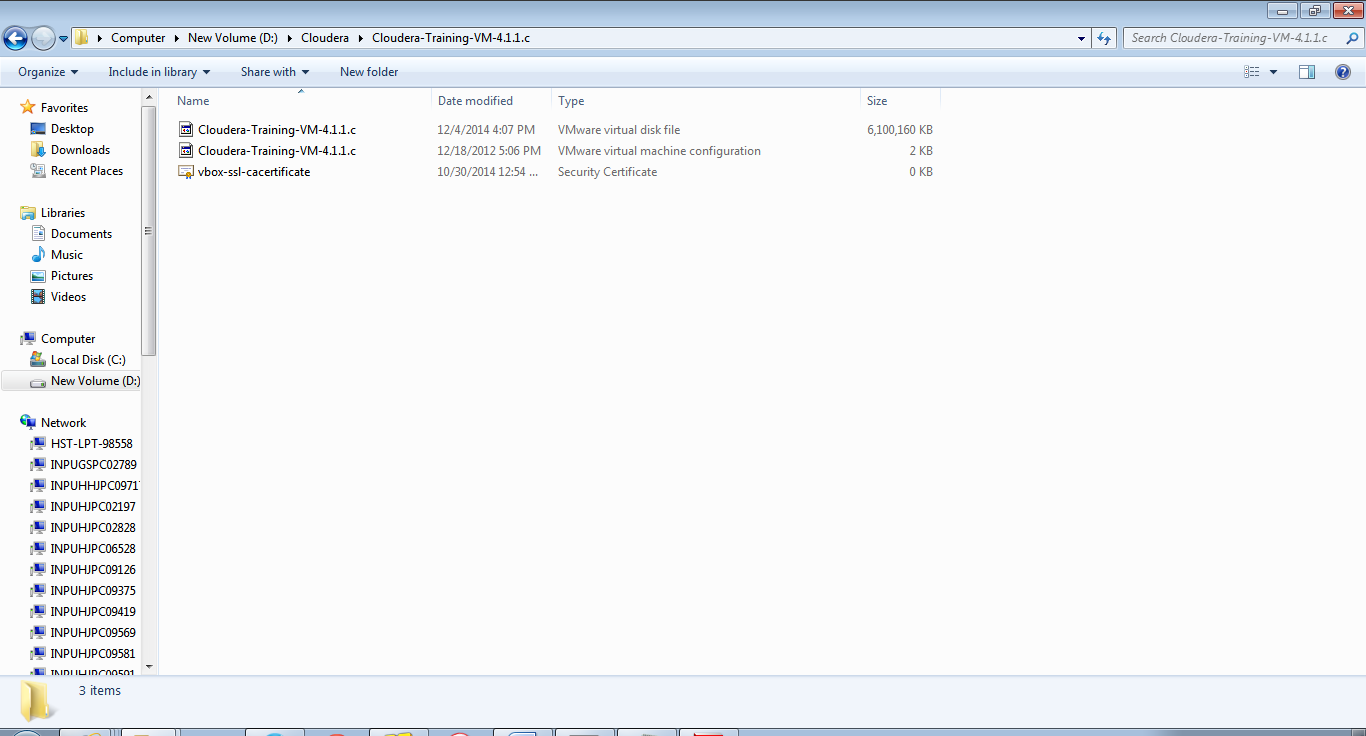
[List all the tables 25](#_Toc413244719)

[Create and Describe Tables 26](#_Toc413244720)

[Insert Data into table 27](#_Toc413244721)

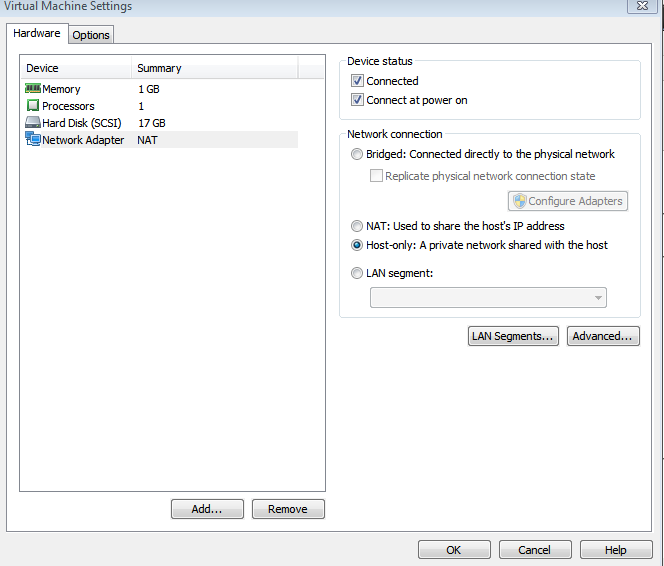
[Retrieving data from table 29](#_Toc413244722)

# HDFS System commands



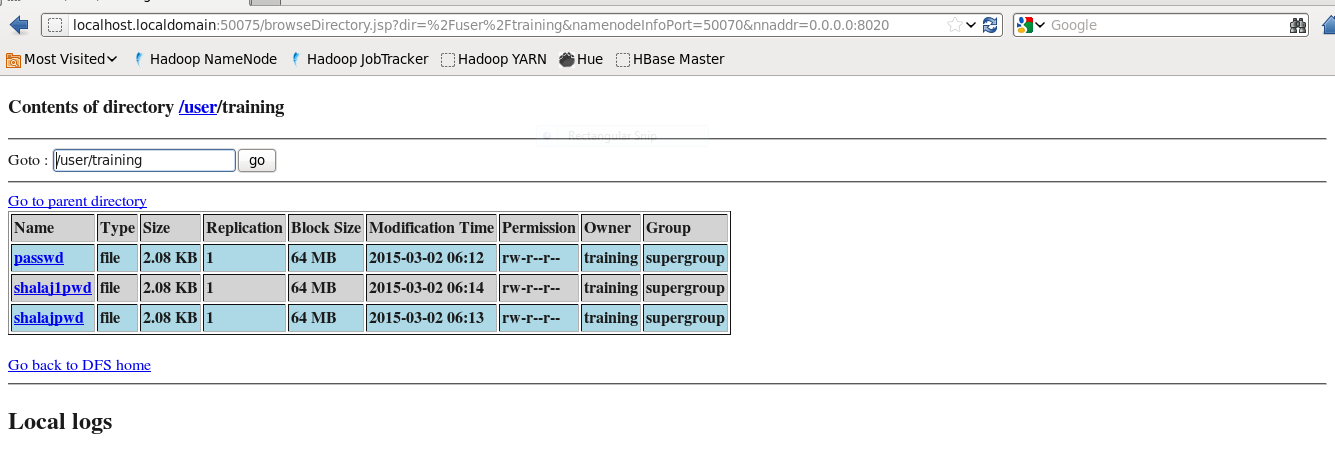
Double click on Cloudera-Training-VM-4.1.1.c , this will create virtual machine

Player>>Manage>>Virtual Machine Settings

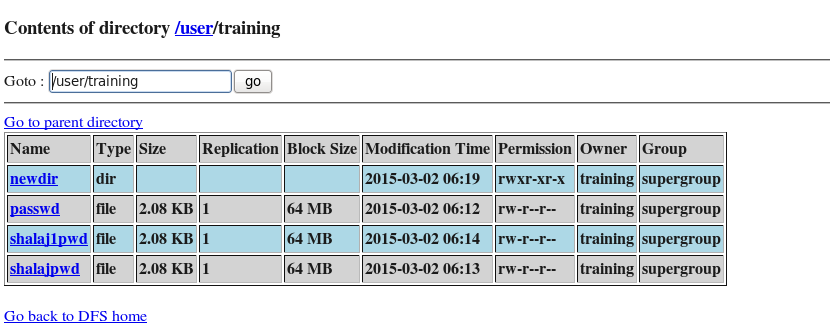


|  |
| --- |
| [training@localhost ~]$ ***hdfs dfs –ls***  Or  [training@localhost ~]$ hadoop fs –ls  [training@localhost ~]$ ***hadoop fs -copyFromLocal /etc/passed***  File will copy Root directory of hadoop  [training@localhost ~]$ ***hdfs dfs -copyFromLocal /etc/passwd shalajpwd***  [training@localhost ~]$ ***hdfs dfs -put /etc/passwd shalaj1pwd*** |

Hadoop NameNode url



|  |
| --- |
| To create new directory  [training@localhost ~]$ ***hdfs dfs -mkdir newdir*** |



|  |
| --- |
| Copy file in newdir  [training@localhost ~]$ ***hdfs dfs -put /etc/passwd newdir/shalaj1pwd*** |

|  |
| --- |
| [training@localhost ~]$ ***hdfs dfs -ls***  Found 4 items  drwxr-xr-x - training supergroup 0 2015-03-02 06:22 newdir  -rw-r--r-- 1 training supergroup 2132 2015-03-02 06:12 passwd  -rw-r--r-- 1 training supergroup 2132 2015-03-02 06:14 shalaj1pwd  -rw-r--r-- 1 training supergroup 2132 2015-03-02 06:13 shalajpwd |

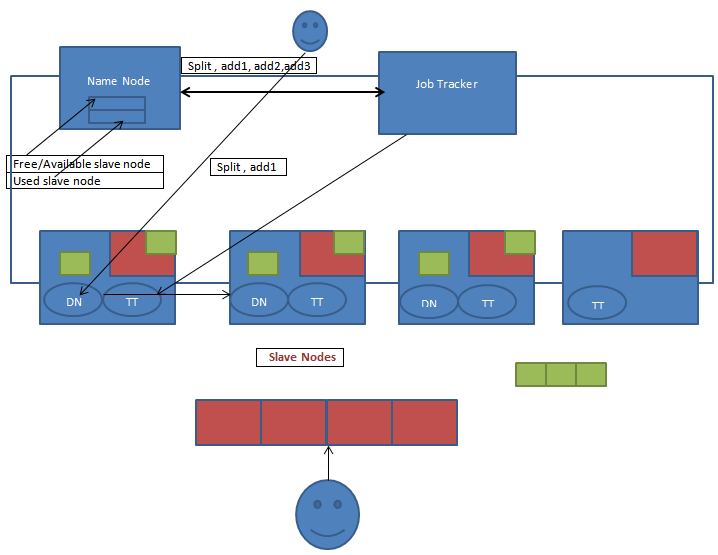
|  |
| --- |
| To see the contents  [training@localhost ~]$ ***hadoop fs -cat passwd*** |

|  |
| --- |
| [training@localhost ~]$ ***hdfs dfs -rm -r newdir***  Deleted newdir |

|  |
| --- |
| [training@localhost ~]$ ***hdfs dfs -copyToLocal shalajpwd***  [training@localhost ~]$ ls -ltr  total 60  drwxrwxr-x 13 training training 4096 Nov 9 2012 src  drwxrwxr-x 5 training training 4096 Dec 18 2012 training\_materials  drwxrwxr-x 2 training training 4096 Dec 18 2012 scripts  drwxr-xr-x 2 training training 4096 Dec 18 2012 lib  drwxr-xr-x 2 training training 4096 Mar 2 00:10 Videos  drwxr-xr-x 2 training training 4096 Mar 2 00:10 Templates  drwxr-xr-x 2 training training 4096 Mar 2 00:10 Public  drwxr-xr-x 2 training training 4096 Mar 2 00:10 Pictures  drwxr-xr-x 2 training training 4096 Mar 2 00:10 Music  drwxr-xr-x 2 training training 4096 Mar 2 00:10 Downloads  drwxr-xr-x 2 training training 4096 Mar 2 00:10 Documents  drwxr-xr-x 9 training training 4096 Mar 2 04:22 eclipse  -rwxr-xr-x 1 training training 2132 Mar 2 06:28 **shalajpwd**  drwxr-xr-x 2 training training 4096 Mar 2 2015 Desktop  drwxrwxr-x 15 training training 4096 Mar 2 2015 workspace |

|  |
| --- |
| [training@localhost ~]$ hdfs dfs -get shalaj1pwd  [training@localhost ~]$ ls -ltr  total 64  drwxrwxr-x 13 training training 4096 Nov 9 2012 src  drwxrwxr-x 5 training training 4096 Dec 18 2012 training\_materials  drwxrwxr-x 2 training training 4096 Dec 18 2012 scripts  drwxr-xr-x 2 training training 4096 Dec 18 2012 lib  drwxr-xr-x 2 training training 4096 Mar 2 00:10 Videos  drwxr-xr-x 2 training training 4096 Mar 2 00:10 Templates  drwxr-xr-x 2 training training 4096 Mar 2 00:10 Public  drwxr-xr-x 2 training training 4096 Mar 2 00:10 Pictures  drwxr-xr-x 2 training training 4096 Mar 2 00:10 Music  drwxr-xr-x 2 training training 4096 Mar 2 00:10 Downloads  drwxr-xr-x 2 training training 4096 Mar 2 00:10 Documents  drwxr-xr-x 9 training training 4096 Mar 2 04:22 eclipse  -rwxr-xr-x 1 training training 2132 Mar 2 06:28 shalajpwd  -rwxr-xr-x 1 training training 2132 Mar 2 06:29 **shalaj1pwd**  drwxr-xr-x 2 training training 4096 Mar 2 2015 Desktop  drwxrwxr-x 15 training training 4096 Mar 2 2015 workspace |

# Hadoop Data Flow



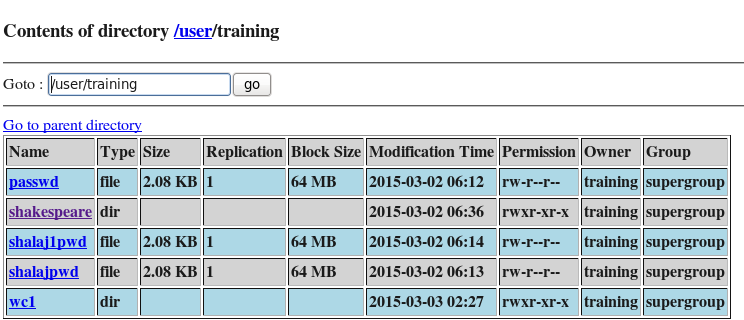
|  |  |
| --- | --- |
| DN | Data Node |
| TT | Task Tracker |
| color | HDFS Area |
| color | Split –sorted/shuffled files |

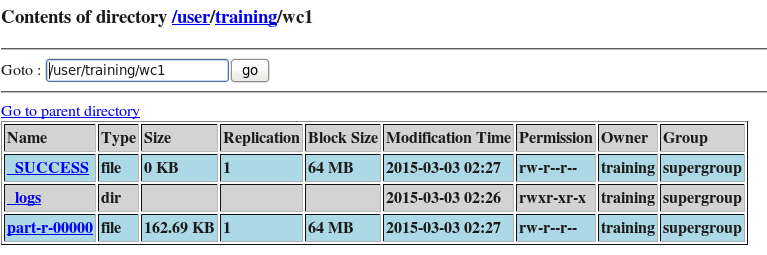
|  |
| --- |
| [training@localhost training\_materials]$ cd developer/  [training@localhost developer]$ cd data/  [training@localhost data]$  [training@localhost data]$ ***tar zxvf shakespeare.tar.gz***  shakespeare/  shakespeare/comedies  shakespeare/glossary  shakespeare/histories  shakespeare/poems  shakespeare/tragedies  [training@localhost data]$ ***hdfs dfs -put Shakespeare***  To display content of Shakespeare directory under hdfs, we can not go to the directory using CD  [training@localhost data]$ ***hdfs dfs -ls shakespeare***  Found 5 items  -rw-r--r-- 1 training supergroup 1784616 2015-03-02 06:36 shakespeare/comedies  -rw-r--r-- 1 training supergroup 58976 2015-03-02 06:36 shakespeare/glossary  -rw-r--r-- 1 training supergroup 1479035 2015-03-02 06:36 shakespeare/histories  -rw-r--r-- 1 training supergroup 268140 2015-03-02 06:36 shakespeare/poems  -rw-r--r-- 1 training supergroup 1752440 2015-03-02 06:36 shakespeare/tragedies |

