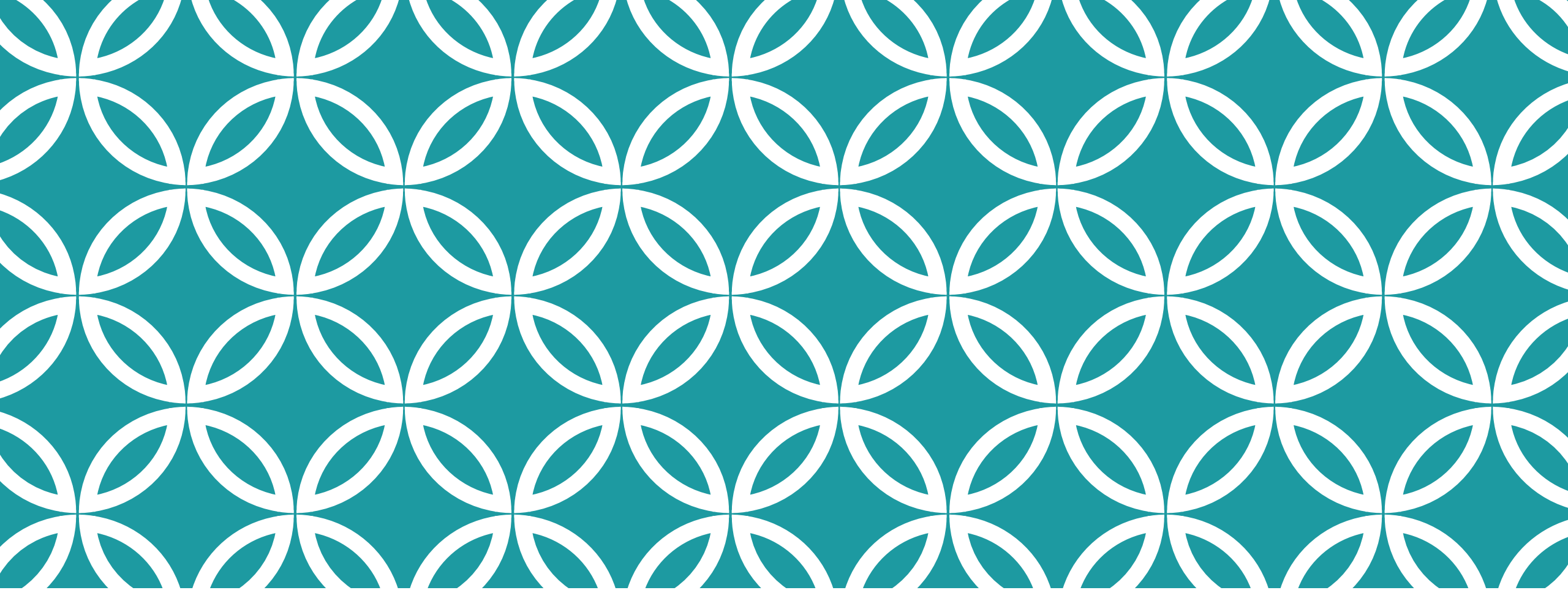


Pattern de jointure

Inner join

User ID	Reputation	Location
3	3738	New York, NY
4	12946	New York, NY
5	17556	San Diego, CA
9	3443	Oakland, CA

User ID	Post ID	Text
3	35314	Not sure why this is getting downvoted.
3	48002	Hehe, of course, it's all true!
5	44921	Please see my post below.
5	44920	Thank you very much for your reply.
8	48675	HTML is not a subset of XML!

A.User ID	A.Reputation	A.Location	B.User ID	B.Post ID	B.Text
3	3738	New York, NY	3	35314	Not sure why this is getting downvoted.
3	3738	New York, NY	3	48002	Hehe, of course, it's all true!
5	17556	San Diego, CA	5	44921	Please see my post below.
5	17556	San Diego, CA	5	44920	Thank you very much for your reply.

Inner JOIN A + B sur la clé ID: Tous les éléments qui ont la même valeur de ID sont fusionnés

Left outer join

User ID	Reputation	Location
3	3738	New York, NY
4	12946	New York, NY
5	17556	San Diego, CA
9	3443	Oakland, CA

User ID	Post ID	Text
3	35314	Not sure why this is getting downvoted.
3	48002	Hehe, of course, it's all true!
5	44921	Please see my post below.
5	44920	Thank you very much for your reply.
8	48675	HTML is not a subset of XML!

