Institute of Technology, Nirma University

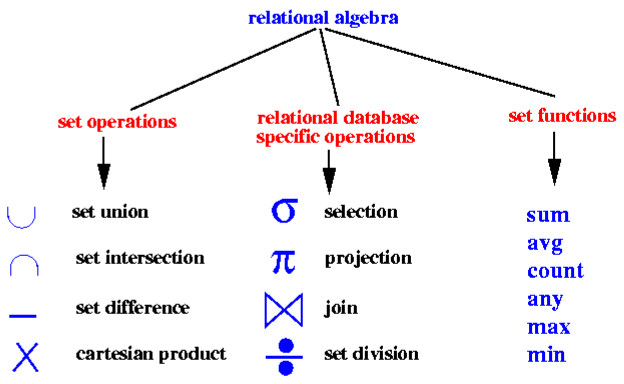
Big Data Analytics (2CS702)

Year 2022-23

Semester: 07

Assignment on,

Relational Algebra Operations: Union, Natural Join



Under The Guidance of,

Prof. Jigna Patel

Prepared by,

01	**Gaurav Sakariya**	**19BCE233**
02	**Yash Savani**	**19BCE243**
03	**Virang Virani**	**19BCE296**

Relational Operations Using MapReduce

We must first understand what a relation represents before obtaining a quick introduction to relational algebra. There is no use in giving a lengthy overview of SQL here because the most of us are already familiar with it. A database table is represented by a relation. The fact that duplicate rows are implicitly deleted in relational algebra but not in SQL implementations is a key difference between the two languages. The implementation of relational algebra operations is discussed in this article, but it can also be applied to implementations that don't remove duplicates. The reader can quickly read over this section if they are familiar with relational algebra.

- **Selection:** selection (WHERE clause in SQL) lets you apply a condition over the data you have and only get the rows that satisfy the condition.
- **Projection:** In order to select some columns only we use the projection operator. It’s analogous to SELECT in SQL.
- **Intersection:** Same as INTERSECT in SQL. It intersects two tables and selects only the common rows.
- **Union:** We concatenate two tables vertically. Similar to UNION in SQL, but the duplicate rows are removed implicitly.
- **Natural Join:** Merge two tables based on some common column. It represents the INNER JOIN in SQL. But the condition is implicit on the column that is common in both tables. The output will only contain rows for which the values in the common column matches.

Union Using Map Reduce

Both selection and projection are operations that are applied on a single table, whereas Union, intersection and difference are among the operations that are applied on two or more tables. Let’s consider that schemas of the two tables are the same, and columns are also ordered in same order.

Map Function: For each row r generate key-value pair (r, r).

Reduce Function: With each key there can be one or two values (As we don’t have duplicate rows), in either case just output first value.

This operation has the map function of the selection and reduce function of projection. Let’s see the working using an example. Here yellow colour represents one table and green colour is used to represent the other one stored at two map workers.

