**ASSIGNMENT 27.1**

**Explain the following in brief with an example.**

**● Map side Join**

**● Reduce side Join**

**● Bucket Map Join**

**● SMBM Join**

**MAP SIDE JOIN:**

As the name indicates, the map side join is performed in the mapper phase of the map reduce process.

In this join we are storing the small file in distributed cache and another file’s input path has to be given in the argument.

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.mapreduce.lib.join.CompositeInputFormat class we can achieve this.

**Advantages of using map side join:**

* Map-side join helps in minimizing the cost that is incurred for sorting and merging in the shuffle and reduce stages.
* Map-side join also helps in improving the performance of the task by decreasing the time to finish the task.

**Disadvantages of Map-side join:**

* Map side join is adequate only when one of the tables on which you perform map-side join operation is small enough to fit into the memory.  Hence it is not suitable to perform map-side join on the tables which are huge data in both of them.

**REDUCE SIDE JOIN:**

As the name indicates, the reduce side join is performed in the reducer phase of the map reduce process.

It is comparatively simple and easier to implement than the map side join as the sorting and shuffling phase sends the values having identical keys to the same reducer and therefore, by default, the data is organized for us.

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.

* Mapper reads the input data which are to be combined based on common column or join key.
* The mapper processes the input and adds a tag to the input to distinguish the input belonging from different sources or data sets or databases.
* The mapper outputs the intermediate key-value pair where the key is nothing but the join key.
* After the sorting and shuffling phase, a key and the list of values is generated for the reducer.
* Now, the reducer joins the values present in the list with the key to give the final aggregated output.

The reduce side join procedure generates a huge network I/O traffic in the sorting and reducer phase where the values of the same key are brought together. So, if you have a large number of different data sets having millions of values, there is a high chance that you will encounter an Out Of Memory Exception i.e. Your RAM is full and therefore, overflown.

**ADVANTAGES:**

* It is very easy to implement as we are taking advantage of the inbuilt sorting and shuffling algorithm in the MapReduce framework which combines values of the same key and send it to the same reducer.
* In the reduce side join, input does not require to follow any strict format and therefore, join operation can be performed on unstructured data as well.

**BUCKET MAP JOIN:**

In bucketing, the data at storage level is distributed in buckets. Each bucket is expected to hold certain rows based on the bucketing key/column.

Let us consider two tables **user and user\_visits.** both table data is bucketed using **user\_id** in 4 buckets . It means bucket 1 of user will contain rows with same user ids as that of bucket 1 of user\_visits. And if a join is performed on these two tables on user\_id columns, if it is possible to send bucket 1 of both tables to same mapper then good amount of optimization can be achieved. This is exactly done in bucketed map join.

**Prerequisites for bucket map join:**

1. Tables being joined are bucketized on the **join columns**.
2. The **number of buckets in one table is a multiple of the number of buckets in the other table**, the buckets can be joined with each other

**Before doing all, we should set**

hive.optimize.bucketmapjoin = **true**

**SMBM JOIN:**

It is another Hive join optimization technique where all the tables need to be bucketed and sorted. In this case joins are very efficient because they require a simple merge of the presorted tables.

Before doing anything, we should set

set hive.enforce.bucketing=true;

set hive.enforce.sorting=true;

set hive.enforce.sortmergebucketmapjoin=false;

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

set hive.optimize.bucketmapjoin = true;

set hive.optimize.bucketmapjoin.sortedmerge = true;

In sort merge bucket Map join while creating the bucketed table we should sort the contents by the column which you are going to cluster by.

In SMB join in Hive, each mapper reads a bucket from the first table and the corresponding bucket from the second table and then a merge sort join is performed. Sort Merge Bucket (SMB) join in hive is mainly used as there is no limit on file or partition or table join. SMB join can best be used when the tables are large. In SMB join the columns are bucketed and sorted using the join columns. All tables should have the same number of buckets in SMB join.

sort merge bucket map join is the fastest of all the joins.