A.User ID	A.Reputation	A.Location	B.User ID	B.Post ID	B.Text
3	3738	New York, NY	3	35314	Not sure why this is getting downvoted.
3	3738	New York, NY	3	48002	Hehe, of course, it's all true!
4	12946	New York, NY	null	null	null
5	17556	San Diego, CA	5	44921	Please see my post below.
5	17556	San Diego, CA	5	44920	Thank you very much for your reply.
9	3443	Oakland, CA	null	null	null

Left Outer Join A + B sur la clé ID: Tous les éléments qui ont la même valeur de ID sont fusionnés ce de A sont gardé avec des champs null dans les colonnes de B

Right outer join

User ID	Reputation	Location
3	3738	New York, NY
4	12946	New York, NY
5	17556	San Diego, CA
9	3443	Oakland, CA

User ID	Post ID	Text
3	35314	Not sure why this is getting downvoted.
3	48002	Hehe, of course, it's all true!
5	44921	Please see my post below.
5	44920	Thank you very much for your reply.
8	48675	HTML is not a subset of XML!

A.User ID	A.Reputation	A.Location	B.User ID	B.Post ID	B.Text
3	3738	New York, NY	3	35314	Not sure why this is getting downvoted.
3	3738	New York, NY	3	48002	Hehe, of course, it's all true!
5	17556	San Diego, CA	5	44921	Please see my post below.
5	17556	San Diego, CA	5	44920	Thank you very much for your reply.
null	null	null	8	48675	HTML is not a subset of XML!

Right Outer Join A + B sur la clé ID: Tous les éléments qui ont la même valeur de ID sont fusionnés ce de B sont gardé avec des champs null dans les colonnes de A

Full outer join

User ID	Reputation	Location
3	3738	New York, NY
4	12946	New York, NY
5	17556	San Diego, CA
9	3443	Oakland, CA

User ID	Post ID	Text
3	35314	Not sure why this is getting downvoted.
3	48002	Hehe, of course, it's all true!
5	44921	Please see my post below.
5	44920	Thank you very much for your reply.
8	48675	HTML is not a subset of XML!

A.User ID	A.Reputation	A.Location	B.User ID	B.Post ID	B.Text
3	3738	New York, NY	3	35314	Not sure why this is getting downvoted.
3	3738	New York, NY	3	48002	Hehe, of course, it's all true!
4	12946	New York, NY	null	null	null
5	17556	San Diego, CA	5	44921	Please see my post below.
5	17556	San Diego, CA	5	44920	Thank you very much for your reply.
null	null	null	8	48675	HTML is not a subset of XML!
9	3443	Oakland, CA	null	null	null

Full outer Join A + B sur la clé ID: Tous les éléments qui ont la même valeur de ID sont fusionnés ce de A et B sont gardé avec des champs null dans les colonnes de B ou B

Anti join

User ID	Reputation	Location
3	3738	New York, NY
4	12946	New York, NY
5	17556	San Diego, CA
9	3443	Oakland, CA

User ID	Post ID	Text
3	35314	Not sure why this is getting downvoted.
3	48002	Hehe, of course, it's all true!
5	44921	Please see my post below.
5	44920	Thank you very much for your reply.
8	48675	HTML is not a subset of XML!

A.User ID	A.Reputation	A.Location	B.User ID	B.Post ID	B.Text
4	12946	New York, NY	null	null	null
null	null	null	8	48675	HTML is not a subset of XML!
9	3443	Oakland, CA	null	null	null