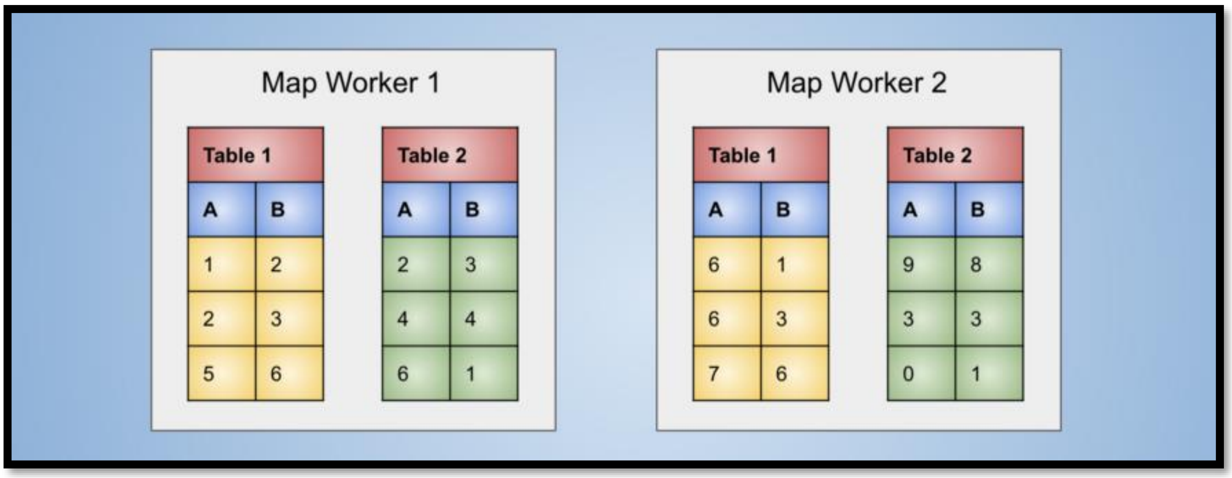

Map Worker 1

Table 1	
A	B
1	2
2	3
5	6

Table 2	
A	B
2	3
4	4
6	1

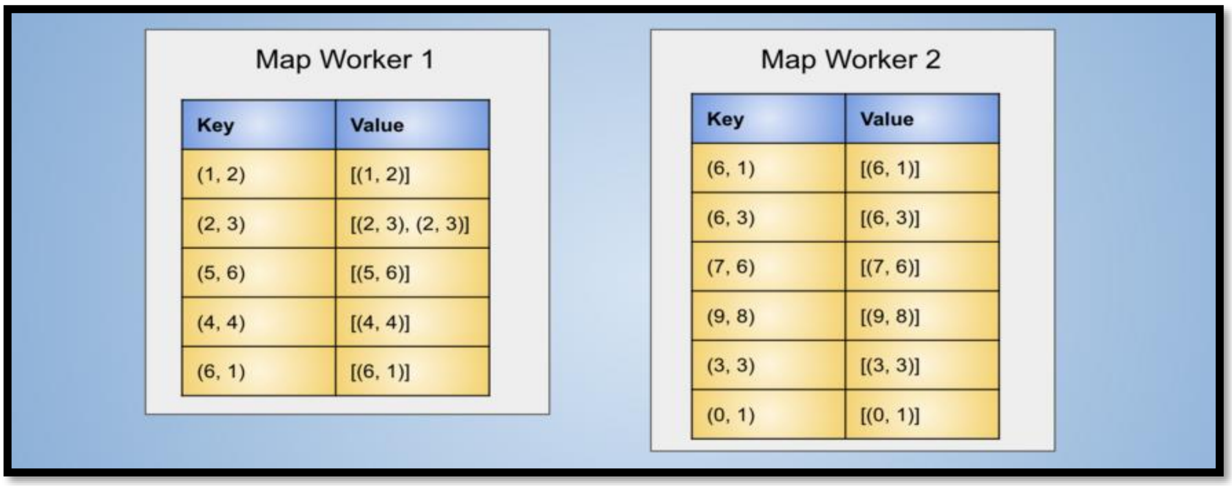
Map Worker 2

Table 1	
A	B
6	1
6	3
7	6

Table 2	
A	B
9	8
3	3
0	1

Initial data at map workers

After applying the map function and grouping the keys we will get output as:


Map Worker 1

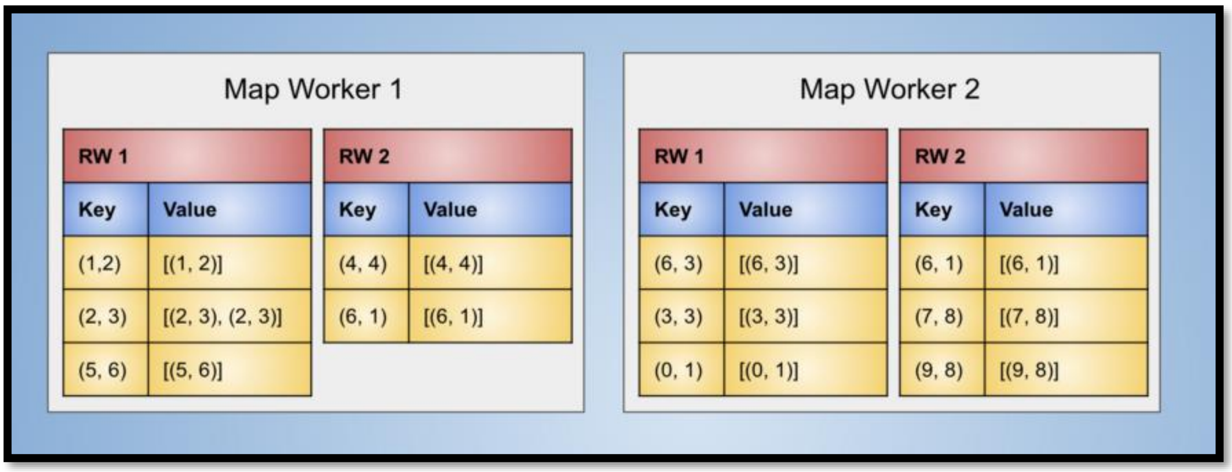
Key	Value
(1, 2)	[(1, 2)]
(2, 3)	[(2, 3), (2, 3)]
(5, 6)	[(5, 6)]
(4, 4)	[(4, 4)]
(6, 1)	[(6, 1)]