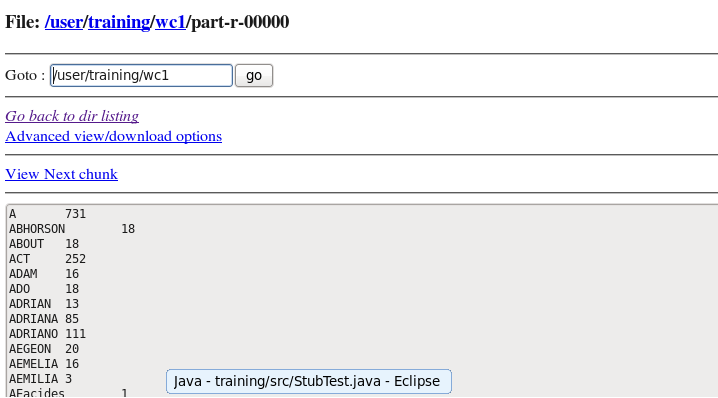
|  |
| --- |
| [training@localhost wordcount]$ pwd  /home/training/training\_materials/developer/exercises/wordcount  [training@localhost wordcount]$ ***javac -classpath `hadoop classpath` \*.java***  [training@localhost wordcount]$  [training@localhost wordcount]$ ls -ltr  total 36  -rw-rw-r-- 1 training training 1714 Dec 18 2012 WordMapper.java  -rw-rw-r-- 1 training training 3186 Dec 18 2012 WordCount.java  -rw-rw-r-- 1 training training 1471 Dec 18 2012 SumReducer.java  drwxrwxr-x 2 training training 4096 Dec 18 2012 stubs\_with\_hints  drwxrwxr-x 2 training training 4096 Dec 18 2012 sample\_solution\_oldapi  drwxrwxr-x 2 training training 4096 Dec 18 2012 sample\_solution  -rw-rw-r-- 1 training training 1586 Mar 3 02:20 SumReducer.class  -rw-rw-r-- 1 training training 1525 Mar 3 02:20 WordCount.class  -rw-rw-r-- 1 training training 1699 Mar 3 02:20 WordMapper.class  [training@localhost wordcount]$ ***jar cvf WordCount.jar \*.class***  added manifest  adding: SumReducer.class(in = 1586) (out= 660)(deflated 58%)  adding: WordCount.class(in = 1525) (out= 833)(deflated 45%)  adding: WordMapper.class(in = 1699) (out= 707)(deflated 58%)  [training@localhost wordcount]$ ls -l  total 40  drwxrwxr-x 2 training training 4096 Dec 18 2012 sample\_solution  drwxrwxr-x 2 training training 4096 Dec 18 2012 sample\_solution\_oldapi  drwxrwxr-x 2 training training 4096 Dec 18 2012 stubs\_with\_hints  -rw-rw-r-- 1 training training 1586 Mar 3 02:20 SumReducer.class  -rw-rw-r-- 1 training training 1471 Dec 18 2012 SumReducer.java  -rw-rw-r-- 1 training training 1525 Mar 3 02:20 WordCount.class  -rw-rw-r-- 1 training training 2915 Mar 3 02:22 WordCount.jar  -rw-rw-r-- 1 training training 3186 Dec 18 2012 WordCount.java  -rw-rw-r-- 1 training training 1699 Mar 3 02:20 WordMapper.class  -rw-rw-r-- 1 training training 1714 Dec 18 2012 WordMapper.java |

Process one single file

|  |
| --- |
| [training@localhost wordcount]$ ***hadoop jar WordCount.jar WordCount shakespeare/comedies wc1***  Command  hadoop jar <jar file name> <driver class name> <input path from hadoop system> <output directory in hadoop system>  15/03/03 02:26:58 WARN mapred.JobClient: Use GenericOptionsParser for parsing the arguments. Applications should implement Tool for the same.  15/03/03 02:26:58 INFO input.FileInputFormat: Total input paths to process : 1  15/03/03 02:26:58 WARN snappy.LoadSnappy: Snappy native library is available  15/03/03 02:26:58 INFO snappy.LoadSnappy: Snappy native library loaded  15/03/03 02:26:58 INFO mapred.JobClient: Running job: job\_201503020010\_0001  15/03/03 02:26:59 INFO mapred.JobClient: map 0% reduce 0%  15/03/03 02:27:07 INFO mapred.JobClient: map 100% reduce 0%  15/03/03 02:27:15 INFO mapred.JobClient: map 100% reduce 100%  15/03/03 02:27:15 INFO mapred.JobClient: Job complete: job\_201503020010\_0001  15/03/03 02:27:15 INFO mapred.JobClient: Counters: 32  15/03/03 02:27:15 INFO mapred.JobClient: File System Counters  15/03/03 02:27:15 INFO mapred.JobClient: FILE: Number of bytes read=7271496  15/03/03 02:27:15 INFO mapred.JobClient: FILE: Number of bytes written=11268931  15/03/03 02:27:15 INFO mapred.JobClient: FILE: Number of read operations=0  15/03/03 02:27:15 INFO mapred.JobClient: FILE: Number of large read operations=0  15/03/03 02:27:15 INFO mapred.JobClient: FILE: Number of write operations=0  15/03/03 02:27:15 INFO mapred.JobClient: HDFS: Number of bytes read=1784735  15/03/03 02:27:15 INFO mapred.JobClient: HDFS: Number of bytes written=166599  15/03/03 02:27:15 INFO mapred.JobClient: HDFS: Number of read operations=2  15/03/03 02:27:15 INFO mapred.JobClient: HDFS: Number of large read operations=0  15/03/03 02:27:15 INFO mapred.JobClient: HDFS: Number of write operations=1  15/03/03 02:27:15 INFO mapred.JobClient: Job Counters  15/03/03 02:27:15 INFO mapred.JobClient: Launched map tasks=1  15/03/03 02:27:15 INFO mapred.JobClient: Launched reduce tasks=1  15/03/03 02:27:15 INFO mapred.JobClient: Data-local map tasks=1  15/03/03 02:27:15 INFO mapred.JobClient: Total time spent by all maps in occupied slots (ms)=7412  15/03/03 02:27:15 INFO mapred.JobClient: Total time spent by all reduces in occupied slots (ms)=7821  15/03/03 02:27:15 INFO mapred.JobClient: Total time spent by all maps waiting after reserving slots (ms)=0  15/03/03 02:27:15 INFO mapred.JobClient: Total time spent by all reduces waiting after reserving slots (ms)=0  15/03/03 02:27:15 INFO mapred.JobClient: Map-Reduce Framework  15/03/03 02:27:15 INFO mapred.JobClient: Map input records=59497  15/03/03 02:27:15 INFO mapred.JobClient: Map output records=328969  15/03/03 02:27:15 INFO mapred.JobClient: Map output bytes=2977801  15/03/03 02:27:15 INFO mapred.JobClient: Input split bytes=119  15/03/03 02:27:15 INFO mapred.JobClient: Combine input records=0  15/03/03 02:27:15 INFO mapred.JobClient: Combine output records=0  15/03/03 02:27:15 INFO mapred.JobClient: Reduce input groups=16769  15/03/03 02:27:15 INFO mapred.JobClient: Reduce shuffle bytes=3635745  15/03/03 02:27:15 INFO mapred.JobClient: Reduce input records=328969  15/03/03 02:27:15 INFO mapred.JobClient: Reduce output records=16769  15/03/03 02:27:15 INFO mapred.JobClient: Spilled Records=986907  15/03/03 02:27:15 INFO mapred.JobClient: CPU time spent (ms)=2270  15/03/03 02:27:15 INFO mapred.JobClient: Physical memory (bytes) snapshot=245239808  15/03/03 02:27:15 INFO mapred.JobClient: Virtual memory (bytes) snapshot=776220672  15/03/03 02:27:15 INFO mapred.JobClient: Total committed heap usage (bytes)=176492544 |

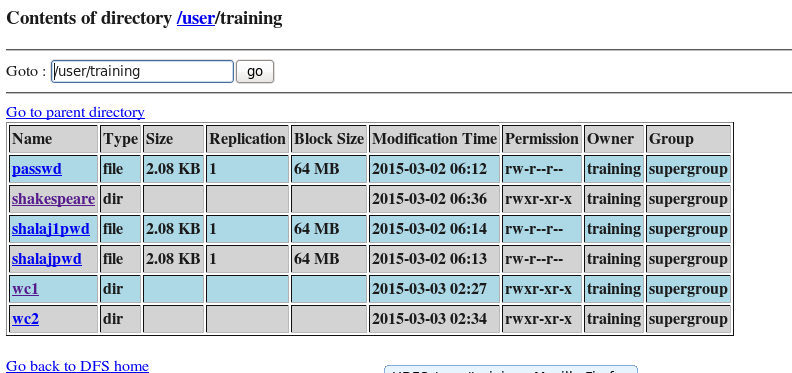


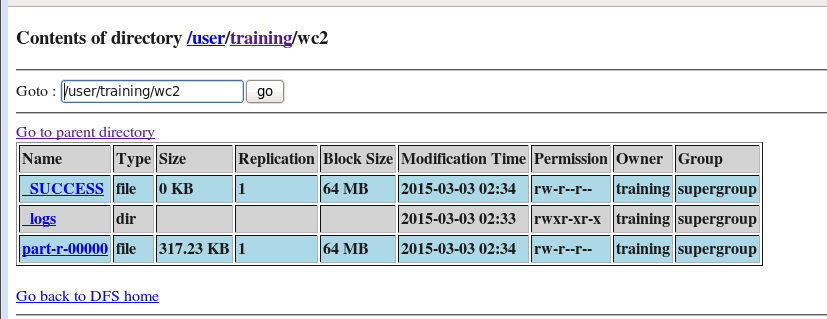


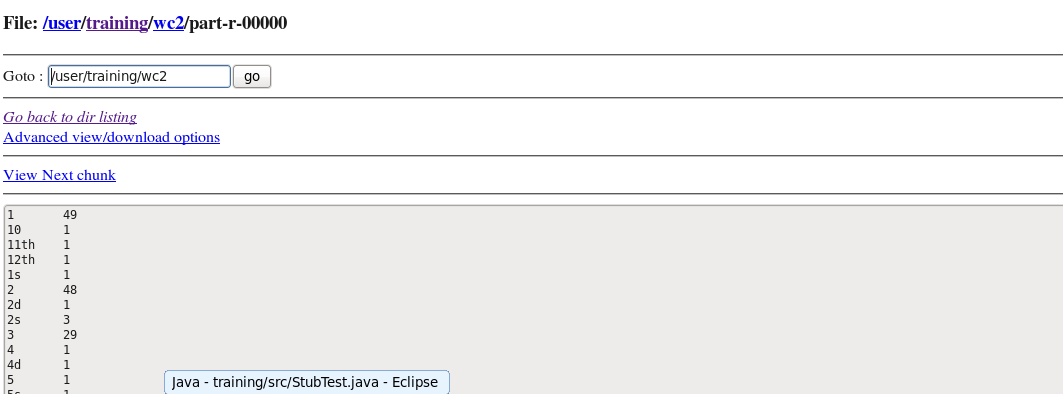


Process whole directory

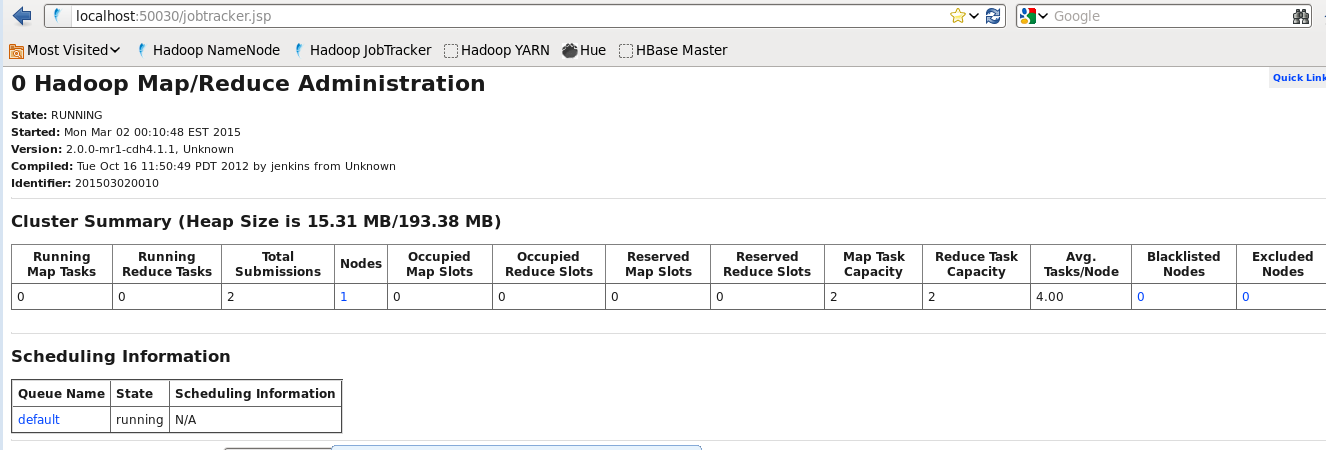
|  |
| --- |
| [training@localhost wordcount]$ ***hadoop jar WordCount.jar WordCount shakespeare wc2***  15/03/03 02:33:35 WARN mapred.JobClient: Use GenericOptionsParser for parsing the arguments. Applications should implement Tool for the same.  15/03/03 02:33:35 INFO input.FileInputFormat: Total input paths to process : 5  15/03/03 02:33:35 WARN snappy.LoadSnappy: Snappy native library is available  15/03/03 02:33:35 INFO snappy.LoadSnappy: Snappy native library loaded  15/03/03 02:33:35 INFO mapred.JobClient: Running job: job\_201503020010\_0002  15/03/03 02:33:36 INFO mapred.JobClient: map 0% reduce 0%  15/03/03 02:33:52 INFO mapred.JobClient: map 39% reduce 0%  15/03/03 02:33:54 INFO mapred.JobClient: map 40% reduce 0%  15/03/03 02:34:04 INFO mapred.JobClient: map 60% reduce 0%  15/03/03 02:34:06 INFO mapred.JobClient: map 80% reduce 0%  15/03/03 02:34:09 INFO mapred.JobClient: map 80% reduce 26%  15/03/03 02:34:12 INFO mapred.JobClient: map 100% reduce 26%  15/03/03 02:34:14 INFO mapred.JobClient: map 100% reduce 100%  15/03/03 02:34:15 INFO mapred.JobClient: Job complete: job\_201503020010\_0002  15/03/03 02:34:15 INFO mapred.JobClient: Counters: 32  15/03/03 02:34:15 INFO mapred.JobClient: File System Counters  15/03/03 02:34:15 INFO mapred.JobClient: FILE: Number of bytes read=20983046  15/03/03 02:34:15 INFO mapred.JobClient: FILE: Number of bytes written=32897522  15/03/03 02:34:15 INFO mapred.JobClient: FILE: Number of read operations=0  15/03/03 02:34:15 INFO mapred.JobClient: FILE: Number of large read operations=0  15/03/03 02:34:15 INFO mapred.JobClient: FILE: Number of write operations=0  15/03/03 02:34:15 INFO mapred.JobClient: HDFS: Number of bytes read=5343801  15/03/03 02:34:15 INFO mapred.JobClient: HDFS: Number of bytes written=324841  15/03/03 02:34:15 INFO mapred.JobClient: HDFS: Number of read operations=10  15/03/03 02:34:15 INFO mapred.JobClient: HDFS: Number of large read operations=0  15/03/03 02:34:15 INFO mapred.JobClient: HDFS: Number of write operations=1  15/03/03 02:34:15 INFO mapred.JobClient: Job Counters  15/03/03 02:34:15 INFO mapred.JobClient: Launched map tasks=5  15/03/03 02:34:15 INFO mapred.JobClient: Launched reduce tasks=1  15/03/03 02:34:15 INFO mapred.JobClient: Data-local map tasks=5  15/03/03 02:34:15 INFO mapred.JobClient: Total time spent by all maps in occupied slots (ms)=62241  15/03/03 02:34:15 INFO mapred.JobClient: Total time spent by all reduces in occupied slots (ms)=19277  15/03/03 02:34:15 INFO mapred.JobClient: Total time spent by all maps waiting after reserving slots (ms)=0  15/03/03 02:34:15 INFO mapred.JobClient: Total time spent by all reduces waiting after reserving slots (ms)=0  15/03/03 02:34:15 INFO mapred.JobClient: Map-Reduce Framework  15/03/03 02:34:15 INFO mapred.JobClient: Map input records=175558  15/03/03 02:34:15 INFO mapred.JobClient: Map output records=974078  15/03/03 02:34:15 INFO mapred.JobClient: Map output bytes=8880434  15/03/03 02:34:15 INFO mapred.JobClient: Input split bytes=594  15/03/03 02:34:15 INFO mapred.JobClient: Combine input records=0  15/03/03 02:34:15 INFO mapred.JobClient: Combine output records=0  15/03/03 02:34:15 INFO mapred.JobClient: Reduce input groups=31809  15/03/03 02:34:15 INFO mapred.JobClient: Reduce shuffle bytes=10828620  15/03/03 02:34:15 INFO mapred.JobClient: Reduce input records=974078  15/03/03 02:34:15 INFO mapred.JobClient: Reduce output records=31809  15/03/03 02:34:15 INFO mapred.JobClient: Spilled Records=2862397  15/03/03 02:34:15 INFO mapred.JobClient: CPU time spent (ms)=8110  15/03/03 02:34:15 INFO mapred.JobClient: Physical memory (bytes) snapshot=905449472  15/03/03 02:34:15 INFO mapred.JobClient: Virtual memory (bytes) snapshot=2327064576  15/03/03 02:34:15 INFO mapred.JobClient: Total committed heap usage (bytes)=834162688 |