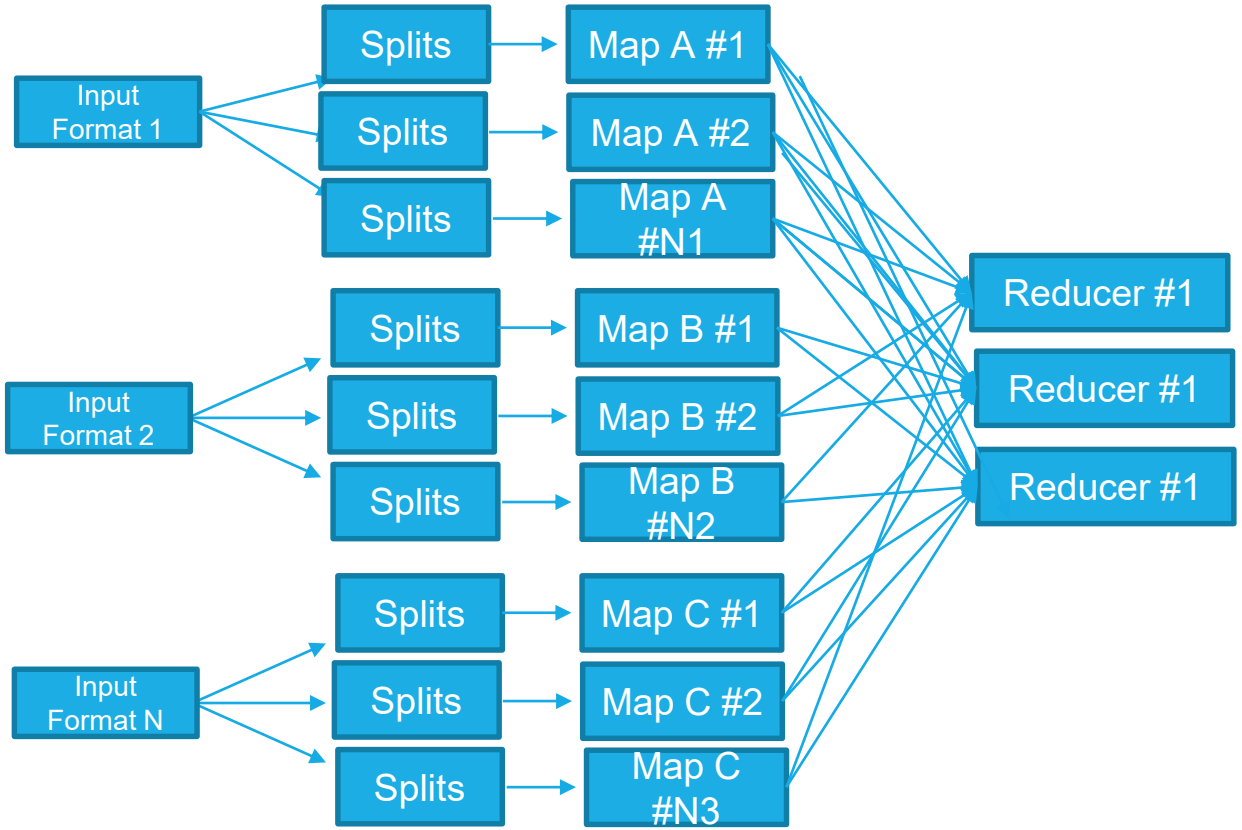
Cartesian product

User ID	Reputation	Location
3	3738	New York, NY
4	12946	New York, NY
5	17556	San Diego, CA
9	3443	Oakland, CA

User ID	Post ID	Text
3	35314	Not sure why this is getting downvoted.
3	48002	Hehe, of course, it's all true!
5	44921	Please see my post below.
5	44920	Thank you very much for your reply.
8	48675	HTML is not a subset of XML!

