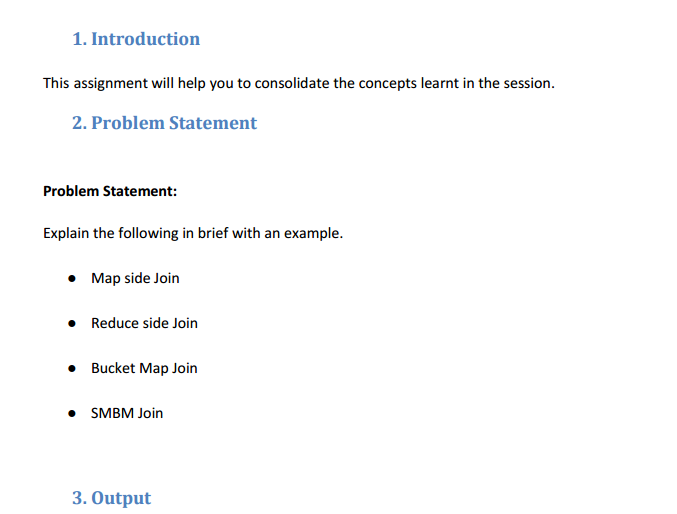
**Assignment 27.1**



* **Map Side Join**
* In case one of the dataset is small, map side join takes place.
* In map side join, a local job runs to create hash-table from content of HDFS file and sends it to every node.

**SET hive.auto.convert.join=true;**

The inputs for to each map must be partitioned and sorted in a specific way. Each input dataset must be divided into the  
same number of partitions, and it must be sorted by the same key (the join key) in each source.

All the records for a particular key must reside in the same partition and which is mandatory. A map-side join can be used to  
join the outputs of several jobs that had the same number of reducers, the same keys and output files that are no bigger than the HDFS block size.

Using the org.apache.hadoop.mapred.join.CompositeInputFormat class we can achieve this. The join type (Inner or Outer) is configurable using  
the join expression.

**For example :**

**func ::= tbl(<class>,”<path>”);**

We can set it to the CompositeInputFormat using,

inner(tbl(org.apache.hadoop.mapreduce.lib.input.SequenceFileInputFormat.class,  
“hdfs://localhost:8000/usr/data”),  
tbl(org.apache.hadoop.mapreduce.lib.input.SequenceFileInputFormat.class,  
“hdfs://localhost:8000/usr/activity”));

We can achieve following kind of joins using Map-Side techniques,

1) Inner Join  
2) Outer Join  
3) Override – MultiFilter for a given key, prefered values from the right source

* **Reduce side Join**

If datasets are large, reduce side join takes place.Reduce-Side joins are simpler than Map-Side joins since the input datasets need not to be structured. But it is less efficient as both datasets have to go through the MapReduce shuffle phase. The records with the same key are brought together in the reducer. We can also use the Secondary Sort technique to control the order of the records.

SET hive.auto.convert.join=true;

* **Bucket Map Join**

The data must be bucketed on the keys used in the ON clause and the number of buckets for one table must be a multiple of the number of buckets for the other table.

When these conditions are met, Hive can join individual buckets between tables in the map phase, because it does not have to fetch the entire content of one table to match against each bucket in the other table.

• set hive.optimize.bucketmapjoin=true;

• SET hive.auto.convert.join=true;

* **SMBM Join**

Sort-Merge-Bucket (SMB) joins can be converted to SMB map joins as well.SMB joins are used wherever the tables are sorted and bucketed. The join boils down to just merging the already sorted tables, allowing this operation to be faster than an ordinary map-join.

• set hive.enforce.sortmergebucketmapjoin=false;

• set hive.auto.convert.sortmerge.join=true;

• set hive.optimize.bucketmapjoin = true;

• set hive.optimize.bucketmapjoin.sortedmerge = true;

In normal join, mappers read data of tables on which join needs to be performed and emit key as join key or column on which is expected to be performed . Thus MapReduce framework distributes join key data to single reducer

In normal Join in hive joining will take place is reducer but since in this joins joins will be made through buffer memory (which contains small dataset) and large dataset will be streamed in reducer which will actually takes long time

Instead of this we can use Map Side Join where joining will be take place in mapper itself

Which will results in faster execution of query as this takes place in mapper through distributed cache

In order to do map side join set.hive.auto.convert=true;

set.hive.nonconditional.task=true;