Map Worker 2

Key	Value
(6, 1)	[(6, 1)]
(6, 3)	[(6, 3)]
(7, 6)	[(7, 6)]
(9, 8)	[(9, 8)]
(3, 3)	[(3, 3)]
(0, 1)	[(0, 1)]

Map and grouping the keys

The data to be sent to reduce workers will look like:


Map Worker 1

RW 1	
Key	Value
(1,2)	[(1, 2)]
(2, 3)	[(2, 3), (2, 3)]
(5, 6)	[(5, 6)]

RW 2	
Key	Value
(4, 4)	[(4, 4)]
(6, 1)	[(6, 1)]

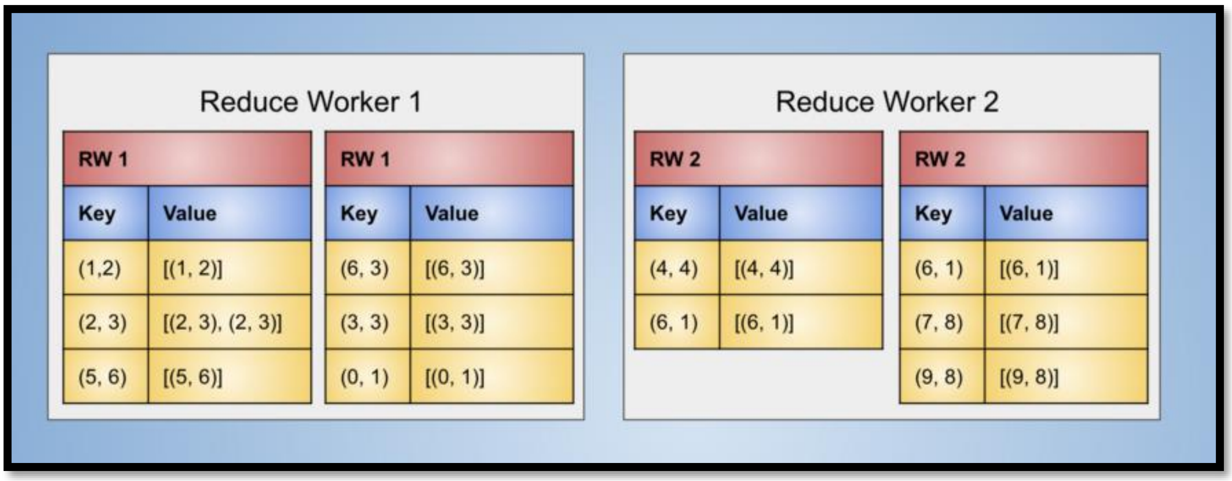
Map Worker 2

RW 1	
Key	Value
(6, 3)	[(6, 3)]
(3, 3)	[(3, 3)]
(0, 1)	[(0, 1)]

RW 2	
Key	Value
(6, 1)	[(6, 1)]
(7, 8)	[(7, 8)]
(9, 8)	[(9, 8)]

Files to be sent to reduce workers

Data at reduce workers after will be:



Reduce Worker 1

RW 1

Key	Value
(1,2)	[(1, 2)]
(2, 3)	[(2, 3), (2, 3)]
(5, 6)	[(5, 6)]

RW 1

Key	Value
(6, 3)	[(6, 3)]
(3, 3)	[(3, 3)]
(0, 1)	[(0, 1)]