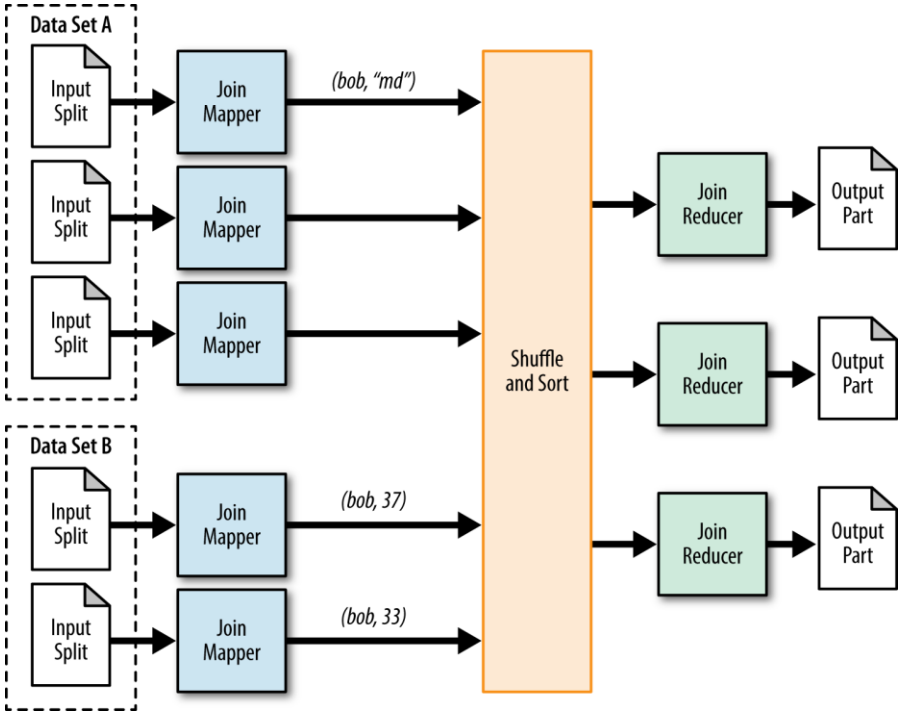
A.User ID	A.Reputation	A.Location	B.User ID	B.Post ID	B.Text
3	3738	New York, NY	3	35314	Not sure why this is getting downvoted.
3	3738	New York, NY	3	48002	Hehe, of course, it's all true!
3	3738	New York, NY	5	44921	Please see my post below.
3	3738	New York, NY	5	44920	Thank you very much for your reply.
3	3738	New York, NY	8	48675	HTML is not a subset of XML!
4	12946	New York, NY	3	35314	Not sure why this is getting downvoted.
4	12946	New York, NY	3	48002	Hehe, of course, it's all true!
4	12946	New York, NY	5	44921	Please see my post below.
4	12946	New York, NY	5	44920	Thank you very much for your reply.
4	12946	New York, NY	8	48675	HTML is not a subset of XML!
5	17556	San Diego, CA	3	35314	Not sure why this is getting downvoted.
5	17556	San Diego, CA	3	48002	Hehe, of course, it's all true!
5	17556	San Diego, CA	5	44921	Please see my post below.
5	17556	San Diego, CA	5	44920	Thank you very much for your reply.
5	17556	San Diego, CA	8	48675	HTML is not a subset of XML!
9	3443	Oakland, CA	3	35314	Not sure why this is getting downvoted.
9	3443	Oakland, CA	3	48002	Hehe, of course, it's all true!
9	3443	Oakland, CA	5	44921	Please see my post below.
9	3443	Oakland, CA	5	44920	Thank you very much for your reply.
9	3443	Oakland, CA	8	48675	HTML is not a subset of XML!

Multiple InputFormat




```
public static void main(String[] args) throws Exception {
Configuration conf = getConf();
Job job = Job.getInstance(conf, "MultipleInputExemple");
job.setJarByClass(PostCommentBuildingDriver.class);
MultipleInputs.addInputPath(job,
                new Path(args[0]), InputFormat1.class, MapperA.class);
MultipleInputs.addInputPath(job,
                new Path(args[1]), InputFormat2.class, MapperB.class);
MultipleInputs.addInputPath(job,
                new Path(args[N]), InputFormatN.class, MapperN.class);
job.setReducerClass(Reducer.class)
job.setOutputFormatClass(OutputFormat.class);
TextOutputFormat.setOutputPath(job, new Path(args[N+1]));
job.setOutputKeyClass(Text.class);
job.setOutputValueClass(Text.class);
System.exit(job.waitForCompletion(true) ? 0 : 2);
}
```

Pattern de jointure : Reduce side Join



Le principe du Reduce side join est de se servir d'un MultipleInput pour lire simultanément plusieurs fichiers et utiliser le mécanisme de distribution de message pour regrouper les entrée qui partage une même clé. La jointure est ensuite faite complètement dans le reducer. Toutes les données sont donc transféré sur les reducer.