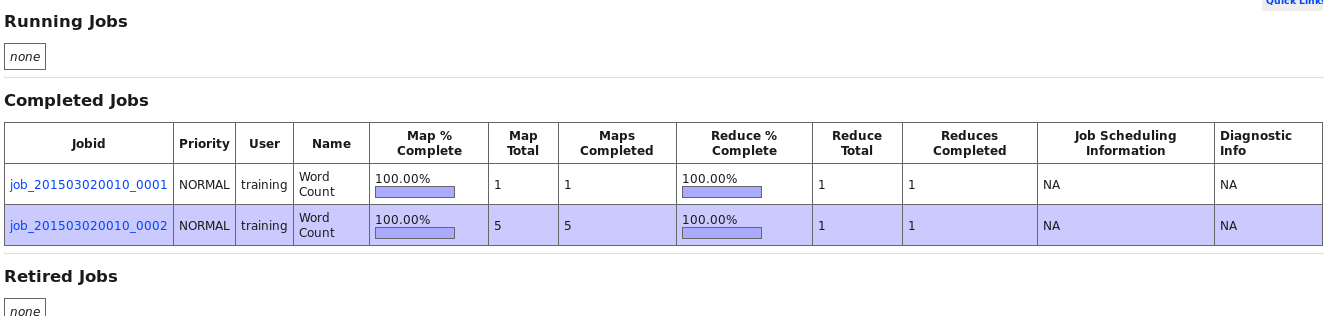


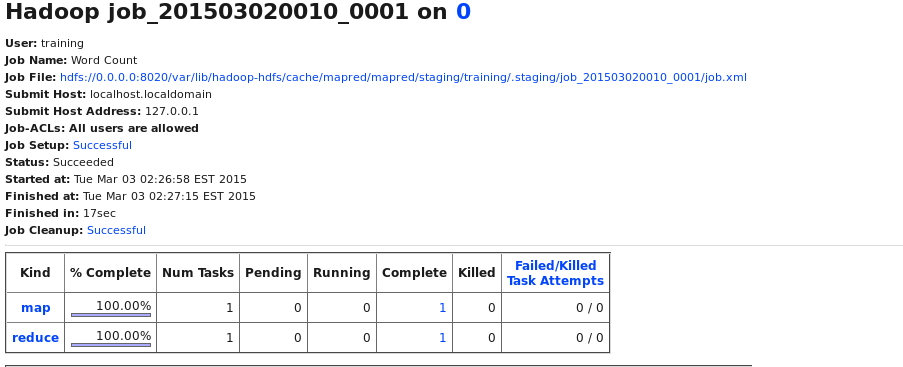


Job Tacker URL



Job information





# Partitioner

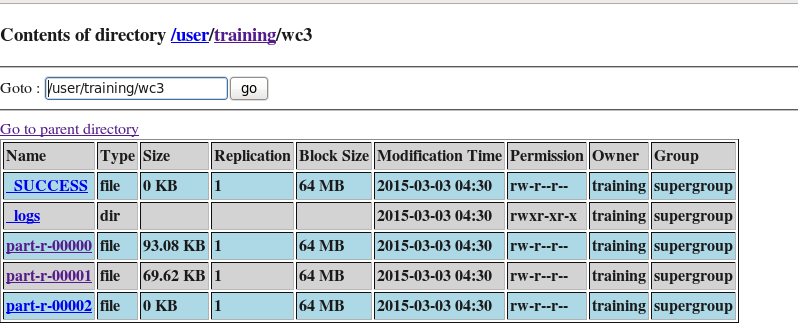
|  |
| --- |
| import java.io.IOException;  import org.apache.hadoop.io.IntWritable;  import org.apache.hadoop.io.LongWritable;  import org.apache.hadoop.io.Text;  import org.apache.hadoop.mapreduce.Mapper;  /\*  \* To define a map function for your MapReduce job, subclass  \* the Mapper class and override the map method.  \* The class definition requires four parameters:  \* The data type of the input key  \* The data type of the input value  \* The data type of the output key (which is the input key type  \* for the reducer)  \* The data type of the output value (which is the input value  \* type for the reducer)  \*/  public class WordMapper extends Mapper<LongWritable, Text, Text, IntWritable> {  /\*  \* The map method runs once for each line of text in the input file.  \* The method receives a key of type LongWritable, a value of type  \* Text, and a Context object.  \*/  @Override  public void map(LongWritable key, Text value, Context context)  throws IOException, InterruptedException {  /\*  \* Convert the line, which is received as a Text object,  \* to a String object.  \*/  String line = value.toString();  /\*  \* The line.split("\\W+") call uses regular expressions to split the  \* line up by non-word characters.  \*  \* If you are not familiar with the use of regular expressions in  \* Java code, search the web for "Java Regex Tutorial."  \*/  for (String word : line.split("\\W+")) {  if (word.length() > 0) {    /\*  \* Call the write method on the Context object to emit a key  \* and a value from the map method.  \*/  context.write(new Text(word), new IntWritable(1));  }  }  }  } |

|  |
| --- |
| import java.io.IOException;  import org.apache.hadoop.io.IntWritable;  import org.apache.hadoop.io.Text;  import org.apache.hadoop.mapreduce.Reducer;  /\*  \* To define a reduce function for your MapReduce job, subclass  \* the Reducer class and override the reduce method.  \* The class definition requires four parameters:  \* The data type of the input key (which is the output key type  \* from the mapper)  \* The data type of the input value (which is the output value  \* type from the mapper)  \* The data type of the output key  \* The data type of the output value  \*/  public class SumReducer extends Reducer<Text, IntWritable, Text, IntWritable> {  /\*  \* The reduce method runs once for each key received from  \* the shuffle and sort phase of the MapReduce framework.  \* The method receives a key of type Text, a set of values of type  \* IntWritable, and a Context object.  \*/  @Override  public void reduce(Text key, Iterable<IntWritable> values, Context context)  throws IOException, InterruptedException {  int wordCount = 0;    /\*  \* For each value in the set of values passed to us by the mapper:  \*/  for (IntWritable value : values) {    /\*  \* Add the value to the word count counter for this key.  \*/  wordCount += value.get();  }    /\*  \* Call the write method on the Context object to emit a key  \* and a value from the reduce method.  \*/  context.write(key, new IntWritable(wordCount));  }  } |

|  |
| --- |
| import java.util.HashMap;  import org.apache.hadoop.io.Text;  import org.apache.hadoop.conf.Configurable;  import org.apache.hadoop.conf.Configuration;  import org.apache.hadoop.mapreduce.Partitioner;  import org.apache.hadoop.io.IntWritable;  public class MyPartitioner<K2, V2> extends Partitioner<Text, IntWritable> implements  Configurable {  private Configuration configuration;  HashMap<String, Integer> months = new HashMap<String, Integer>();  /\*\*  \* Set up the months hash map in the setConf method.  \*/  @Override  public void setConf(Configuration configuration) {  /\*  \* Add the months to a HashMap.  \*/  /\*  \* TODO implement  \*/  }  /\*\*  \* Implement the getConf method for the Configurable interface.  \*/  @Override  public Configuration getConf() {  return configuration;  }  /\*\*  \* You must implement the getPartition method for a partitioner class.  \* This method receives the three-letter abbreviation for the month  \* as its value. (It is the output value from the mapper.)  \* It should return an integer representation of the month.  \* Note that January is represented as 0 rather than 1.  \*  \* For this partitioner to work, the job configuration must have been  \* set so that there are exactly 12 reducers.  \*/  public int getPartition(Text key, IntWritable value, int numReduceTasks) {  /\*  \* TODO implement  \* Change the return 0 statement below to return a string representation  \* of the month. Call months.get(value.toString()) and cast it to int.  \*/  String k = key.toString();  char ch = k.charAt(0);  int x = (int)ch;  if( (x>=65 && x <=77) || (x>=97 && x<=109)){  return 0;  } else {  return 1;  }  }  } |

|  |
| --- |
| import org.apache.hadoop.fs.Path;  import org.apache.hadoop.io.IntWritable;  import org.apache.hadoop.io.Text;  import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;  import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;  import org.apache.hadoop.mapreduce.Job;  /\*  \* MapReduce jobs are typically implemented by using a driver class.  \* The purpose of a driver class is to set up the configuration for the  \* MapReduce job and to run the job.  \* Typical requirements for a driver class include configuring the input  \* and output data formats, configuring the map and reduce classes,  \* and specifying intermediate data formats.  \*  \* The following is the code for the driver class:  \*/  public class WordCount {  public static void main(String[] args) throws Exception {  /\*  \* The expected command-line arguments are the paths containing  \* input and output data. Terminate the job if the number of  \* command-line arguments is not exactly 2.  \*/  if (args.length != 2) {  System.out.printf(  "Usage: WordCount <input dir> <output dir>\n");  System.exit(-1);  }  /\*  \* Instantiate a Job object for your job's configuration.  \*/  Job job = new Job();    /\*  \* Specify the jar file that contains your driver, mapper, and reducer.  \* Hadoop will transfer this jar file to nodes in your cluster running  \* mapper and reducer tasks.  \*/  job.setJarByClass(WordCount.class);    /\*  \* Specify an easily-decipherable name for the job.  \* This job name will appear in reports and logs.  \*/  job.setJobName("Word Count");  /\*  \* Specify the paths to the input and output data based on the  \* command-line arguments.  \*/  FileInputFormat.setInputPaths(job, new Path(args[0]));  FileOutputFormat.setOutputPath(job, new Path(args[1]));  /\*  \* Specify the mapper and reducer classes.  \*/  job.setMapperClass(WordMapper.class);  job.setReducerClass(SumReducer.class);  /\*  \* For the word count application, the input file and output  \* files are in text format - the default format.  \*  \* In text format files, each record is a line delineated by a  \* by a line terminator.  \*  \* When you use other input formats, you must call the  \* SetInputFormatClass method. When you use other  \* output formats, you must call the setOutputFormatClass method.  \*/    /\*  \* For the word count application, the mapper's output keys and  \* values have the same data types as the reducer's output keys  \* and values: Text and IntWritable.  \*  \* When they are not the same data types, you must call the  \* setMapOutputKeyClass and setMapOutputValueClass  \* methods.  \*/  /\*  \* Specify the job's output key and value classes.  \*/  job.setOutputKeyClass(Text.class);  job.setOutputValueClass(IntWritable.class);  job.setPartitionerClass(MyPartitioner.class);  job.setNumReduceTasks(2);  /\*  \* Start the MapReduce job and wait for it to finish.  \* If it finishes successfully, return 0. If not, return 1.  \*/  boolean success = job.waitForCompletion(true);  System.exit(success ? 0 : 1);  }  } |

|  |
| --- |
| [training@localhost wordcount]$ **jar cvf WordCount.jar \*.class**  [training@localhost wordcount]$ ***hadoop jar WordCount.jar WordCount shakespeare/comedies wc3***  15/03/03 04:30:27 WARN mapred.JobClient: Use GenericOptionsParser for parsing the arguments. Applications should implement Tool for the same.  15/03/03 04:30:27 INFO input.FileInputFormat: Total input paths to process : 1  15/03/03 04:30:27 WARN snappy.LoadSnappy: Snappy native library is available  15/03/03 04:30:27 INFO snappy.LoadSnappy: Snappy native library loaded  15/03/03 04:30:28 INFO mapred.JobClient: Running job: job\_201503020010\_0003  15/03/03 04:30:29 INFO mapred.JobClient: map 0% reduce 0%  15/03/03 04:30:34 INFO mapred.JobClient: map 100% reduce 0%  15/03/03 04:30:40 INFO mapred.JobClient: map 100% reduce 66%  15/03/03 04:30:42 INFO mapred.JobClient: map 100% reduce 100%  15/03/03 04:30:42 INFO mapred.JobClient: Job complete: job\_201503020010\_0003  15/03/03 04:30:42 INFO mapred.JobClient: Counters: 32  15/03/03 04:30:42 INFO mapred.JobClient: File System Counters  15/03/03 04:30:42 INFO mapred.JobClient: FILE: Number of bytes read=7276081  15/03/03 04:30:42 INFO mapred.JobClient: FILE: Number of bytes written=11632345  15/03/03 04:30:42 INFO mapred.JobClient: FILE: Number of read operations=0  15/03/03 04:30:42 INFO mapred.JobClient: FILE: Number of large read operations=0  15/03/03 04:30:42 INFO mapred.JobClient: FILE: Number of write operations=0  15/03/03 04:30:42 INFO mapred.JobClient: HDFS: Number of bytes read=1784735  15/03/03 04:30:42 INFO mapred.JobClient: HDFS: Number of bytes written=166599  15/03/03 04:30:42 INFO mapred.JobClient: HDFS: Number of read operations=2  15/03/03 04:30:42 INFO mapred.JobClient: HDFS: Number of large read operations=0  15/03/03 04:30:42 INFO mapred.JobClient: HDFS: Number of write operations=3  15/03/03 04:30:42 INFO mapred.JobClient: Job Counters  15/03/03 04:30:42 INFO mapred.JobClient: Launched map tasks=1  15/03/03 04:30:42 INFO mapred.JobClient: Launched reduce tasks=3  15/03/03 04:30:42 INFO mapred.JobClient: Data-local map tasks=1  15/03/03 04:30:42 INFO mapred.JobClient: Total time spent by all maps in occupied slots (ms)=5062  15/03/03 04:30:42 INFO mapred.JobClient: Total time spent by all reduces in occupied slots (ms)=12821  15/03/03 04:30:42 INFO mapred.JobClient: Total time spent by all maps waiting after reserving slots (ms)=0  15/03/03 04:30:42 INFO mapred.JobClient: Total time spent by all reduces waiting after reserving slots (ms)=0  15/03/03 04:30:42 INFO mapred.JobClient: Map-Reduce Framework  15/03/03 04:30:42 INFO mapred.JobClient: Map input records=59497  15/03/03 04:30:42 INFO mapred.JobClient: Map output records=328969  15/03/03 04:30:42 INFO mapred.JobClient: Map output bytes=2977801  15/03/03 04:30:42 INFO mapred.JobClient: Input split bytes=119  15/03/03 04:30:42 INFO mapred.JobClient: Combine input records=0  15/03/03 04:30:42 INFO mapred.JobClient: Combine output records=0  15/03/03 04:30:42 INFO mapred.JobClient: Reduce input groups=16769  15/03/03 04:30:42 INFO mapred.JobClient: Reduce shuffle bytes=3635757  15/03/03 04:30:42 INFO mapred.JobClient: Reduce input records=328969  15/03/03 04:30:42 INFO mapred.JobClient: Reduce output records=16769  15/03/03 04:30:42 INFO mapred.JobClient: Spilled Records=986907  15/03/03 04:30:42 INFO mapred.JobClient: CPU time spent (ms)=2480  15/03/03 04:30:42 INFO mapred.JobClient: Physical memory (bytes) snapshot=339746816  15/03/03 04:30:42 INFO mapred.JobClient: Virtual memory (bytes) snapshot=1554337792  15/03/03 04:30:42 INFO mapred.JobClient: Total committed heap usage (bytes)=208474112 |



Part-r-00000 contains word that start with (a-m ,A-M) and Part –r-00001 contains letter from (n-z,N-Z)

# HBase

HBase is the Hadoop database, which provides random, real time read/write access to very large data. HBase is an open-source, distributed, versioned, column-oriented store modeled after Google's Bigtable

## HBase Data Model Overview

**Table:** Applications store data in a HBase table. Tables are made of rows and columns. Table cells are the intersection of row and column coordinates.