Reduce Worker 2

RW 2

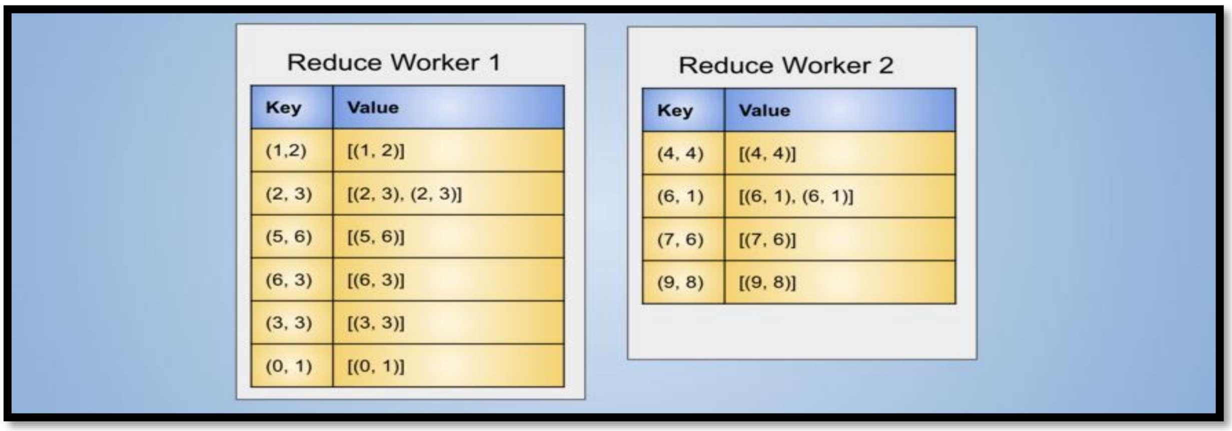
Key	Value
(4, 4)	[(4, 4)]
(6, 1)	[(6, 1)]

RW 2

Key	Value
(6, 1)	[(6, 1)]
(7, 8)	[(7, 8)]
(9, 8)	[(9, 8)]

Files At reduce workers

At reduce workers aggregation on keys will be done.



Reduce Worker 1

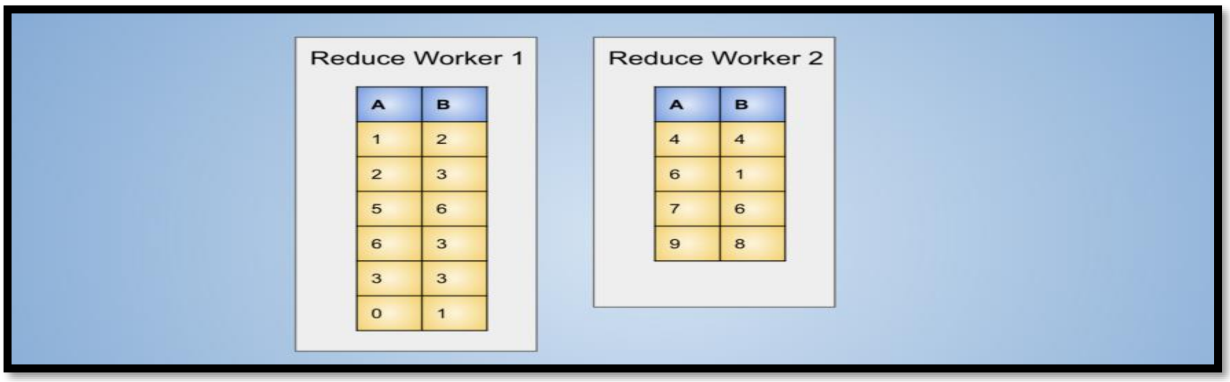
Key	Value
(1,2)	[(1, 2)]
(2, 3)	[(2, 3), (2, 3)]
(5, 6)	[(5, 6)]
(6, 3)	[(6, 3)]
(3, 3)	[(3, 3)]
(0, 1)	[(0, 1)]

Reduce Worker 2

Key	Value
(4, 4)	[(4, 4)]
(6, 1)	[(6, 1), (6, 1)]
(7, 6)	[(7, 6)]
(9, 8)	[(9, 8)]

Aggregated data at reduce workers

The final output after applying the reduce function which takes only the first value and ignores everything else is as follows:



Reduce Worker 1

A	B
1	2
2	3
5	6
6	3
3	3
0	1

Reduce Worker 2

A	B
4	4
6	1
7	6
9	8

Final table after union

Here we note that in this case same as projection we can this done without moving data around in case, we are not interested in removing duplicates. And hence this operation is also efficient its terms of data shuffle across machines.