Pattern de jointure : Reduce side Join

```
public static class MapperA extends Mapper<Object, Text, Text, Text> {
private Text outkey = new Text();
private Text outvalue = new Text();
public void map(Object key, Text value, Context context)vthrows
IOException, InterruptedException {
outkey.set(GETKEY(value));
outvalue.set("A" + GETVALUE(value));
context.write(outkey, outvalue);
}
}
public static class MapperB extends Mapper<Object, Text, Text, Text> {
private Text outkey = new Text();
private Text outvalue = new Text();
public void map(Object key, Text value, Context context)vthrows
IOException, InterruptedException {
outkey.set(GETKEY(value));
outvalue.set("B" + GETVALUE(value));
context.write(outkey, outvalue);
}
}
```

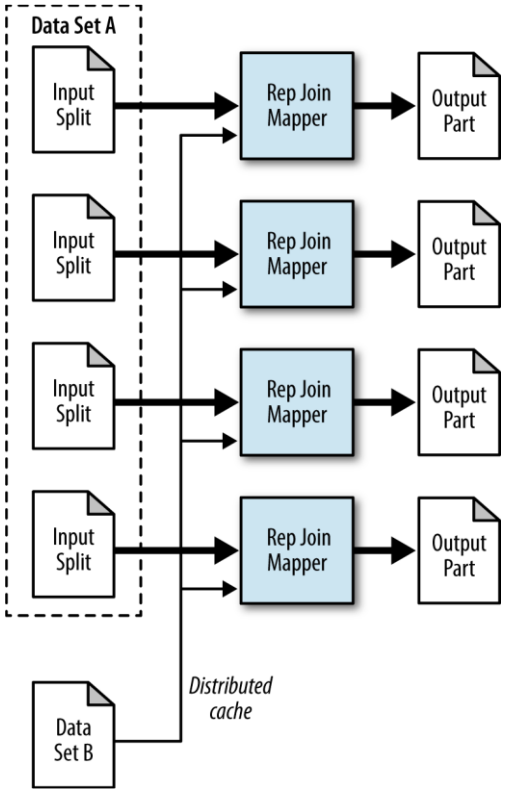
Pattern de jointure : Reduce side Join

```
if (joinType.equalsIgnoreCase("inner")) {
// If both lists are not empty, join A with B
if (!listA.isEmpty() && !listB.isEmpty()) {
for (Text A : listA) {
 for (Text B : listB) {
   context.write(A, B);
}
```

Pattern de jointure : Reduce side Join

```
if (joinType.equalsIgnoreCase("leftouter")) {
// For each entry in A,
for (Text A : listA) {
// If list B is not empty, join A and B
if (!listB.isEmpty()) {
for (Text B : listB) {
context.write(A, B);
}
} else {
// Else, output A by itself
context.write(A, EMPTY_TEXT);
}
```

Pattern de jointure : Replicated Join



Le principe du replicated join est d'effectuer la jointure uniquement dans les mappers. Pour cela il faut impérativement que toutes les tables à « joindre » sauf une rentre en mémoire des mappeurs. Cette technique ne s'applique qu'à l'inner join ou au left outer join.

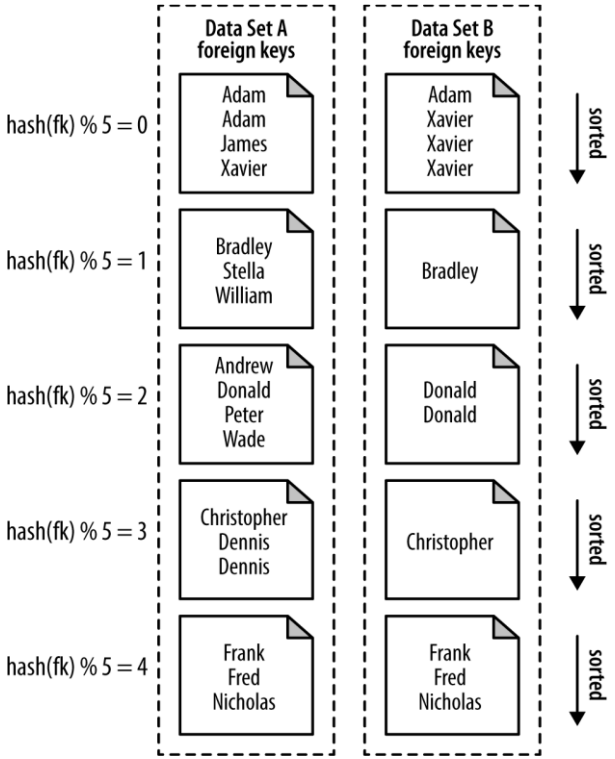
Pattern de jointure : Replicated Join

```
public static class ReplicatedJoinMapper extends Mapper<Object, Text, Text, Text> {
  private HashMap<String, String> userIdToInfo = new HashMap<String, String>();
  private String joinType = null;
  public void setup(Context context) throws IOException,InterruptedException {
   Path[] files = DistributedCache.getLocalCacheFiles(context.getConfiguration());
   // Read all files in the DistributedCache
   for (Path p : files) {
     BufferedReader rdr = new BufferedReader(new InputStreamReader(new GZIPInputStream(new
FileInputStream(new File(p.toString()))))));
     String line = null;
     // For each record in the user file
    while ((line = rdr.readLine()) != null) {
     // Get the user ID for this record
     // split the line line.split ….
     String userId = parsed.get("Id");
     // Map the user ID to the record
     userIdToInfo.put(userId, line); //ATTENTION DANS CE CAS CLE UNIQUE
   }
}
```

