



HMIN122M Entrepôts de données et big-data

TP Hadoop / Map-Reduce

Auteur:

Gracia-Moulis Kévin (21604392) Canta Thomas (21607288)

Master 1 - AIGLE/DECOL Faculté des sciences de Montpellier Année universitaire 2020/2021

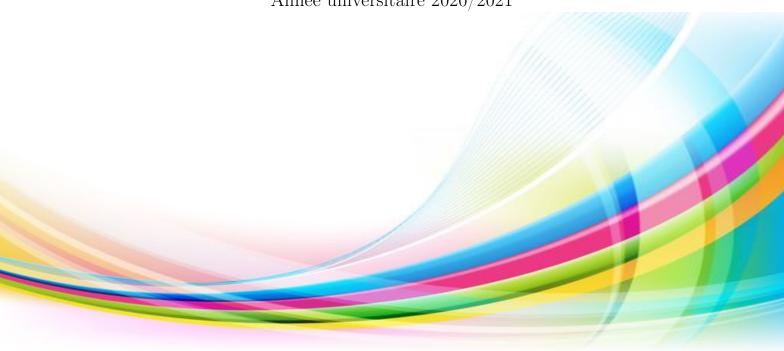


Table des matières

Partie 1		2
Exercice	1	 2
Exercice	3	 2
Exercice	1	 3
Noi	abre de produit différent vendu	 3
Noi	abre total d'exemplaire d'un produit vendu	 4
Mo	tant des ventes par Date / Category	 4
Exercice	Ď	 4

Partie 1

Exercice 1

Modifier la fonction reduce du programme WordCount.java afin que seulement les mots dont le nombre d'occurrences est supérieur ou égal à deux soient affichés :

```
public void reduce(Text key, Iterable <IntWritable > values, Context context)
throws IOException, InterruptedException {
   int sum = 0;
   for (IntWritable val : values)
      sum += val.get();
   // Exercice 1
   if (sum >= 2)
      context.write(key, new IntWritable(sum));
}
```

Listing 1 – Modification de la fonction reduce

Exercice 3

Compléter le code dans la classe Group By.java qui est fournie afin d'implémenter un opérateur de regroupement sur l'attribut Customer-ID du fichier de données fourni dans le répertoire input-group By :

```
o private final static String emptyWords[] = { "" };
  @Override
  public void map(LongWritable key, Text value, Context context) throws
     IOException, InterruptedException {
    String line = value.toString();
    String[] words = line.split(",");
5
    if (Arrays.equals(words, emptyWords))
      return:
8
    String word = words [5];
10
    double number = Double.parseDouble(words[20]);
11
    DoubleWritable write = new DoubleWritable(number);
12
    context.write(new Text(word), write);
13
14 }
```

Listing 2 – Fonction map de GroupBy

```
public void reduce(Text key, Iterable < DoubleWritable > values, Context context)
    throws IOException, InterruptedException {
    double sum = 0;

    for(DoubleWritable val : values)
        sum += val.get();

    context.write(key, new DoubleWritable(sum));

}
```

Listing 3 – Fonction reduce de GroupBy

Exercice 4

Modifier le programme précédent afin de calculer le montant des ventes par Date et State :

Il n'est pas nécessaire de modifier la fonction reduce.

```
o public void map(LongWritable key, Text value, Context context) throws
     IOException, Interrupted Exception {
    String line = value.toString();
    String[] words = line.split(",");
3
    if (Arrays.equals(words, emptyWords))
4
      return;
    String word1 = words [2]; // Date
    String word2 = words[10]; // State
    String word = word1 + "||" + word2;
    double number = Double.parseDouble(words[17]); // Sales
10
    DoubleWritable write = new DoubleWritable(number);
11
    context.write(new Text(word), write);
12
13 }
```

Listing 4 – Fonction map de GroupBy

Nombre de produit différent vendu

Il suffit de modifier notre variable write.

Listing 5 – Fonction map de GroupBy

Nombre total d'exemplaire d'un produit vendu

Listing 6 – Fonction map de GroupBy

Montant des ventes par Date / Category

Il suffit de modifier notre variable word2 comme étant la catégorie.

Listing 7 – Fonction map de GroupBy

Exercice 5

Sur la base des programmes WordCount.java et GroupBy.java, définir une classe Join.java permettant de joindre les lignes concernant les informations des clients et des commandes contenus dans le répertoire input-join :

```
o public static class Map extends Mapper<LongWritable, Text, Text, Text> {
    private final static String emptyWords[] = { "" };
2
    @Override
3
    public void map(LongWritable key, Text value, Context context) throws
      IOException, InterruptedException {
      String line = value.toString();
5
      String [] words = line.split (" \setminus |");
6
      if (Arrays.equals(words, emptyWords))
8
         return;
9
10
      if (words.length == 8) // on Customer
11
         context.write(new Text(words[0]), new Text("||"+words[1]));
12
13
      else // on Order
14
         context.write(new Text(words[1]), new Text(words[8]));
15
16
17 }
```

Listing 8 – Fonction map de Join

```
o public static class Reduce extends Reducer<Text, Text, Text, Text, Text> {
    @Override
2
    public void reduce(Text key, Iterable < Text > values, Context context)
3
         throws IOException , InterruptedException {
4
5
6
       ArrayList < String > cust = new ArrayList <>();
       ArrayList < String > comment = new ArrayList <>();
7
       for (Text val : values) {
9
         String line = val.toString();
10
         String [] words = line.split (" \setminus | \setminus | ");
11
12
         if (words.length == 2) cust.add(words[1]);
13
         else comment.add(words[0]);
14
15
       for(String _cust : cust)
17
         for (String _comment : comment)
18
           context.write(new Text(_cust), new Text(_comment));
19
20
21 }
```

Listing 9 – Fonction reduce de Join