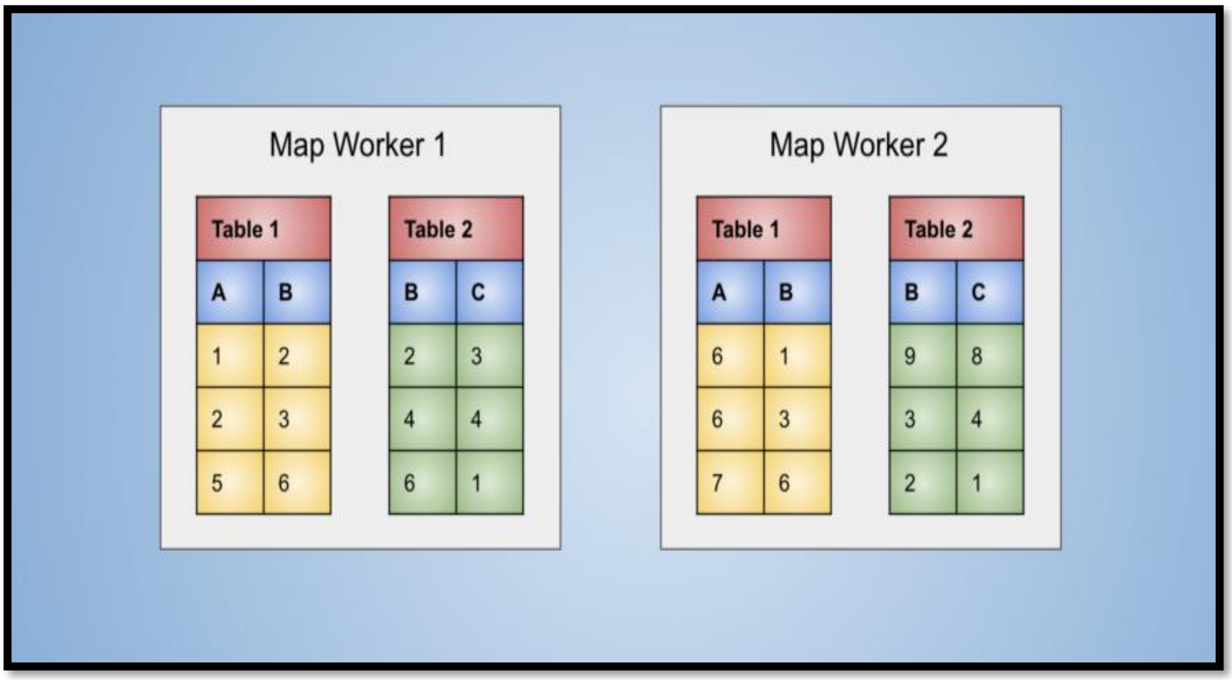
Natural Join Using Map Reduce

The natural join will keep the rows that matches the values in the common column for both tables. To perform natural join we will have to keep track of from which table the value came from. If the values for the same key are from different tables we need to form pairs of those values along with key to get a single row of the output. Join can explode the number of rows as we have to form each and every possible combination of the values for both tables.

Map Function: For two relations Table 1(A, B) and Table 2(B, C) the map function will create key-value pairs of form b: [(T1, a)] for table 1 where T1 represents the fact that the value a came from table 1, for table 2 key-value pairs will be of the form b: [(T2, c)].

Reduce Function: For a given key b construct all possible combinations for the values where one value is from table T1 and the other value is from table T2. The output will consist of key-value pairs of form b: [(a, c)] which represent one row a, b, c for the output table.

For an example let's consider joining Table 1 and Table 2, where B is the common column.