Pattern de jointure : Replicated Join

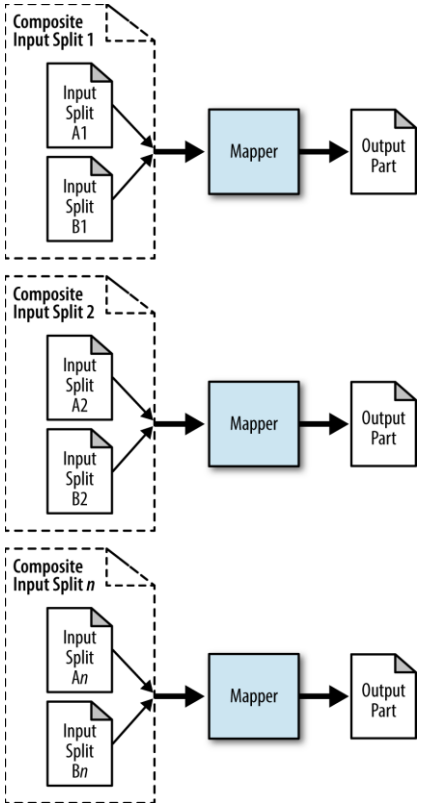
```
public void map(Object key, Text value, Context context) throws IOException,
InterruptedException {
 parsed = split the line …
 String userId = parsed.get("UserId");
 String userInformation = userIdToInfo.get(userId);
 // If the user information is not null, then output
 if (userInformation != null) {
  outvalue.set(userInformation);
  context.write(value, outvalue);
 }
 else if (joinType.equalsIgnoreCase("leftouter")) {
  context.write(value, new Text());
 }
```

Pattern de jointure: Composite Join



Le principe du composite join est de préparer les données avant le split. Chaque « table » doit être triée et découpé en un même nombre de partitions. Ce pattern permet de faire des Inner join et des full join très efficace.

Pattern de jointure: Composite Join



On utilise ensuite un CompositeInputFormat pour faire la jointure. Le composite input format fait le merge des deux inputs splits et renvoie les record reader renvoie un tuple au mapper.

