**Big Data Frameworks CSE3120**

Lab – 5 Map Reduce Join Experiment

**Name:** Naveen Nidadavolu

**Roll No:** 22MIA1049

**Map Reduce Join**

**Aim:** To perform a join operation using MapReduce on two datasets—students.txt and marks.txt—to combine student names with their respective marks based on Student\_ID.

**Algorithm**

1. **Initialize MapReduce Job**

Set job configuration, input paths, and output path.

Specify mapper, reducer, and output key-value classes.

1. **StudentMapper (Map Phase)**

For each line:

* 1. Split by space.
  2. Emit (Student\_ID, "STUDENT:Name").

1. **MarksMapper (Map Phase)**

For each line:

* 1. Split by space.
  2. Emit (Student\_ID, "MARKS:Marks").

1. **Shuffle and Sort Phase**

Group records by Student\_ID.

1. **JoinReducer (Reduce Phase)**

For each key:

* 1. Initialize name and marks.
  2. For each value:
     1. If prefix is STUDENT:, assign to name.
     2. If prefix is MARKS:, assign to marks.
  3. Emit (Student\_ID, "Name Marks") if both present.

1. **Write Output**

Output the joined records to the specified directory.

1. **Complete Job**
2. Wait for job completion and exit.

**Procedure**

1. Create input text files (students.txt and marks.txt) and upload them to HDFS.
2. Run the MapReduce join program to process the input files.
3. The Mapper reads and tags records from both files based on Student\_ID.
4. The Reducer merges records with the same Student\_ID to produce the final joined output.
5. The output is stored in HDFS.
6. Retrieve and display the results.

**Program**

**import java.io.IOException;**

**import org.apache.hadoop.conf.Configuration;**

**import org.apache.hadoop.fs.Path;**

**import org.apache.hadoop.io.Text;**

**import org.apache.hadoop.mapreduce.Job;**

**import org.apache.hadoop.mapreduce.Mapper;**

**import org.apache.hadoop.mapreduce.Reducer;**

**import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;**

**import org.apache.hadoop.mapreduce.lib.input.MultipleInputs;**

**import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;**

**import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;**

**public class MapReduceJoin {**

**public static class StudentMapper extends Mapper<Object, Text, Text, Text> {**

**public void map(Object key, Text value, Context context) throws IOException, InterruptedException {**

**String[] tokens = value.toString().split("\\s+");**

**if (tokens.length == 2 && !tokens[0].equals("Student\_ID")) { // Ignore header**

**context.write(new Text(tokens[0]), new Text("STUDENT:" + tokens[1]));**

**}**

**}**

**}**

**public static class MarksMapper extends Mapper<Object, Text, Text, Text> {**

**public void map(Object key, Text value, Context context) throws IOException, InterruptedException {**

**String[] tokens = value.toString().split("\\s+");**

**if (tokens.length == 2 && !tokens[0].equals("Student\_ID")) { // Ignore header**

**context.write(new Text(tokens[0]), new Text("MARKS:" + tokens[1]));**

**}**

**}**

**}**

**public static class JoinReducer extends Reducer<Text, Text, Text, Text> {**

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

**String name = "";**

**String marks = "";**

**for (Text val : values) {**

**String value = val.toString();**

**if (value.startsWith("STUDENT:")) {**

**name = value.split(":")[1];**

**} else if (value.startsWith("MARKS:")) {**

**marks = value.split(":")[1];**

**}**

**}**

**if (!name.isEmpty() && !marks.isEmpty()) {**

**context.write(key, new Text(name + "\t" + marks));**

**}**

**}**

**}**

**public static void main(String[] args) throws Exception {**

**Configuration conf = new Configuration();**

**Job job = Job.getInstance(conf, "MapReduce Join");**

**job.setJarByClass(MapReduceJoin.class);**

**job.setReducerClass(JoinReducer.class);**

**job.setOutputKeyClass(Text.class);**

**job.setOutputValueClass(Text.class);**

**MultipleInputs.addInputPath(job, new Path(args[0]), TextInputFormat.class, StudentMapper.class);**

**MultipleInputs.addInputPath(job, new Path(args[1]), TextInputFormat.class, MarksMapper.class);**

**FileOutputFormat.setOutputPath(job, new Path(args[2]));**

**System.exit(job.waitForCompletion(true) ? 0 : 1);**

**}**

**}**

**1.Input files**

* File 1 (Students): Student\_ID Name
* File 2 (Marks): Student\_ID Marks

A screenshot of a computer

AI-generated content may be incorrect.

**2.Running MapReduceJoin Program**

A computer screen with white text

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

**Output:**

A screenshot of a computer

AI-generated content may be incorrect.

**Result:** Successfully performed a join operation using MapReduce, merging student details with their marks based on Student\_ID.