**Columns & Column Families:** Columns in HBase are grouped into column families. All column members of a column family have the same prefix. For example, the courses:history and courses:math columns are both members of the courses column family. Physically, all column family members are stored together in the filesystem. Because tuning and storage specifications are done at the column family level, it is recommended that all column family members have the same general access pattern and size characteristics.

**Cell Value:** A rowkey, column (column family & column), and version (timestamp) identifies a specific cell in HBase. HBase supports a "bytes-in/bytes-out" interface via Put, Get and Result, so anything that can be converted to an array of bytes can be stored as a value. Input can be strings, numbers, complex objects, or even images, as long as they can be rendered as bytes.

**Versions**: It is possible to have many versions of data within a cell, where the row and column are the same but the cell address differs only in its version dimension. A version is specified as a long integer (timestamp). The HBase version dimension is stored in decreasing order so when reading from a store file, the most recent values are found first.

**Row Key:** Table row keys are also byte arrays. Therefore almost anything can serve as a row key, from strings to binary representations of longs or even serialized data structures. Rows are lexicographically sorted with the lowest order appearing first in a table. The empty byte array is used to denote both the start and end of a table's namespace. All table accesses are via the table row key - its primary key.

**Row orient DB**

|  |  |  |  |
| --- | --- | --- | --- |
| Empno | Ename | gender | DOJ |
| 1 | shalaj | M | 10-may-10 |
| 2 | anuj | M | 15-jun-10 |

**Column orient DB**

|  |  |  |
| --- | --- | --- |
| Empno | 1 | 2 |
| Ename | shalaj | anuj |
| gender | M | M |
| DOJ | 10-may-10 | 15-jun-10 |

Row oriented

2,anuj,M,15-Jun-10

1,shalaj,M,10-May-10

Column oriented

10-jun-10,15-jun-10

M,M

Shalaj,anuj

1,2

Column oriented databases has implicit indexing on column

Capability to handle/processing Null values is very fast

Internally storage like as below (key value)

(1)1,(2)2

(1)Shalaj,(2)anuj

(1)10-jun-10,(2)15-jun-10

(1)M,(2)M

Zookeeper communicates with name node, job tracker etc. A centralized service used to maintain configuration information for HBase.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Bill Memo**  **BillNo:1**  **BillDate: 10-jun-11**  **CustomerName: someone**  **Address :some address**   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **BillNo** | **itemcode** | **itemName** | **Quanitity** | **Price** | | 1 | 1212 | Pen | 10 | 15 | | 1 | 3434 | Pencil | 22 | 10 | |

RDBMS contains data into more than one table with primary and foreign key concept as described below

**BillMaster**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **BillNo** | **BillDate** | **CustomerName** | **Address** | **Totals** |
| 1 | 10-jun-11 | Some one | Some Address | 370 |

**BillDetails**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **BillNo** | **itemcode** | **itemName** | **Quanitity** | **Price** |
| 1 | 1212 | Pen | 10 | 15 |
| 1 | 3434 | Pencil | 22 | 10 |

DeNormalize:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **BillNo** | **BillDate** | **CustomerName** | **Address** | **Totals** | **itemcode** | **itemName** | **Quanitity** | **Price** |
| 1 | 10-jun-11 | Some one | Some Address | 370 | 1212 | Pen | 10 | 15 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Itemcode1** | **itemName1** | **Quanitity1** | **Price1** |
| 3434 | Pencil | 22 | 10 |

In case of Hbase we can create column family

Most frequent access column should be kept under one column family for better performance

Table-> Bill (BillHeader (billno,billdate,cutomername,address),BillDetails (srno,itemcode,itemname,….)) - here BillHeader and BillDetails are column family

Hbase never delete the old data it insert new data with updated version , all data kept together , latest timestamp has the latest data.

By default 3 versions can be created per column family, we can configure it, more number of version take more space.

Hbase has TTL(time to live) concept , we can set TTL to column , if column exceed TTL it will automatically delete

Colum key= column family: column qualifier

Empty cells (nulls) are not stored , that is called sparsing

Hbase shell is used for entering into hbase prompt

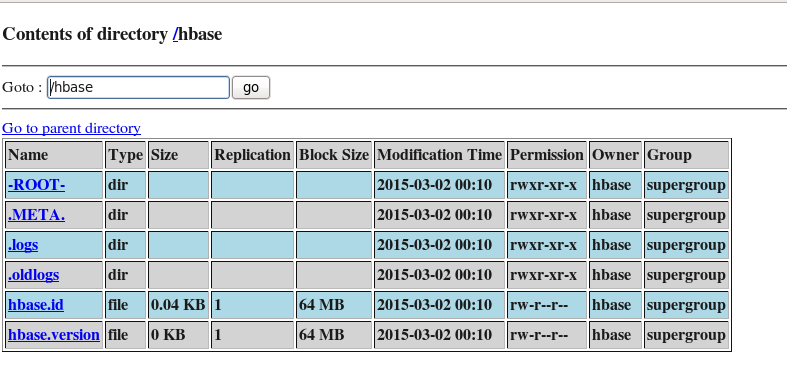
### Hbase prompt

|  |
| --- |
| [training@localhost wordcount]$ ***hbase shell***  15/03/04 02:15:32 WARN conf.Configuration: 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 0.92.1-cdh4.1.1, rUnknown, Tue Oct 16 11:44:19 PDT 2012  hbase(main):001:0> help  HBase Shell, version 0.92.1-cdh4.1.1, rUnknown, Tue Oct 16 11:44:19 PDT 2012  Type 'help "COMMAND"', (e.g. 'help "get"' -- the quotes are necessary) for help on a specific command.  Commands are grouped. Type 'help "COMMAND\_GROUP"', (e.g. 'help "general"') for help on a command group.  COMMAND GROUPS:  Group name: general  Commands: status, version, whoami  Group name: ddl  Commands: alter, alter\_async, alter\_status, create, describe, disable, disable\_all, drop, drop\_all, enable, enable\_all, exists, is\_disabled, is\_enabled, list, show\_filters  Group name: dml  Commands: count, delete, deleteall, get, get\_counter, incr, put, scan, truncate  Group name: tools  Commands: assign, balance\_switch, balancer, close\_region, compact, flush, hlog\_roll, major\_compact, move, split, unassign, zk\_dump  Group name: replication  Commands: add\_peer, disable\_peer, enable\_peer, list\_peers, remove\_peer, start\_replication, stop\_replication  Group name: security  Commands: grant, revoke, user\_permission  SHELL USAGE:  Quote all names in HBase Shell such as table and column names. Commas delimit  command parameters. Type <RETURN> after entering a command to run it.  Dictionaries of configuration used in the creation and alteration of tables are  Ruby Hashes. They look like this:  {'key1' => 'value1', 'key2' => 'value2', ...}  and are opened and closed with curley-braces. Key/values are delimited by the  '=>' character combination. Usually keys are predefined constants such as  NAME, VERSIONS, COMPRESSION, etc. Constants do not need to be quoted. Type  'Object.constants' to see a (messy) list of all constants in the environment.  If you are using binary keys or values and need to enter them in the shell, use  double-quote'd hexadecimal representation. For example:  hbase> get 't1', "key\x03\x3f\xcd"  hbase> get 't1', "key\003\023\011"  hbase> put 't1', "test\xef\xff", 'f1:', "\x01\x33\x40"  The HBase shell is the (J)Ruby IRB with the above HBase-specific commands added.  For more on the HBase Shell, see http://hbase.apache.org/docs/current/book.html |

### List all the tables

|  |
| --- |
| hbase(main):002:0> list  TABLE  0 row(s) in 0.9790 seconds |

All tables are stored into default hbase directory

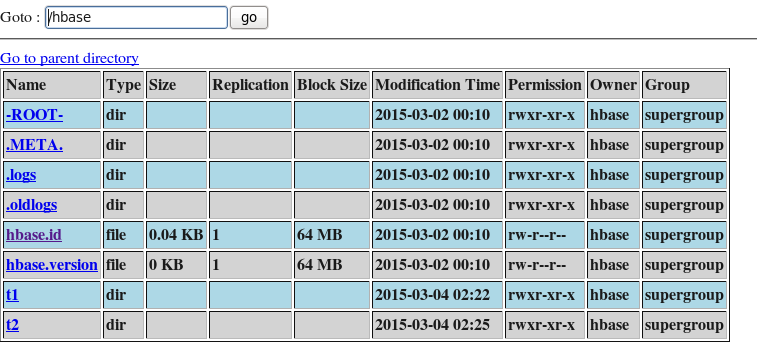


### Create and Describe Tables

Create table t1 , with one column family namely ‘fam1’ having only one version , if we omit version by default 3 versions created for the same

|  |
| --- |
| hbase(main):003:0> ***create 't1',{NAME => 'fam1',VERSION => 1}***  0 row(s) in 1.4150 seconds  hbase(main):005:0> ***describe 't1'***  DESCRIPTION ENABLED  {NAME => 't1', FAMILIES => [{NAME => 'fam1', BLOOMFILTER => 'NONE', REPLICATION\_SCOPE => '0', VERSIONS => true  '3', COMPRESSION => 'NONE', MIN\_VERSIONS => '0', TTL => '2147483647', BLOCKSIZE => '65536', IN\_MEMORY => '  false', BLOCKCACHE => 'true'}]}  1 row(s) in 0.0390 seconds |

|  |
| --- |
| hbase(main):006:0> ***create 't2', 'fam1','fam2'***  0 row(s) in 1.0940 seconds  hbase(main):007:0> ***describe 't2'***  DESCRIPTION ENABLED  {NAME => 't2', FAMILIES => [{NAME => 'fam1', BLOOMFILTER => 'NONE', REPLICATION\_SCOPE => '0', VERSIONS => true  '3', COMPRESSION => 'NONE', MIN\_VERSIONS => '0', TTL => '2147483647', BLOCKSIZE => '65536', IN\_MEMORY => '  false', BLOCKCACHE => 'true'}, {NAME => 'fam2', BLOOMFILTER => 'NONE', REPLICATION\_SCOPE => '0', VERSIONS  => '3', COMPRESSION => 'NONE', MIN\_VERSIONS => '0', TTL => '2147483647', BLOCKSIZE => '65536', IN\_MEMORY =  > 'false', BLOCKCACHE => 'true'}]}  1 row(s) in 0.0390 seconds  hbase(main):008:0> ***list***  TABLE  t1  t2  2 row(s) in 0.0270 seconds  hbase(main):074:0> ***create 'blogs', {NAME=>'meta',NAME=>'body',NAME=>'comment'}***  0 row(s) in 1.1270 seconds |



BLOOM Filter – concept to find out nonexistence record without scanning whole table

|  |
| --- |
| hbase(main):009:0> ***create 'users',{NAME => 'metadata'},{NAME => 'photo'}***  0 row(s) in 1.0880 seconds |

### Insert Data into table

To insert row on a table we need to use put command

Put ‘tablename’ ,’rowkey’,’colfam:col’,’value’ [timestamp]

|  |
| --- |
| hbase(main):012:0> ***put 'users','1234','metadata:name','shalaj'***  0 row(s) in 0.0320 seconds  hbase(main):014:0> ***put 'users','1234','photo:image','myimage'***  0 row(s) in 0.0350 seconds  hbase(main):016:0> ***scan 'users'***  ROW COLUMN+CELL  1234 column=metadata:name, timestamp=1425454946315, value=shalaj  1234 column=photo:image, timestamp=1425455096240, value=myimage  1 row(s) in 0.0480 seconds  hbase(main):019:0> ***put 'users','2222','metadata:name','Anuj'***  0 row(s) in 0.0040 seconds  hbase(main):021:0> ***scan 'users'***  ROW COLUMN+CELL  1234 column=metadata:name, timestamp=1425455324904, value=shalaj  1234 column=photo:image, timestamp=1425455096240, value=myimage  2222 column=metadata:name, timestamp=1425455303085, value=Anuj  2 row(s) in 0.0210 seconds  hbase(main):022:0> ***put 'users','12345','metadata:firstname','shalaj1'***  0 row(s) in 0.0250 seconds  hbase(main):023:0> ***put 'users','12345','metadata:lastname','shukla'***  0 row(s) in 0.0140 seconds  hbase(main):024:0> ***scan 'users'***  ROW COLUMN+CELL  1234 column=metadata:name, timestamp=1425455324904, value=shalaj  1234 column=photo:image, timestamp=1425455096240, value=myimage  12345 column=metadata:firstname, timestamp=1425455530875, value=shalaj1  12345 column=metadata:lastname, timestamp=1425455559245, value=shukla  2222 column=metadata:name, timestamp=1425455303085, value=Anuj  3 row(s) in 0.0350 seconds |

|  |
| --- |
| hbase(main):032:0> ***put 'users' ,'user1','metadata:fname','Doug'***  0 row(s) in 0.0290 seconds  hbase(main):033:0> ***put 'users' ,'user1','metadata:lname','Cutting'***  0 row(s) in 0.0110 seconds  hbase(main):034:0> ***put 'users' ,'user1','photo','<hadoop.gif>'***  0 row(s) in 0.0320 seconds  hbase(main):035:0***> scan 'users'***  ROW COLUMN+CELL  1234 column=metadata:name, timestamp=1425455324904, value=shalaj  1234 column=photo:image, timestamp=1425455096240, value=myimage  12345 column=metadata:firstname, timestamp=1425455530875, value=shalaj1  12345 column=metadata:lastname, timestamp=1425455559245, value=shukla  2222 column=metadata:name, timestamp=1425455303085, value=Anuj  user1 column=metadata:fname, timestamp=1425460309065, value=Doug  user1 column=metadata:lname, timestamp=1425460335592, value=Cutting  user1 column=photo:, timestamp=1425460363903, value=<hadoop.gif>  4 row(s) in 0.0470 seconds |

|  |
| --- |
| To update the current data we use put command only, if same row key and same column used during insertion it act like update command  hbase(main):036:0> ***put 'users' ,'user1','metadata:lname','Cutting1'***  0 row(s) in 0.0260 seconds  hbase(main):037:0> ***scan 'users'***  ROW COLUMN+CELL  1234 column=metadata:name, timestamp=1425455324904, value=shalaj  1234 column=photo:image, timestamp=1425455096240, value=myimage  12345 column=metadata:firstname, timestamp=1425455530875, value=shalaj1  12345 column=metadata:lastname, timestamp=1425455559245, value=shukla  2222 column=metadata:name, timestamp=1425455303085, value=Anuj  user1 column=metadata:fname, timestamp=1425460309065, value=Doug  user1 column=metadata:lname, timestamp=1425460822782, value=Cutting1  user1 column=photo:, timestamp=1425460363903, value=<hadoop.gif>  4 row(s) in 0.0540 seconds |