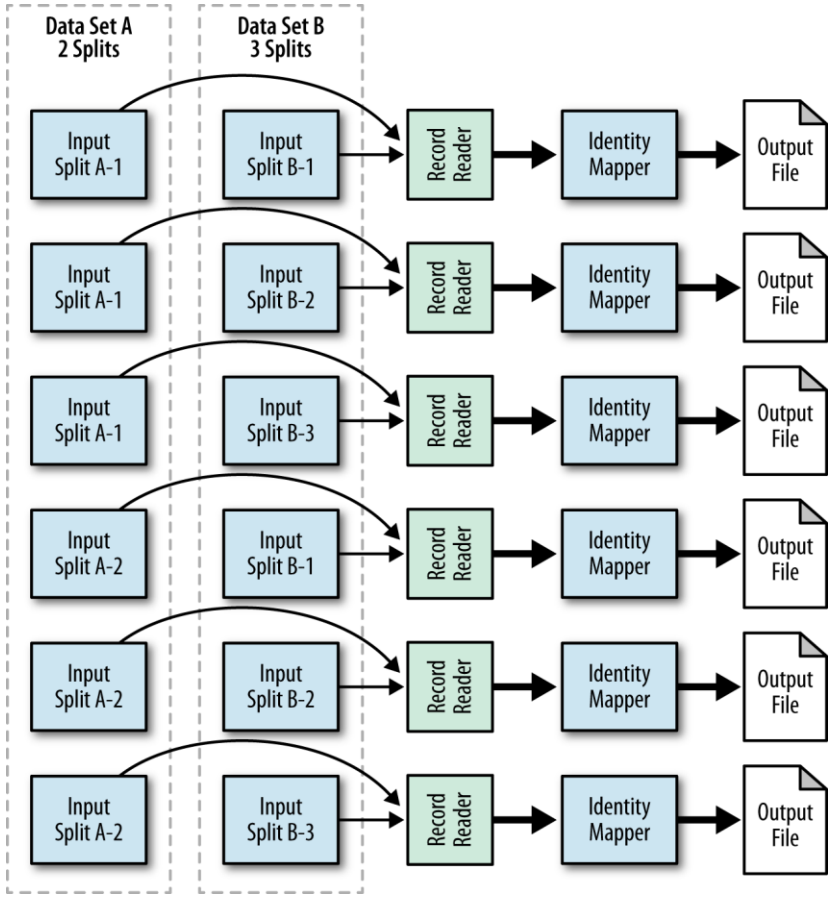
Pattern de jointure: Composite Join

```
public static void main(String[] args) throws Exception {
Path userPath = new Path(args[0]);
Path commentPath = new Path(args[1]);
Path outputDir = new Path(args[2]);
String joinType = args[3];
JobConf conf = new JobConf("CompositeJoin");
conf.setJarByClass(CompositeJoinDriver.class);
conf.setMapperClass(CompositeMapper.class);
conf.setNumReduceTasks(0);
conf.setInputFormat(CompositeInputFormat.class);
conf.set("mapred.join.expr", CompositeInputFormat.compose(joinType,
KeyValueTextInputFormat.class, userPath, commentPath));
TextOutputFormat.setOutputPath(conf, outputDir);
conf.setOutputKeyClass(Text.class);
conf.setOutputValueClass(Text.class);
RunningJob job = JobClient.runJob(conf);
while (!job.isComplete()) {
Thread.sleep(1000);
}
System.exit(job.isSuccessful() ? 0 : 1);
}
```

Pattern de jointure: Composite Join

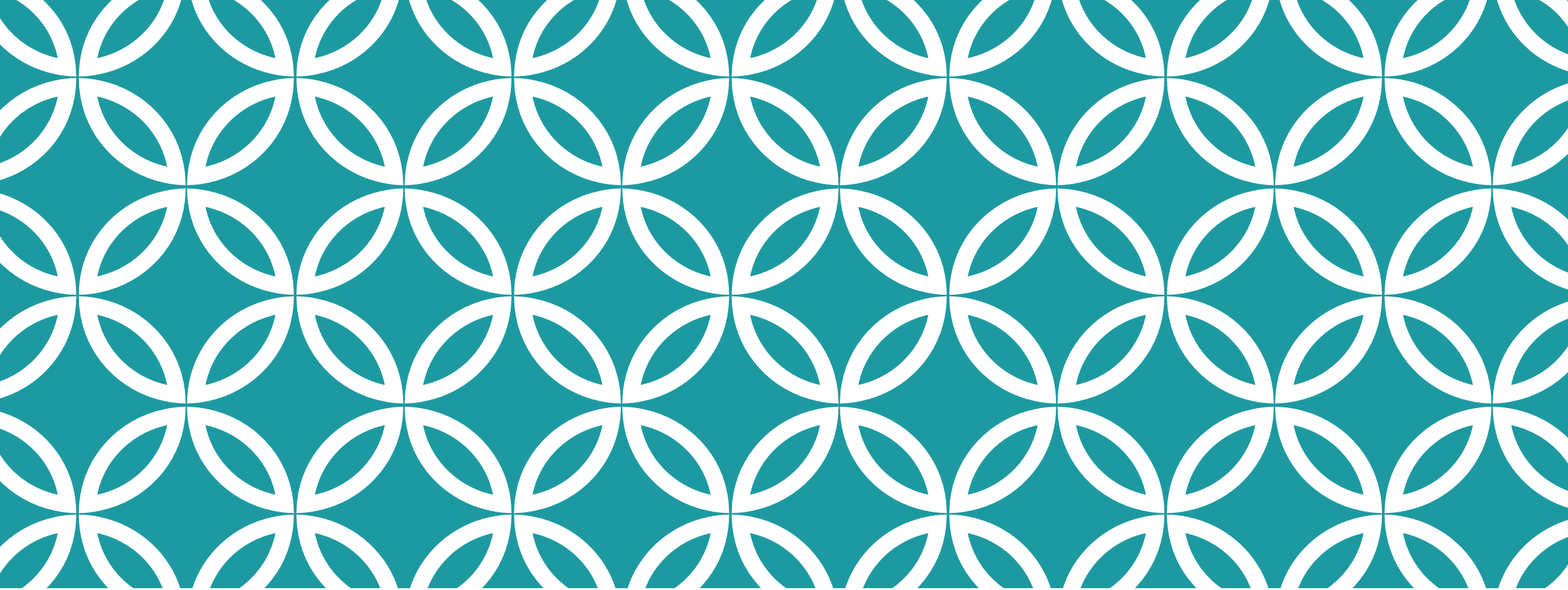
```
public static class CompositeMapper extends MapReduceBase implements
Mapper<Text, TupleWritable, Text, Text> {
public void map(Text key, TupleWritable value,
OutputCollector<Text, Text> output,
Reporter reporter) throws IOException {
// Get the first two elements in the tuple and output them
output.collect((Text) value.get(0), (Text) value.get(1));
}
}
```