Map Worker 1

Table 1	
A	B
1	2
2	3
5	6

Table 2	
B	C
2	3
4	4
6	1

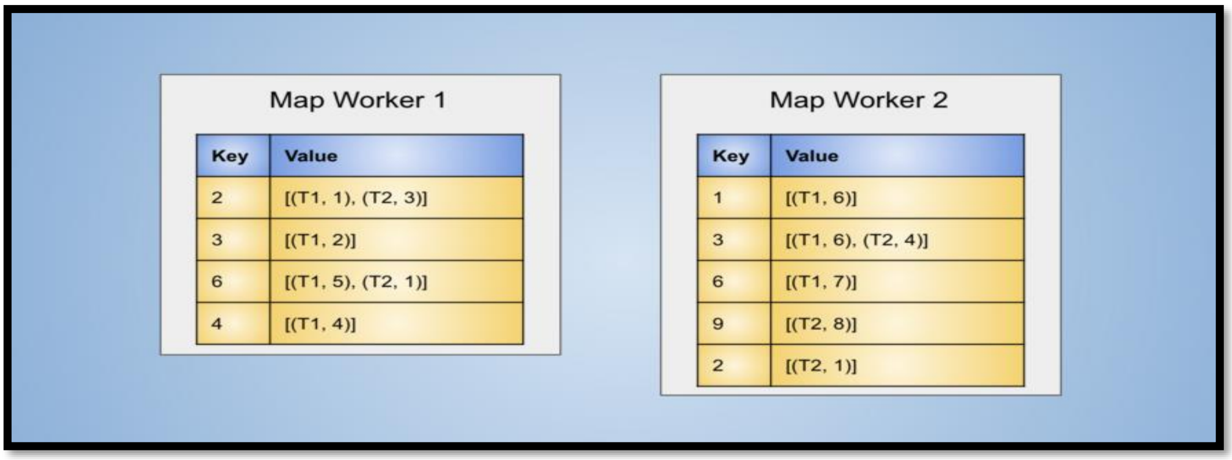
Map Worker 2

Table 1	
A	B
6	1
6	3
7	6

Table 2	
B	C
9	8
3	4
2	1

Initial data at map workers

The data after applying the map function and grouping at the map workers will look like:



Map Worker 1

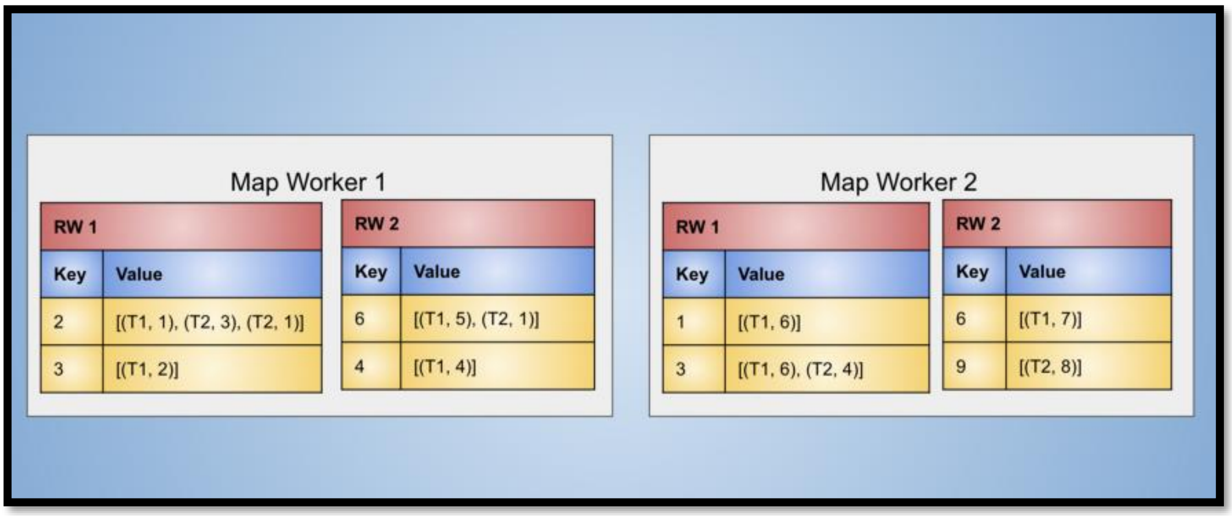
Key	Value
2	[(T1, 1), (T2, 3)]
3	[(T1, 2)]
6	[(T1, 5), (T2, 1)]
4	[(T1, 4)]

Map Worker 2

Key	Value
1	[(T1, 6)]
3	[(T1, 6), (T2, 4)]
6	[(T1, 7)]
9	[(T2, 8)]
2	[(T2, 1)]

Data at map workers after applying map function and grouping the keys

As has been the case so far files for reduce workers will be created at the map workers