### Retrieving data from table

We can get only data associate with one rowkey

**get ‘tablename’,’rowkey’ [columns,timestamp,version]**

|  |
| --- |
| hbase(main):039:0> ***get 'users','user1',COLUMN=>'metadata:lname'***  COLUMN CELL  metadata:lname timestamp=1425460822782, value=Cutting1  1 row(s) in 0.0230 seconds  hbase(main):038:0> ***get 'users','user1',{COLUMN=>'metadata:lname',VERSIONS=>2}***  COLUMN CELL  metadata:lname timestamp=1425460822782, value=Cutting1  metadata:lname timestamp=1425460335592, value=Cutting  2 row(s) in 0.0500 seconds |

* Get the multiple columns

|  |
| --- |
| hbase(main):040:0> ***get 'users','user1',{COLUMN=>['metadata:lname','metadata:fname']}***  COLUMN CELL  metadata:fname timestamp=1425460309065, value=Doug  metadata:lname timestamp=1425460822782, value=Cutting1  2 row(s) in 0.0220 seconds |

* Get the all columns under specified column family

|  |
| --- |
| hbase(main):042:0> ***get 'users','user1','metadata'***  COLUMN CELL  metadata:fname timestamp=1425460309065, value=Doug  metadata:lname timestamp=1425460822782, value=Cutting1  2 row(s) in 0.0100 seconds |

* Get all columns from all column families

|  |
| --- |
| hbase(main):043:0> ***get 'users','user1'***  COLUMN CELL  metadata:fname timestamp=1425460309065, value=Doug  metadata:lname timestamp=1425460822782, value=Cutting1  photo: timestamp=1425460363903, value=<hadoop.gif>  3 row(s) in 0.0200 seconds |

* Scan with specific column

|  |
| --- |
| hbase(main):046:0> ***scan 'users',{COLUMNS=>'metadata:fname'}***  ROW COLUMN+CELL  user1 column=metadata:fname, timestamp=1425460309065, value=Doug  1 row(s) in 0.0260 seconds  hbase(main):049:0> ***scan 'users',{COLUMNS=>['metadata:fname','metadata:lname']}***  ROW COLUMN+CELL  user1 column=metadata:fname, timestamp=1425460309065, value=Doug  user1 column=metadata:lname, timestamp=1425460822782, value=Cutting1  1 row(s) in 0.0140 seconds |

* Scan with limit

|  |
| --- |
| hbase(main):047:0> **scan 'users',{STARTROW=>'user1',LIMIT=>10}**  ROW COLUMN+CELL  user1 column=metadata:fname, timestamp=1425460309065, value=Doug  user1 column=metadata:lname, timestamp=1425460822782, value=Cutting1  user1 column=photo:, timestamp=1425460363903, value=<hadoop.gif>  1 row(s) in 0.0910 seconds |

### Counting number of rows

Count ‘tablename’,[,interval}]

|  |
| --- |
| hbase(main):050:0> ***count 'users'***  4 row(s) in 0.0420 seconds |

### Delete Commands

Delete columns from a row

|  |
| --- |
| ***scan 'users'***  ROW COLUMN+CELL  1234 column=metadata:name, timestamp=1425455324904, value=shalaj  1234 column=photo:image, timestamp=1425455096240, value=myimage  12345 column=metadata:firstname, timestamp=1425455530875, value=shalaj1  12345 column=metadata:lastname, timestamp=1425455559245, value=shukla  2222 column=metadata:name, timestamp=1425455303085, value=Anuj  user1 column=metadata:fname, timestamp=1425460309065, value=Doug  user1 column=metadata:lname, timestamp=1425460822782, value=Cutting1  user1 column=photo:, timestamp=1425460363903, value=<hadoop.gif>  4 row(s) in 0.0290 seconds  hbase(main):051:0> ***delete 'users','2222','metadata:name'***  0 row(s) in 0.0300 seconds  hbase(main):052:0> ***scan 'users'***  ROW COLUMN+CELL  1234 column=metadata:name, timestamp=1425455324904, value=shalaj  1234 column=photo:image, timestamp=1425455096240, value=myimage  12345 column=metadata:firstname, timestamp=1425455530875, value=shalaj1  12345 column=metadata:lastname, timestamp=1425455559245, value=shukla  user1 column=metadata:fname, timestamp=1425460309065, value=Doug  user1 column=metadata:lname, timestamp=1425460822782, value=Cutting1  user1 column=photo:, timestamp=1425460363903, value=<hadoop.gif>  3 row(s) in 0.0310 seconds |

Delete an entire row

|  |
| --- |
| hbase(main):054:0> ***deleteall 'users','12345'***  0 row(s) in 0.0090 seconds  hbase(main):055:0> ***scan 'users'***  ROW COLUMN+CELL  1234 column=metadata:name, timestamp=1425455324904, value=shalaj  1234 column=photo:image, timestamp=1425455096240, value=myimage  user1 column=metadata:fname, timestamp=1425460309065, value=Doug  user1 column=metadata:lname, timestamp=1425460822782, value=Cutting1  user1 column=photo:, timestamp=1425460363903, value=<hadoop.gif>  2 row(s) in 0.0270 seconds |

Delete all rows

|  |
| --- |
| ***truncate ‘users’*** |

Drop a table

Need to disable first before dropping table

|  |
| --- |
| ***drop ‘users’*** |

### Changing Column families

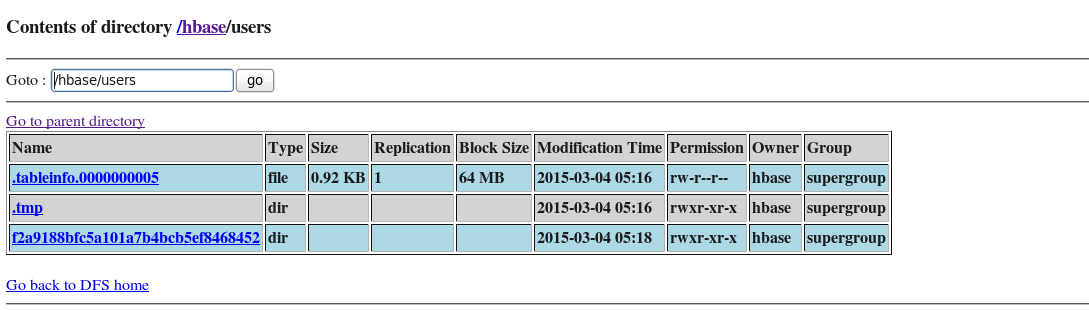
Table must be disabled first

|  |
| --- |
| hbase(main):056:0> ***disable 'users'***  0 row(s) in 2.0780 seconds  Add new column family  hbase(main):057:0> ***alter 'users',{NAME=>'newfamily'}***  Updating all regions with the new schema...  1/1 regions updated.  Done.  0 row(s) in 1.1730 seconds  hbase(main):058:0> ***describe 'users'***  DESCRIPTION ENABLED  {NAME => 'users', FAMILIES => [{NAME => 'metadata', BLOOMFILTER => 'NONE', REPLICATION\_SCOPE => '0', VERSI false  ONS => '3', COMPRESSION => 'NONE', MIN\_VERSIONS => '0', TTL => '2147483647', BLOCKSIZE => '65536', IN\_MEMO  RY => 'false', BLOCKCACHE => 'true'}, {NAME => 'newfamily', BLOOMFILTER => 'NONE', REPLICATION\_SCOPE => '0  ', COMPRESSION => 'NONE', VERSIONS => '3', TTL => '2147483647', MIN\_VERSIONS => '0', BLOCKSIZE => '65536',  IN\_MEMORY => 'false', BLOCKCACHE => 'true'}, {NAME => 'photo', BLOOMFILTER => 'NONE', REPLICATION\_SCOPE =  > '0', VERSIONS => '3', COMPRESSION => 'NONE', MIN\_VERSIONS => '0', TTL => '2147483647', BLOCKSIZE => '655  36', IN\_MEMORY => 'false', BLOCKCACHE => 'true'}]}  1 row(s) in 0.0220 seconds |

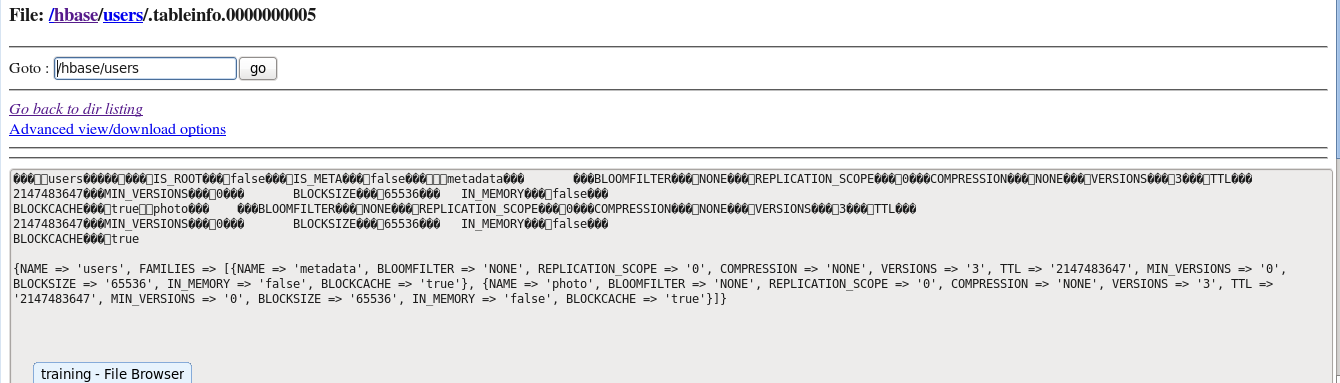
Delete existing column family -> all columns under that family will be removed

|  |
| --- |
| hbase(main):060:0> ***alter 'users',{NAME=>'newfamily',METHOD=>'delete'}***  Updating all regions with the new schema...  1/1 regions updated.  Done.  0 row(s) in 1.1940 seconds  hbase(main):061:0> ***describe 'users'***  DESCRIPTION ENABLED  {NAME => 'users', FAMILIES => [{NAME => 'metadata', BLOOMFILTER => 'NONE', REPLICATION\_SCOPE => '0', VERSI false  ONS => '3', COMPRESSION => 'NONE', MIN\_VERSIONS => '0', TTL => '2147483647', BLOCKSIZE => '65536', IN\_MEMO  RY => 'false', BLOCKCACHE => 'true'}, {NAME => 'photo', BLOOMFILTER => 'NONE', REPLICATION\_SCOPE => '0', V  ERSIONS => '3', COMPRESSION => 'NONE', MIN\_VERSIONS => '0', TTL => '2147483647', BLOCKSIZE => '65536', IN\_  MEMORY => 'false', BLOCKCACHE => 'true'}]}  1 row(s) in 0.0240 seconds  hbase(main):062:0> ***enable 'users'***  0 row(s) in 2.0720 seconds |

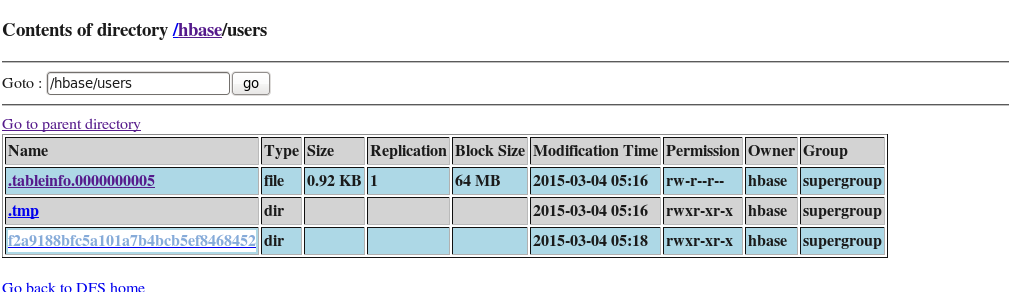
Table in Hbase is a folder under hbase folder in hadoop system , we can see it from hadoop NameNode URL

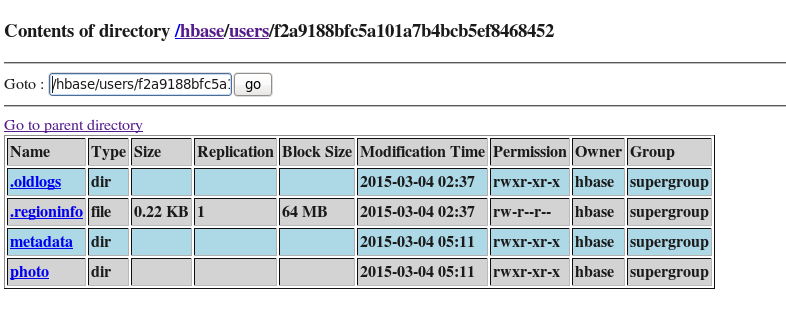


Here .tableinfo.0000000005 is folder containing table description



Random file name (f2a9188bfc5a101a7b4bcb5ef8468452) contains all the column family





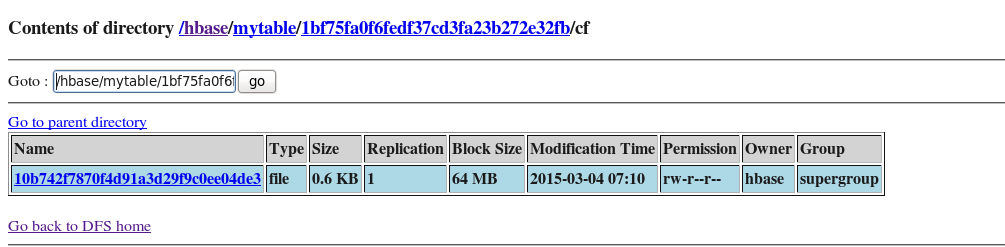
### Flush and Compaction

|  |
| --- |
| hbase(main):076:0> ***create 'mytable','cf'***  0 row(s) in 1.0690 seconds  [training@localhost ~]$ ***hadoop fs -ls /hbase/mytable/\*/cf***  [training@localhost ~]$  No data has been added so no Region Store files exist in the column Family directory |

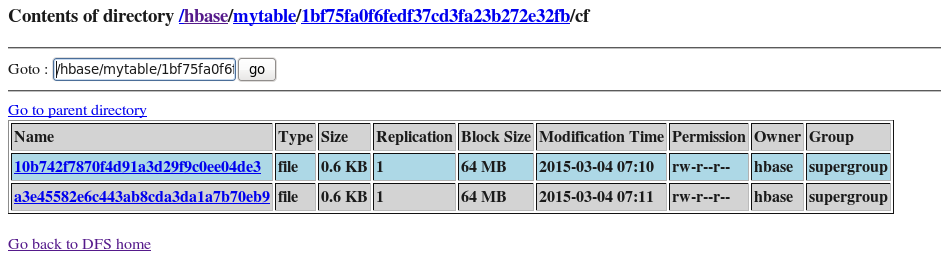
|  |
| --- |
| hbase(main):077:0> ***put 'mytable','row1','cf:col','foo'***  0 row(s) in 0.0240 seconds  [training@localhost ~]$ ***hadoop fs -ls /hbase/mytable/\*/cf***  [training@localhost ~]$  The put is in the Region Memstore and has not yet been flushed to disk |