Pattern de jointure: Cartesian Product



Pattern de jointure: Cartesian Product

```
public static class CartesianInputFormat extends FileInputFormat {
public static final String LEFT_INPUT_FORMAT = "cart.left.inputformat";
public static final String LEFT_INPUT_PATH = "cart.left.path";
public static final String RIGHT_INPUT_FORMAT = "cart.right.inputformat";
public static final String RIGHT_INPUT_PATH = "cart.right.path";
public static void setLeftInputInfo(JobConf job,
Class<? extends FileInputFormat> inputFormat, String inputPath) {
job.set(LEFT_INPUT_FORMAT, inputFormat.getCanonicalName());
job.set(LEFT_INPUT_PATH, inputPath);
}
public static void setRightInputInfo(JobConf job,
Class<? extends FileInputFormat> inputFormat, String inputPath) {
job.set(RIGHT_INPUT_FORMAT, inputFormat.getCanonicalName());
job.set(RIGHT_INPUT_PATH, inputPath);
}
public InputSplit[] getSplits(JobConf conf, int numSplits)
throws IOException {
// Get the input splits from both the left and right data sets
InputSplit[] leftSplits = getInputSplits(conf,
conf.get(LEFT_INPUT_FORMAT), conf.get(LEFT_INPUT_PATH),
numSplits);
InputSplit[] rightSplits = getInputSplits(conf,
conf.get(RIGHT_INPUT_FORMAT), conf.get(RIGHT_INPUT_PATH),
numSplits);
// Create our CompositeInputSplits, size equal to
// left.length * right.length
CompositeInputSplit[] returnSplits =
new CompositeInputSplit[leftSplits.length *
rightSplits.length];
```



Pattern d’organisation

Pattern d'organisation: Le tri

Le design pattern de tri est un pattern difficile car on le tri est un processus global. L'implémentation du tri en Map Reduce consiste à utiliser deux étapes. Une première étape va échantillonner l'ensemble des valeurs à trier en n'en prenant qu'un nombre limité que l'on peut trier localement (par exemple 100K). On va utiliser ces 100K milles éléments triés pour construire une fonction de partionement qui équilibre la charge et qui garantit que l'on peut faire un tri local dans les reducers.

Pattern d'organisation: Le tri

Soit les 20 valeurs triées suivantes (tirées aléatoirement) :
2 4 5 9 10 12 15 16 17 18 22 23 23 35 43 65 78 96 99 100

Supposons que nous ayons que 4 reducers:
2 4 5 9 10 12 15 16 17 18 22 23 23 35 43 65 78 96 99 100

En choisissant la fonction de partionnement suivante :

R#1 < 12 <= R#2 < 22 <= R#3 < 65 <= R#4

Chaque reducer recevra un nombre « similaire » d'élément et tous les éléments qu'un reducer reçoit pourront être trié en parallèle. C'est avec cette technique que Yahoo a gagner le concours du Terasort.