Map Worker 1

RW 1

Key	Value
2	[(T1, 1), (T2, 3), (T2, 1)]
3	[(T1, 2)]

RW 2

Key	Value
6	[(T1, 5), (T2, 1)]
4	[(T1, 4)]

Map Worker 2

RW 1

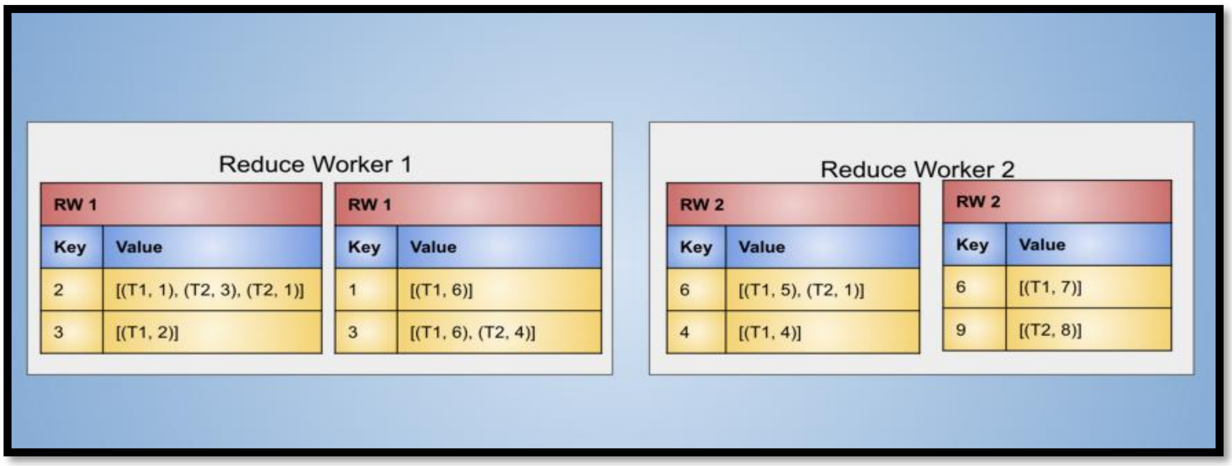
Key	Value
1	[(T1, 6)]
3	[(T1, 6), (T2, 4)]

RW 2

Key	Value
6	[(T1, 7)]
9	[(T2, 8)]

Files constructed for reduce workers

The data at the reduce workers will be:



Reduce Worker 1

RW 1

Key	Value
2	[(T1, 1), (T2, 3), (T2, 1)]
3	[(T1, 2)]

RW 1

Key	Value
1	[(T1, 6)]
3	[(T1, 6), (T2, 4)]

Reduce Worker 2

RW 2

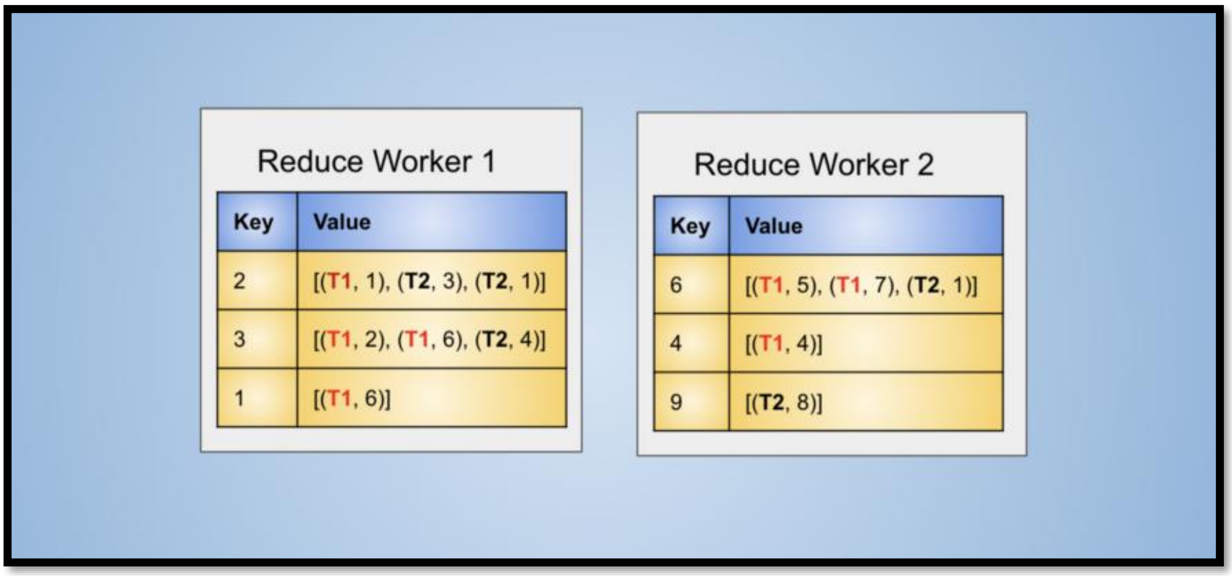
Key	Value
6	[(T1, 5), (T2, 1)]
4	[(T1, 4)]