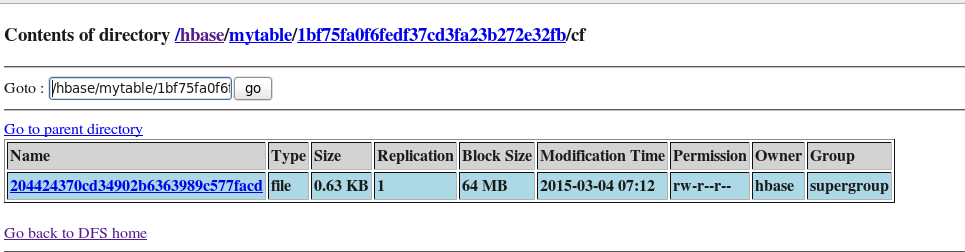
|  |
| --- |
| hbase(main):078:0> ***flush 'mytable'***  0 row(s) in 0.0730 seconds  [training@localhost ~]$ ***hadoop fs -ls /hbase/mytable/\*/cf***  Found 1 items  -rw-r--r-- 1 hbase supergroup 613 2015-03-04 06:46 /hbase/mytable/76c44257e77174d73b5a3f338a7cfbaa/cf/e563905770074fc0a771cd78145643ea  The flush has written the data in the memstore to a region store file. |



|  |
| --- |
| hbase(main):079:0> ***put 'mytable','row2','cf:col','bar'***  0 row(s) in 0.0100 seconds  hbase(main):080:0> flush 'mytable'  0 row(s) in 0.0670 seconds  [training@localhost ~]$ **hadoop fs -ls /hbase/mytable/\*/cf**  Found 2 items  -rw-r--r-- 1 hbase supergroup 613 2015-03-04 06:50 /hbase/mytable/76c44257e77174d73b5a3f338a7cfbaa/cf/0d709eb926a54259b20ad33b2b189dae  -rw-r--r-- 1 hbase supergroup 613 2015-03-04 06:46 /hbase/mytable/76c44257e77174d73b5a3f338a7cfbaa/cf/e563905770074fc0a771cd78145643ea  The second flush writes the contents of the Memstore into a new Region Store file |



|  |
| --- |
| hbase(main):081:0> ***major\_compact 'mytable'***  0 row(s) in 0.0660 seconds  [training@localhost ~]$ **hadoop fs -ls /hbase/mytable/\*/cf**  Found 1 items  -rw-r--r-- 1 hbase supergroup 646 2015-03-04 07:00 /hbase/mytable/76c44257e77174d73b5a3f338a7cfbaa/cf/07be9585fb8a4a649d61c18472bdcea3  A major compact will merge the smaller region store files into a single larger file. |



### Using Java API in HBase

|  |
| --- |
| base(main):106:0> ***create 'user', {NAME=>'info',VERSIONS=>1}***  0 row(s) in 1.0590 seconds  hbase(main):107:0> ***create 'USER', {NAME=>'INFO',VERSIONS=>1}***  0 row(s) in 1.0590 seconds |

|  |
| --- |
| import org.apache.hadoop.conf.Configuration;  import org.apache.hadoop.hbase.HBaseConfiguration;  import org.apache.hadoop.hbase.HColumnDescriptor;  import org.apache.hadoop.hbase.HTableDescriptor;  import org.apache.hadoop.hbase.client.HBaseAdmin;  import org.apache.hadoop.hbase.client.HTable;  import org.apache.hadoop.hbase.client.Put;  import org.apache.hadoop.hbase.client.Scan;  import org.apache.hadoop.hbase.client.ResultScanner;  import org.apache.hadoop.hbase.client.Result;  import org.apache.hadoop.hbase.util.Bytes;  import java.io.IOException;  public class MakeTable {  public static final byte[] TABLE\_NAME = "USER".getBytes();  public static final byte[] COLFAM\_NAME = "INFO".getBytes();  public static final byte[] COL\_VALUE = "NAME".getBytes();  public static void main(String[] args) throws IOException,  InterruptedException {  Configuration conf = HBaseConfiguration.create();  HBaseAdmin admin = new HBaseAdmin(conf);  if(admin.tableExists(TABLE\_NAME)) {  admin.disableTable(TABLE\_NAME);  admin.deleteTable(TABLE\_NAME);  }  HTableDescriptor desc = new HTableDescriptor(TABLE\_NAME);  HColumnDescriptor coldef = new HColumnDescriptor(COLFAM\_NAME);  desc.addFamily(coldef);  coldef.setMaxVersions(1);  admin.createTable(desc);  HTable userTable = new HTable(conf, TABLE\_NAME);  Put row1 = new Put(Bytes.toBytes("42"));  row1.add(COLFAM\_NAME, COL\_VALUE, Bytes.toBytes("Diana"));  Put row2 = new Put(Bytes.toBytes("43"));  row2.add(COLFAM\_NAME, COL\_VALUE, Bytes.toBytes("Doug"));  Put row3 = new Put(Bytes.toBytes("44"));  row3.add(COLFAM\_NAME, COL\_VALUE, Bytes.toBytes("Steve"));  userTable.put(row1);  userTable.put(row2);  userTable.put(row3);  admin.flush(TABLE\_NAME);  Scan userScan = new Scan();  ResultScanner scanner = userTable.getScanner(userScan);  for (Result result : scanner ) {  System.out.println(Bytes.toString(result.getValue(COLFAM\_NAME,  COL\_VALUE)));  }  userTable.close();  }  } |

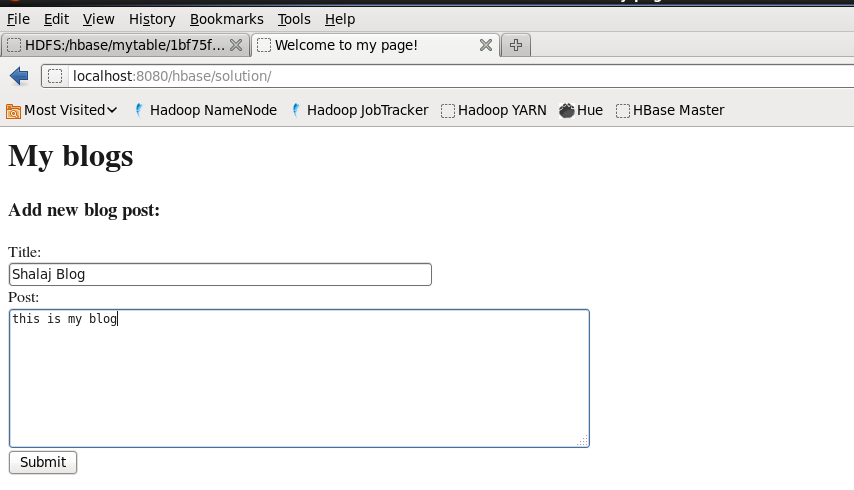
|  |
| --- |
| [training@localhost java]$ ***javac -classpath `hbase classpath`:`hadoop classpath` \*.java***  [training@localhost java]$ **jar cvf MakeTable.jar \*.class**  added manifest  adding: MakeTable.class(in = 2793) (out= 1451)(deflated 48%)  [training@localhost java]$ ***java -cp `hbase classpath`:`hadoop classpath`:MakeTable.jar MakeTable*** |

|  |
| --- |
| hbase(main):108:0> ***scan 'USER'***  ROW COLUMN+CELL  42 column=INFO:NAME, timestamp=1425535690278, value=Diana  43 column=INFO:NAME, timestamp=1425535690287, value=Doug  44 column=INFO:NAME, timestamp=1425535690288, value=Steve  3 row(s) in 0.0330 seconds |

**Web Application**

|  |
| --- |
| hbase(main):111:0> ***create 'blogs','meta','body','comment'*** |

|  |
| --- |
| [training@localhost java]$ ***sudo /etc/init.d/tomcat6 status***  tomcat6 is stopped [ OK ]  [training@localhost java]$ ***sudo /etc/init.d/tomcat6 start***  Starting tomcat6: [ OK ] |





|  |
| --- |
| hbase(main):110:0> **scan 'blogs'**  ROW COLUMN+CELL  123,9223370611318011719 column=body:, timestamp=1425536764091, value=this is my blog  123,9223370611318011719 column=meta:title, timestamp=1425536764091, value=Shalaj Blog  123,9223370611318011719,1425536790207 column=comment:comment\_author, timestamp=1425536790208, value=shalaj  123,9223370611318011719,1425536790207 column=comment:comment\_body, timestamp=1425536790208, value=my first comment  2 row(s) in 0.0330 seconds |

|  |
| --- |
| [training@localhost solution]$ ***pwd***  /usr/share/tomcat6/webapps/hbase/solution  [training@localhost solution]$ ***ls -ltr***  total 16  -rw-rw-rw- 1 1106 1104 1148 Dec 18 2012 newpost.jsp  -rw-rw-rw- 1 1106 1104 1077 Dec 18 2012 newcomment.jsp  -rw-rw-rw- 1 1106 1104 1691 Dec 18 2012 index.jsp  -rw-rw-rw- 1 1106 1104 2474 Dec 18 2012 blog.jsp |

**Index.jsp**

|  |
| --- |
| <%@ page import="java.io.\*,java.util.\*,org.apache.hadoop.hbase.\*,org.apache.hadoop.hbase.client.\*,org.apache.hadoop.hbase.util.\*,org.apache.hadoop.conf.\*,org.apache.zookeeper.\*"%>  <html>  <title>Welcome to my page!</title>  <h1>My blogs</h1>  <ul>  <%  // 1. Create a Configuration  Configuration conf = HBaseConfiguration.create();  // 2. Instantiate the HTable for the table "blogs" using conf  HTable table = new HTable(conf, "blogs");  Scan s = new Scan();  // 3. use s.addFamily to retreive the "meta" column family.  // // Hint: use the HBase utility method Bytes.toBytes to convert a String to byte[]  s.addFamily(Bytes.toBytes("meta"));  //4. Get the scanner  ResultScanner scanner = table.getScanner(s);  try {  for (Result row : scanner) {  //5. Find the rowkey and title of a row  String key = null;  String title = null;  NavigableMap<byte[],byte[]> map = row.getFamilyMap(Bytes.toBytes("meta"));  title = Bytes.toString(map.get(Bytes.toBytes("title")));  key = Bytes.toString(row.getRow());  Long reverseTimestamp= Long.parseLong(key.substring(4));  Long epoch = Math.abs(reverseTimestamp - Long.MAX\_VALUE);  Date dateOfPost = new Date(epoch);  out.println("<li><a href=\"blog.jsp?blogid=" + key + "\">" + title + " (" + dateOfPost + ")</a>");  }  } finally {  scanner.close();  }  %>  </ul>  <h3> Add new blog post:</h3>  <FORM METHOD="POST" ACTION="newpost.jsp">  Title:<br>  <INPUT TYPE="text" NAME="title" SIZE=50><br>  Post:<br>  <TEXTAREA NAME="body" ROWS=8 COLS=80></TEXTAREA>  <BR>  <INPUT TYPE="SUBMIT" VALUE="Submit">  <INPUT TYPE="hidden" NAME="blogid" VALUE="<%=request.getParameter("blogid")%>" >  </FORM>    </html> |

**blog.jsp**

|  |
| --- |
| <%@ page import="java.io.\*,java.util.\*,org.apache.hadoop.hbase.\*,org.apache.hadoop.hbase.client.\*,org.apache.hadoop.hbase.util.\*,org.apache.hadoop.hbase.filter.\*,org.apache.hadoop.conf.\*,org.apache.zookeeper.\*"%>  <html>  <title>Welcome to my page!</title>  <%  HBaseConfiguration hbaseconfig = new HBaseConfiguration();  HTable table = new HTable(hbaseconfig, "blogs");  String blogid = request.getParameter("blogid");  //1. Get the row whose row key is blogid from above  Get g = new Get(Bytes.toBytes(blogid));  Result r = table.get(g);  //2. Extract the rowkey, blog text (column "body") and blog title (column "meta:title")  byte [] key = r.getRow();  String keyStr = Bytes.toString(key);  String blogText = Bytes.toString(r.getValue(Bytes.toBytes("body"), Bytes.toBytes("")));  String blogTitle = Bytes.toString(r.getValue(Bytes.toBytes("meta"),Bytes.toBytes("title")));  Long reverseTimestamp= Long.parseLong(keyStr.substring(4));  Long epoch = Math.abs(reverseTimestamp - Long.MAX\_VALUE);  Date dateOfPost = new Date(epoch);  %>  <h1><%= blogTitle %></h1>  <%= dateOfPost %>  <br><br>  <%= blogText %>  <hr>  <h3>Comments:</h3>  <ul>  <%  // Scan for all comments belonging to this blog post  Scan s = new Scan();  s.addFamily(Bytes.toBytes("comment"));  // Use a PrefixFilter  PrefixFilter filter = new PrefixFilter(key);  s.setFilter(filter);  ResultScanner scanner = table.getScanner(s);  try {  for (Result row: scanner) {  %>  <li>  <%  NavigableMap<byte[],byte[]> map = row.getFamilyMap(Bytes.toBytes("comment"));  keyStr = Bytes.toString(row.getRow());  String commentAuthor = Bytes.toString(map.get(Bytes.toBytes("comment\_author")));  String commentBody = Bytes.toString(map.get(Bytes.toBytes("comment\_body")));  Long timeOfComment= Long.parseLong(keyStr.substring(keyStr.lastIndexOf(",")+1));  Date dateOfComment = new Date(timeOfComment);  out.println(commentAuthor + " said...<br>");  out.println(commentBody + "<br>");  out.println(dateOfComment + "<br><br>");  }  } finally {  scanner.close();  }  %>  </ul>  <!-- print new comment form -->  <h3>Add comment:</h3>  <FORM METHOD="POST" ACTION="newcomment.jsp">  <table>  <tr><td>Your name:</td><td><INPUT TYPE="TEXT" NAME="author" SIZE="50"></td></tr>  <tr><td>Comment: </td><td> <INPUT TYPE="TEXT" NAME="body" SIZE="50"></td></tr>  </table>  <INPUT TYPE="SUBMIT" VALUE="Submit">  <INPUT TYPE="hidden" NAME="blogid" VALUE="<%=request.getParameter("blogid")%>" >  </FORM>  <p>  <a href="index.jsp">Home</a>  </p>  </html> |

**newpost.jsp**

|  |
| --- |
| <%@ page import="java.io.\*,java.util.\*,org.apache.hadoop.hbase.\*,org.apache.hadoop.hbase.client.\*,org.apache.hadoop.hbase.util.\*,org.apache.hadoop.conf.\*,org.apache.zookeeper.\*"%>  <%  HBaseConfiguration hbaseconfig = new HBaseConfiguration();  HTable table = new HTable(hbaseconfig, "blogs");  //1. Create a Put object with a key that looks like "123,9223370761422928544"  // 123 represents the blogger id  // The other value is a reverse timestamp, which should be computed using System.currentTimeMillis()  Put newrow = new Put(Bytes.toBytes("123," + (Long.MAX\_VALUE - System.currentTimeMillis())));  String newPostTitle = request.getParameter("title");  String newPostBody = request.getParameter("body");  //2. Add the value of newPostTitle to column "meta:title" and  // the value of newPostBody to the column "body:"  newrow.add(Bytes.toBytes("meta"), Bytes.toBytes("title"), Bytes.toBytes(newPostTitle));  newrow.add(Bytes.toBytes("body"), Bytes.toBytes(""), Bytes.toBytes(newPostBody));  table.put(newrow);  %>  <html>  <title>Thanks</title>  <h3>This has been published:</h3>  <p>  <%= newPostBody %>  </p>  <a href="index.jsp">Return to list</a>  </html> |

**newcomment.jsp**

|  |
| --- |
| <%@ page import="java.io.\*,java.util.\*,org.apache.hadoop.hbase.\*,org.apache.hadoop.hbase.client.\*,org.apache.hadoop.hbase.util.\*,org.apache.hadoop.conf.\*,org.apache.zookeeper.\*"%>  <%  String blogid = request.getParameter("blogid");  HBaseConfiguration hbaseconfig = new HBaseConfiguration();  HTable table = new HTable(hbaseconfig, "blogs");  //1. Create a Put object with key: the blogid (from above) + a comma + the current timestamp  Put newrow = new Put(Bytes.toBytes(blogid +","+ System.currentTimeMillis()));  //2. Add the value of author to column "comment:comment\_author" and  // the value of body to the column "comment:comment\_body"  String author = request.getParameter("author");  String body = request.getParameter("body");  newrow.add(Bytes.toBytes("comment"), Bytes.toBytes("comment\_author"), Bytes.toBytes(author));  newrow.add(Bytes.toBytes("comment"), Bytes.toBytes("comment\_body"), Bytes.toBytes(body));  table.put(newrow);  %>  <html>  <title>Thanks</title>  <h3>Your comment has been submited</h3>  <a href="blog.jsp?blogid=<%=blogid%>">Return to blog</a>  </html> |

# HIVE Overview

Hive is an open-source data warehousing solution built on top of Hadoop.

Hive provides an SQL dialect, called Hive Query Language (abbreviated HiveQL or just HQL) for querying data stored in a Hadoop cluster.

Go to hive prompt

|  |
| --- |
| [training@localhost wordcount]$ ***hive***  Logging initialized using configuration in file:/etc/hive/conf.dist/hive-log4j.properties  Hive history file=/tmp/training/hive\_job\_log\_training\_201503090150\_1441292576.txt  hive> |

## Create DataBases

|  |
| --- |
| hive> ***CREATE DATABASE financials;***  OK  Time taken: 1.021 seconds  hive> ***CREATE DATABASE human\_resources;***  OK  Time taken: 0.146 seconds  hive***> SHOW DATABASES;***  OK  default  financials  human\_resources  Time taken: 0.102 seconds  hive>  hive> ***SHOW DATABASES LIKE 'h.\*';***  OK  human\_resources  Time taken: 0.112 seconds |

We can execute hive command from outside the hive prompt by using hive –e option

|  |
| --- |
| [training@localhost solution]$ ***hive -e 'show DATABASES';***  Logging initialized using configuration in file:/etc/hive/conf.dist/hive-log4j.properties  Hive history file=/tmp/training/hive\_job\_log\_training\_201503090200\_765420669.txt  OK  **default**  financials  human\_resources  Time taken: 3.458 seconds  [training@localhost solution]$ |

All databases created under /user/hive/warehouse folder, default database is created with name **default**

|  |
| --- |
| [training@localhost solution]$ ***hdfs dfs -ls /user/hive***  Found 1 items  drwxrwxrwx - hue supergroup 0 2015-03-09 01:54 /user/hive/warehouse  [training@localhost solution]$ |

|  |
| --- |
| [training@localhost solution]$ ***hdfs dfs -ls /user/hive/warehouse***  Found 2 items  drwxr-xr-x - training supergroup 0 2015-03-09 01:54 /user/hive/warehouse/financials.db  drwxr-xr-x - training supergroup 0 2015-03-09 01:54 /user/hive/warehouse/human\_resources.db |

|  |
| --- |
| hive> ***USE financials;***  OK  Time taken: 0.149 seconds  hive> ***set hive.cli.print.current.db=true;***  hive (financials)>  hive (financials)> ***USE default;***  OK  Time taken: 0.043 seconds  hive (default)> |

## Hive column types

Each type maps to a native data type in Java.

|  |  |
| --- | --- |
| **Type** | **Description** |
| TINYINT | 1 byte |
| SMALLINT | 2 bytes |
| INT | 4 bytes |
| BIGINT | 8 bytes |
| FLOAT | Single precision |
| DOUBLE | Double precision |
| STRING | Sequence of characters |
| BOOLEAN | True/false |
| TIMESTAMP | YYYY/MM/DD HH:MM:SS.fffffffff |

**Collection Data Types**

|  |  |  |
| --- | --- | --- |
| **Type** | **Description** | **Literal syntax examples** |
| STRUCT | Analogous to a C struct or an “object.”  Fields can be accessed using the “dot” notation.  For example, if a column name is of  type STRUCT {first STRING; last STRING}, then  the first name field can be referenced using name.first. | struct('John', 'Doe') |
| MAP | A collection of key-value tuples, where the fields are accessed using array notation (e.g., ['key']).  For example, if a column name is of type MAP with key→value pairs 'first'→'John' and 'last'→'Doe', then the last  name can be referenced using name['last']. | map('first', 'John',  'last', 'Doe') |
| ARRAY | Ordered sequences of the same type that are indexable using zero-based integers.  For example, if a column name is of type  ARRAY of strings with the value ['John', 'Doe'], then  the second element can be referenced using name[1]. | array('John', 'Doe') |

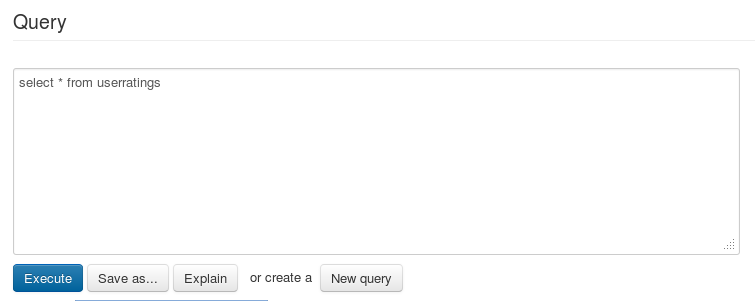
|  |
| --- |
| **CREATE TABLE** employees (  name STRING,  salary FLOAT,  subordinates ARRAY<STRING>,  deductions **MAP**<STRING, FLOAT>,  address STRUCT<street:STRING, city:STRING, **state**:STRING, zip:INT>  )  **ROW** FORMAT DELIMITED  FIELDS TERMINATED **BY** '\001'  COLLECTION ITEMS TERMINATED **BY** '\002'  **MAP** KEYS TERMINATED **BY** '\003'  LINES TERMINATED **BY** '\n'  STORED **AS** TEXTFILE; |

|  |
| --- |
| [training@localhost tomcat6]$ ***cd /home/training/training\_materials/hive\_and\_pig/data/***  [training@localhost data]$ ***ls -ltr***  total 7388  -rw-rw-r-- 1 training training 2505716 Dec 18 2012 training.sql  -rw-rw-r-- 1 training training 23876 Dec 18 2012 thriller.txt  -rwxrwxr-x 1 training training 4948405 Dec 18 2012 ml-data.tgz  -rw-rw-r-- 1 training training 51553 Dec 18 2012 comedy.txt  -rw-rw-r-- 1 training training 25038 Dec 18 2012 action.txt  [training@localhost data]$ ***tar xvfz ml-data.tgz***  ml-data/  ml-data/README  ml-data/allbut.pl  ml-data/mku.sh  ml-data/u.data  ml-data/u.genre  ml-data/u.info  ml-data/u.item  ml-data/u.occupation  ml-data/u.user  ml-data/ub.test  ml-data/u1.test  ml-data/u1.base  ml-data/u2.test  ml-data/u2.base  ml-data/u3.test  ml-data/u3.base  ml-data/u4.test  ml-data/u4.base  ml-data/u5.test  ml-data/u5.base  ml-data/ua.test  ml-data/ua.base  ml-data/ub.base  [training@localhost data]$  [training@localhost data]$ ***cd ml-data***  [training@localhost ml-data]$ ***ls -ltr***  total 15776  -rw-rw---- 1 training training 6401 Jul 19 2000 README  -rwxrwx--- 1 training training 643 Jul 19 2000 mku.sh  -rwxrwx--- 1 training training 716 Jul 19 2000 allbut.pl  -rw-rw---- 1 training training 22628 Jul 19 2000 u.user  -rw-rw---- 1 training training 193 Jul 19 2000 u.occupation  -rw-rw---- 1 training training 236344 Jul 19 2000 u.item  -rw-rw---- 1 training training 36 Jul 19 2000 u.info  -rw-rw---- 1 training training 202 Jul 19 2000 u.genre  -rw-rw---- 1 training training 1979173 Jul 19 2000 u.data  -rw-rw---- 1 training training 392629 Mar 8 2001 u1.test  -rw-rw---- 1 training training 1586544 Mar 8 2001 u1.base  -rw-rw---- 1 training training 395225 Mar 8 2001 u2.test  -rw-rw---- 1 training training 1583948 Mar 8 2001 u2.base  -rw-rw---- 1 training training 396627 Mar 8 2001 u3.test  -rw-rw---- 1 training training 1582546 Mar 8 2001 u3.base  -rw-rw---- 1 training training 397295 Mar 8 2001 u4.test  -rw-rw---- 1 training training 1581878 Mar 8 2001 u4.base  -rw-rw---- 1 training training 397397 Mar 8 2001 u5.test  -rw-rw---- 1 training training 1581776 Mar 8 2001 u5.base  -rw-rw---- 1 training training 1792501 Mar 8 2001 ua.base  -rw-rw---- 1 training training 186672 Mar 8 2001 ua.test  -rw-rw---- 1 training training 1792476 Mar 8 2001 ub.base  -rw-rw---- 1 training training 186697 Mar 8 2001 ub.test  [training@localhost ml-data]$ ***hadoop fs -copyFromLocal u.data /user/training***  [training@localhost ml-data]$ ***hadoop fs -copyFromLocal u.item /user/training***  [training@localhost ml-data]$ ***hadoop fs -put u.user /user/training***  [training@localhost ml-data]$ ***hadoop fs -put u.info /user/training***  [training@localhost ml-data]$ |

|  |
| --- |
| hive (default)> CREATE TABLE UserRatings (userid INT,movieid INT,rating INT,unixtime BIGINT)  > ROW FORMAT DELIMITED  > FIELDS TERMINATED BY '\t'  > STORED AS TEXTFILE;  OK  Time taken: 0.506 seconds  hive (default)> ***DESCRIBE UserRatings;***  OK  userid int  movieid int  rating int  unixtime bigint  Time taken: 0.144 seconds |

|  |
| --- |
| hive (default)> ***LOAD DATA INPATH '/user/training/u.data'***  ***> INTO TABLE UserRatings;***  Loading data to table default.userratings  OK  Time taken: 0.49 seconds  hive (default)> |

Go to HUE URL and execute query





|  |
| --- |
| hive (default)> ***select \* from userratings limit 10;***  OK  196 242 3 881250949  186 302 3 891717742  22 377 1 878887116  244 51 2 880606923  166 346 1 886397596  298 474 4 884182806  115 265 2 881171488  253 465 5 891628467  305 451 3 886324817  6 86 3 883603013  Time taken: 0.187 seconds  hive (default)> |

|  |
| --- |
| [training@localhost ml-data]$ ***hadoop fs -ls /user/hive/warehouse/userratings***  Found 1 items  -rw-r--r-- 1 training supergroup 1979173 2015-03-09 02:54 /user/hive/warehouse/userratings/u.data |

1. Create employee table in hive , field seprated by |

|  |
| --- |
| hive (default)> CREATE TABLE Employee (employeeid INT,employeename STRING,salary INT)  > ROW FORMAT DELIMITED  > FIELDS TERMINATED BY '|'  > STORED AS TEXTFILE;  OK  Time taken: 0.21 seconds |

1. Create employee.data file in local system , field separated by ‘|’

|  |
| --- |
| [training@localhost ml-data]$ vi employee.data  123|abc|10000  456|xyz|20000  789|pqr|30000  111|aaa|40000  222|bbb|50000  333|ccc|60000  444|ddd|70000  555|eee|80000 |

1. Put this file into hadoop system

|  |
| --- |
| training@localhost ml-data]$ ***hadoop fs -put employee.data /user/training*** |

1. Load data from this file to employee table from hive prompt

|  |
| --- |
| hive (default)> ***LOAD DATA INPATH '/user/training/employee.data' INTO TABLE employee;***  Loading data to table default.employee  OK  Time taken: 0.322 seconds |

## External Table

It is possible to create a table which points to an existing directory rather than having the directory automatically created under **/user/hive/warehouse**.

|  |
| --- |
| **CREATE EXTERNAL TABLE** movies (id INT, name STRING, year INT)  **ROW FORMAT DELIMITED**  **FIELDS TERMINATED BY** ','  **STORED AS TEXTFILE**  **LOCATION** ‘ /user/ian/movieFileDirectory ‘ |

**DROPing** an External Table **only removes metadata**.

**Notes** :

–Dropping a **regular table** deletes the table‟s directory .

–Dropping an **external table** simply removes the table‟s definition from the Hive Metastore,The directory is not deleted

1. Create hql file

|  |
| --- |
| [training@localhost ml-data]$ ***vi User\_Record.hql***  CREATE EXTERNAL TABLE User\_Record(  USER\_ID INT,  Age INT,  Sex STRING,  Occupation STRING,  Zipcode BIGINT)  ROW FORMAT DELIMITED  FIELDS TERMINATED BY '|'  STORED AS TEXTFILE  LOCATION '/user/training/external'; |

1. Put this file under hadoop system

|  |
| --- |
| training@localhost ml-data]$ ***hadoop fs -put User\_Record.hql /user/training*** |

Below command creates table and external directory as well

1. Execute hive –f to run the file

|  |
| --- |
| [training@localhost ml-data]$ ***hive -f User\_Record.hql***  Logging initialized using configuration in file:/etc/hive/conf.dist/hive-log4j.properties  Hive history file=/tmp/training/hive\_job\_log\_training\_201503090400\_436615408.txt  OK  Time taken: 3.954 seconds |

1. Check if external directory created under /user/training

|  |
| --- |
| [training@localhost ml-data]$ ***hadoop fs -ls /user/training***  Found 13 items  -rw-r--r-- 1 training supergroup 209 2015-03-09 03:58 /user/training/User\_Record.hql  drwxr-xr-x - training supergroup 0 2015-03-09 04:00 **/user/training/external**  -rw-r--r-- 1 training supergroup 2132 2015-03-02 06:12 /user/training/passwd  [training@localhost ml-data]$ |

1. Copy u.user under external directory

|  |
| --- |
| [training@localhost ml-data]$ ***hadoop fs -cp /user/training/u.user /user/training/external***  [training@localhost ml-data]$ |

1. Check if file copied

|  |
| --- |
| [training@localhost ml-data]$ ***hadoop fs -ls /user/training/external***  Found 1 items  -rw-r--r-- 1 training supergroup 22628 2015-03-09 04:02 /user/training/external/u.user  [training@localhost ml-data]$ |

1. Execute select query in hive prompt

|  |
| --- |
| hive (default)> ***select \* from User\_Record LIMIT 10;***  OK  1 24 M technician 85711  2 53 F other 94043  3 23 M writer 32067  4 24 M technician 43537  5 33 F other 15213  6 42 M executive 98101  7 57 M administrator 91344  8 36 M administrator 5201  9 29 M student 1002  10 53 M lawyer 90703  Time taken: 0.118 seconds |

So we load the data by just copying the file in external directory, without executing load data command

# Sqoop

* Sqoop is a tool designed to transfer data between Hadoop and relational databases.
* You can use Sqoop to import data from a relational database management system (RDBMS) such as MySQL or Oracle into the Hadoop Distributed File System (HDFS), transform the data in Hadoop MapReduce, and then export the data back into an RDBMS.
* Sqoop automates most of this process, relying on the database to describe the schema for the data to be imported.
* Sqoop uses MapReduce to import and export the data, which provides parallel operation as well as fault tolerance.