RW 2

Key	Value
6	[(T1, 7)]
9	[(T2, 8)]

Data at reduce workers

Applying aggregation of keys at the reduce workers we get:



Reduce Worker 1

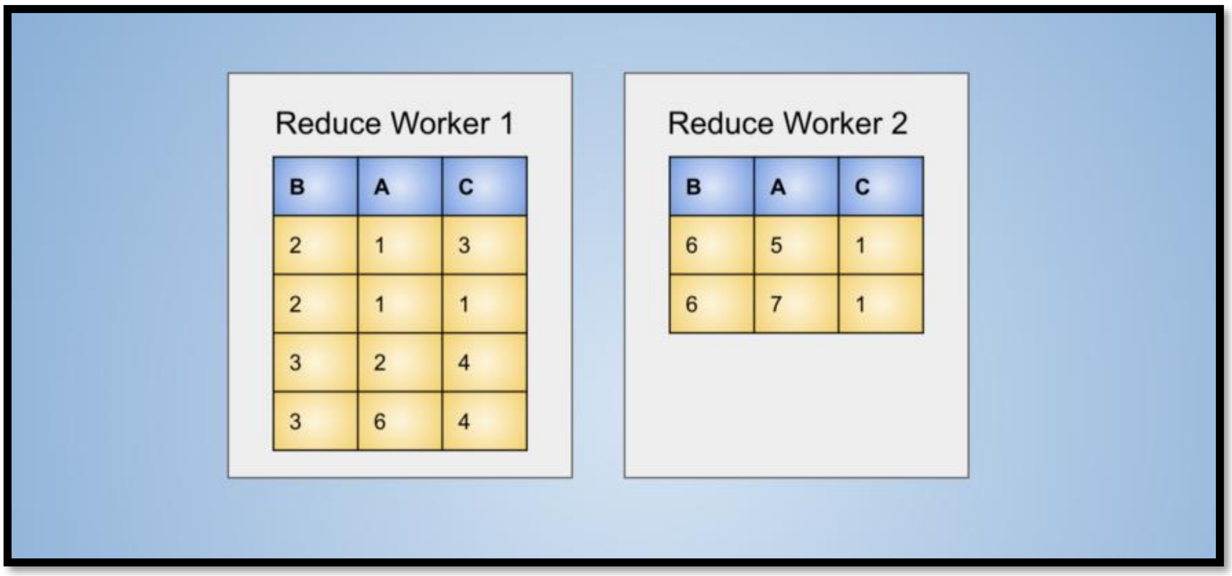
Key	Value
2	[(T1, 1), (T2, 3), (T2, 1)]
3	[(T1, 2), (T1, 6), (T2, 4)]
1	[(T1, 6)]

Reduce Worker 2

Key	Value
6	[(T1, 5), (T1, 7), (T2, 1)]
4	[(T1, 4)]
9	[(T2, 8)]

Data after aggregation of keys at the reduce workers

After applying the reduce function which will create a row by taking one value from table T1 and other one from T2. If there are only values from T1 or T2 in the values list that won't constitute a row in output.



Reduce Worker 1

B	A	C
2	1	3
2	1	1
3	2	4
3	6	4

Reduce Worker 2

B	A	C
6	5	1
6	7	1

Output of the join

As we need to keep context from which table a value came from, we can't get rid of the data that needs to be sent across the workers for application of reduce task, this operation also becomes costly as compared to others we discussed so far. The fact that for each list of values we need to create pairs also plays a major factor in the computation cost associated with this operation.