Sqoop syntax

|  |
| --- |
| sqoop import \  --username user \  --password pass \  --connect jdbc:mysql://dbserver.example.com/db \  --hive-import \  --fields-terminated-by '\t' \  --table t1 |

## Example to migrate data from mysql to hive

Go to mysql prompt

|  |
| --- |
| [training@localhost ~]$ ***mysql -u training -p***  Enter password:  Welcome to the MySQL monitor. Commands end with ; or \g.  Your MySQL connection id is 210  Server version: 5.1.66 Source distribution  Copyright (c) 2000, 2012, Oracle and/or its affiliates. All rights reserved.  Oracle is a registered trademark of Oracle Corporation and/or its  affiliates. Other names may be trademarks of their respective  owners.  Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.  mysql> |

|  |
| --- |
| mysql> use training  Reading table information for completion of table and column names  You can turn off this feature to get a quicker startup with -A  Database changed  mysql> DESCRIBE Movies;  +---------------+--------------+------+-----+---------+-------+  | Field | Type | Null | Key | Default | Extra |  +---------------+--------------+------+-----+---------+-------+  | movieid | int(11) | NO | PRI | 0 | |  | movie\_name | varchar(255) | YES | | NULL | |  | release\_date | char(11) | YES | | NULL | |  | imdb\_url | varchar(255) | YES | | NULL | |  | unknown\_genre | tinyint(4) | YES | | NULL | |  | action | tinyint(4) | YES | | NULL | |  | adventure | tinyint(4) | YES | | NULL | |  | animation | tinyint(4) | YES | | NULL | |  | children | tinyint(4) | YES | | NULL | |  | comedy | tinyint(4) | YES | | NULL | |  | crime | tinyint(4) | YES | | NULL | |  | documentary | tinyint(4) | YES | | NULL | |  | drama | tinyint(4) | YES | | NULL | |  | fantasy | tinyint(4) | YES | | NULL | |  | film\_noir | tinyint(4) | YES | | NULL | |  | horror | tinyint(4) | YES | | NULL | |  | musical | tinyint(4) | YES | | NULL | |  | mystery | tinyint(4) | YES | | NULL | |  | romance | tinyint(4) | YES | | NULL | |  | sci\_fi | tinyint(4) | YES | | NULL | |  | thriller | tinyint(4) | YES | | NULL | |  | war | tinyint(4) | YES | | NULL | |  | western | tinyint(4) | YES | | NULL | |  +---------------+--------------+------+-----+---------+-------+  23 rows in set (0.00 sec) |

Execute sqoop import from dollor prompt

|  |
| --- |
| [training@localhost ml-data]$ ***sqoop import \***  ***> --username training \***  ***> --password training \***  ***> --connect jdbc:mysql://localhost/training \***  ***> --hive-import \***  ***> --fields-terminated-by '\001' \***  ***> --table Movies***  15/03/09 05:43:21 WARN tool.BaseSqoopTool: Setting your password on the command-line is insecure. Consider using -P instead.  15/03/09 05:43:21 INFO manager.MySQLManager: Preparing to use a MySQL streaming resultset.  15/03/09 05:43:21 INFO tool.CodeGenTool: Beginning code generation  15/03/09 05:43:21 INFO manager.SqlManager: Executing SQL statement: SELECT t.\* FROM `Movies` AS t LIMIT 1  15/03/09 05:43:22 INFO manager.SqlManager: Executing SQL statement: SELECT t.\* FROM `Movies` AS t LIMIT 1  15/03/09 05:43:22 INFO orm.CompilationManager: HADOOP\_HOME is /usr/lib/hadoop  Note: /tmp/sqoop-training/compile/8f63adf5383b83def4f01a4b34b5b71b/Movies.java uses or overrides a deprecated API.  Note: Recompile with -Xlint:deprecation for details.  15/03/09 05:43:23 INFO orm.CompilationManager: Writing jar file: /tmp/sqoop-training/compile/8f63adf5383b83def4f01a4b34b5b71b/Movies.jar  15/03/09 05:43:23 WARN manager.MySQLManager: It looks like you are importing from mysql.  15/03/09 05:43:23 WARN manager.MySQLManager: This transfer can be faster! Use the --direct  15/03/09 05:43:23 WARN manager.MySQLManager: option to exercise a MySQL-specific fast path.  15/03/09 05:43:23 INFO manager.MySQLManager: Setting zero DATETIME behavior to convertToNull (mysql)  15/03/09 05:43:24 INFO mapreduce.ImportJobBase: Beginning import of Movies  15/03/09 05:43:26 WARN mapred.JobClient: Use GenericOptionsParser for parsing the arguments. Applications should implement Tool for the same.  15/03/09 05:43:28 INFO db.DataDrivenDBInputFormat: BoundingValsQuery: SELECT MIN(`movieid`), MAX(`movieid`) FROM `Movies`  15/03/09 05:43:29 INFO mapred.JobClient: Running job: job\_201503020010\_0005  15/03/09 05:43:30 INFO mapred.JobClient: map 0% reduce 0%  15/03/09 05:43:52 INFO mapred.JobClient: map 50% reduce 0%  15/03/09 05:44:01 INFO mapred.JobClient: map 100% reduce 0%  15/03/09 05:44:04 INFO mapred.JobClient: Job complete: job\_201503020010\_0005  15/03/09 05:44:04 INFO mapred.JobClient: Counters: 23  15/03/09 05:44:04 INFO mapred.JobClient: File System Counters  15/03/09 05:44:04 INFO mapred.JobClient: FILE: Number of bytes read=0  15/03/09 05:44:04 INFO mapred.JobClient: FILE: Number of bytes written=797596  15/03/09 05:44:04 INFO mapred.JobClient: FILE: Number of read operations=0  15/03/09 05:44:04 INFO mapred.JobClient: FILE: Number of large read operations=0  15/03/09 05:44:04 INFO mapred.JobClient: FILE: Number of write operations=0  15/03/09 05:44:04 INFO mapred.JobClient: HDFS: Number of bytes read=450  15/03/09 05:44:04 INFO mapred.JobClient: HDFS: Number of bytes written=234674  15/03/09 05:44:04 INFO mapred.JobClient: HDFS: Number of read operations=4  15/03/09 05:44:04 INFO mapred.JobClient: HDFS: Number of large read operations=0  15/03/09 05:44:04 INFO mapred.JobClient: HDFS: Number of write operations=4  15/03/09 05:44:04 INFO mapred.JobClient: Job Counters  15/03/09 05:44:04 INFO mapred.JobClient: Launched map tasks=4  15/03/09 05:44:04 INFO mapred.JobClient: Total time spent by all maps in occupied slots (ms)=54290  15/03/09 05:44:04 INFO mapred.JobClient: Total time spent by all reduces in occupied slots (ms)=0  15/03/09 05:44:04 INFO mapred.JobClient: Total time spent by all maps waiting after reserving slots (ms)=0  15/03/09 05:44:04 INFO mapred.JobClient: Total time spent by all reduces waiting after reserving slots (ms)=0  15/03/09 05:44:04 INFO mapred.JobClient: Map-Reduce Framework  15/03/09 05:44:04 INFO mapred.JobClient: Map input records=1682  15/03/09 05:44:04 INFO mapred.JobClient: Map output records=1682  15/03/09 05:44:04 INFO mapred.JobClient: Input split bytes=450  15/03/09 05:44:04 INFO mapred.JobClient: Spilled Records=0  15/03/09 05:44:04 INFO mapred.JobClient: CPU time spent (ms)=2300  15/03/09 05:44:04 INFO mapred.JobClient: Physical memory (bytes) snapshot=182530048  15/03/09 05:44:04 INFO mapred.JobClient: Virtual memory (bytes) snapshot=1552662528  15/03/09 05:44:04 INFO mapred.JobClient: Total committed heap usage (bytes)=63963136  15/03/09 05:44:04 INFO mapreduce.ImportJobBase: Transferred 0 bytes in 39.5671 seconds (0 bytes/sec)  15/03/09 05:44:04 INFO mapreduce.ImportJobBase: Retrieved 1682 records.  15/03/09 05:44:04 INFO manager.SqlManager: Executing SQL statement: SELECT t.\* FROM `Movies` AS t LIMIT 1  15/03/09 05:44:05 INFO hive.HiveImport: Removing temporary files from import process: hdfs://0.0.0.0:8020/user/training/Movies/\_logs  15/03/09 05:44:05 INFO hive.HiveImport: Loading uploaded data into Hive  15/03/09 05:44:08 INFO hive.HiveImport: Logging initialized using configuration in file:/etc/hive/conf.dist/hive-log4j.properties  15/03/09 05:44:08 INFO hive.HiveImport: Hive history file=/tmp/training/hive\_job\_log\_training\_201503090544\_2068862058.txt  15/03/09 05:44:14 INFO hive.HiveImport: OK  15/03/09 05:44:14 INFO hive.HiveImport: Time taken: 5.571 seconds  15/03/09 05:44:14 INFO hive.HiveImport: Loading data to table default.movies  15/03/09 05:44:14 INFO hive.HiveImport: OK  15/03/09 05:44:14 INFO hive.HiveImport: Time taken: 0.227 seconds  15/03/09 05:44:14 INFO hive.HiveImport: Hive import complete.  15/03/09 05:44:14 INFO hive.HiveImport: Export directory is empty, removing it. |

Now come to hive prompt and execute select query

|  |
| --- |
| hive (default)> ***select \* from MOVIES limit 10;***  OK  1 Toy Story (1995) 01-Jan-1995 http://us.imdb.com/M/title-exact?Toy%20Story%20(1995) 0 0 0 1 1 1 0 0 00 0 0 0 0 0 0 0 0 0  2 GoldenEye (1995) 01-Jan-1995 http://us.imdb.com/M/title-exact?GoldenEye%20(1995) 0 1 1 0 0 0 0 0 00 0 0 0 0 0 0 1 0 0  3 Four Rooms (1995) 01-Jan-1995 http://us.imdb.com/M/title-exact?Four%20Rooms%20(1995) 0 0 0 0 0 0 0 0 00 0 0 0 0 0 0 1 0 0  4 Get Shorty (1995) 01-Jan-1995 http://us.imdb.com/M/title-exact?Get%20Shorty%20(1995) 0 1 0 0 0 1 0 0 10 0 0 0 0 0 0 0 0 0  5 Copycat (1995) 01-Jan-1995 http://us.imdb.com/M/title-exact?Copycat%20(1995) 0 0 0 0 0 0 1 0 1 00 0 0 0 0 0 1 0 0  6 Shanghai Triad (Yao a yao yao dao waipo qiao) (1995) 01-Jan-1995 http://us.imdb.com/Title?Yao+a+yao+yao+dao+waipo+qiao+(1995) 0 0 0 00 0 0 0 1 0 0 0 0 0 0 0 0 0 0  7 Twelve Monkeys (1995) 01-Jan-1995 http://us.imdb.com/M/title-exact?Twelve%20Monkeys%20(1995) 0 0 0 0 0 0 0 01 0 0 0 0 0 0 1 0 0 0  8 Babe (1995) 01-Jan-1995 http://us.imdb.com/M/title-exact?Babe%20(1995) 0 0 0 0 1 1 0 0 1 0 00 0 0 0 0 0 0 0  9 Dead Man Walking (1995) 01-Jan-1995 http://us.imdb.com/M/title-exact?Dead%20Man%20Walking%20(1995) 0 0 0 0 0 0 0 01 0 0 0 0 0 0 0 0 0 0  10 Richard III (1995) 22-Jan-1996 http://us.imdb.com/M/title-exact?Richard%20III%20(1995) 0 0 0 0 0 0 0 0 10 0 0 0 0 0 0 0 1 0  Time taken: 0.153 seconds |

Note: we haven’t created any table on hive , it only copied the data into /user/hive/wareshouse folder

|  |
| --- |
| [training@localhost ml-data]$ ***hadoop fs -ls /user/hive/warehouse***  Found 5 items  drwxr-xr-x - training supergroup 0 2015-03-09 03:40 /user/hive/warehouse/employee  drwxr-xr-x - training supergroup 0 2015-03-09 01:54 /user/hive/warehouse/financials.db  drwxr-xr-x - training supergroup 0 2015-03-09 01:54 /user/hive/warehouse/human\_resources.db  drwxr-xr-x - training supergroup 0 2015-03-09 05:44 /user/hive/warehouse/movies  drwxr-xr-x - training supergroup 0 2015-03-09 03:11 /user/hive/warehouse/userratings  [training@localhost ml-data]$ ***hadoop fs -ls /user/hive/warehouse/movies***  Found 5 items  -rw-r--r-- 1 training supergroup 0 2015-03-09 05:44 /user/hive/warehouse/movies/\_SUCCESS  -rw-r--r-- 1 training supergroup 58800 2015-03-09 05:43 /user/hive/warehouse/movies/part-m-00000  -rw-r--r-- 1 training supergroup 58986 2015-03-09 05:43 /user/hive/warehouse/movies/part-m-00001  -rw-r--r-- 1 training supergroup 58458 2015-03-09 05:43 /user/hive/warehouse/movies/part-m-00002  -rw-r--r-- 1 training supergroup 58430 2015-03-09 05:43 /user/hive/warehouse/movies/part-m-00003 |

Check Movies structure at Hive prompt , column type get changed automatically

|  |
| --- |
| hive (default)> describe Movies;  OK  movieid int  movie\_name string  release\_date string  imdb\_url string  unknown\_genre tinyint  action tinyint  adventure tinyint  animation tinyint  children tinyint  comedy tinyint  crime tinyint  documentary tinyint  drama tinyint  fantasy tinyint  film\_noir tinyint  horror tinyint  musical tinyint  mystery tinyint  romance tinyint  sci\_fi tinyint  thriller tinyint  war tinyint  western tinyint  Time taken: 0.132 seconds  hive (default)> |

To display column name along with values set ***hive.cli.print.header=true;***

|  |
| --- |
| hive (default)> ***set hive.cli.print.header=true;***  hive (default)> select \* from movies limit 1;  OK  movieid movie\_name release\_date imdb\_url unknown\_genre action adventure animation children comedy crime documentary drama fantasy film\_noir horror musical mystery romance sci\_fi thriller war western  1 Toy Story (1995) 01-Jan-1995 http://us.imdb.com/M/title-exact?Toy%20Story%20(1995) 0 0 0 1 1 1 0 0 00 0 0 0 0 0 0 0 0 0  Time taken: 0.128 seconds |

**INSERT OVERWRITE**

Create table /copy data from existing table

Copies data from one Hive table into another table

Overwrites contents of second table

|  |
| --- |
| INSERT OVERWRITE TABLE t2 SELECT \* FROM t; |

INSERT INTO

* Like INSERT OVERWRITE, it copies data from one table into another
* However, it appends to (rather than overwrite) the contents of the second table
* INSERT INTO is only supported in Hive versions 0.8 and later

|  |
| --- |
| INSERT INTO TABLE t2 SELECT \* FROM t; |

Configuration files related to hive

|  |
| --- |
| [training@localhost ml-data]$ ***sudo su***  [root@localhost ml-data]# ***cd /etc/hive***  [root@localhost hive]# ls  conf conf.dist  [root@localhost hive]# ***cd conf***  [root@localhost conf]# ***ls -ltr***  total 72  -rw-r--r--. 1 root root 2966 Oct 16 2012 hive-log4j.properties  -rw-r--r--. 1 root root 2561 Oct 16 2012 hive-exec-log4j.properties  -rw-r--r--. 1 root root 2378 Oct 16 2012 hive-env.sh.template  -rw-r--r--. 1 root root 53457 Oct 16 2012 hive-default.xml.template  -rw-r--r--. 1 root root 2291 Dec 18 2012 hive-site.xml |