Three-Way Triangle Joins Algorithms

1. Computing the three-way triangle joins in one map-reduce round.

The Map Function:

As we know, the attributes in the schemas of R, S and T are fetched as keys to the map function and the name of the schemas and is fetched as the value. That is, ((x,b,y),(R,a)), ((x,y,c),(S,b)) and ((a,x,y),(T,c)) is the key-value pair. In this program, we use 0 indicates the relation is R ,1 indicates the relation is S and 2 indicates the relation is T.

Depending on the project requirement, we have at most 128 reducers available. So we need to use hash values of entire R(A,B),S(B,C) and T(C,A). Choose a hash function h that is each a,b,c divides 5 that maps A-,B- and C-values to 5 buckets. So there will be 125 Reduce processes, one for each (A-bucket, B-bucket, C-bucket) pair.

The Reduce Function:

Each key value (a,b),(b,c),(c,a)will be associated with a list of pairs that are either of the form (R,a) or (S,a). Grouping together all the values associated with the same key value (a,b,c).

Restore the value of key (a,b,c) and in each reducer if the value of (a,b,c) is same then output from the value of key.

1. Computing the result using a simple cascade of two-way joins in two rounds.

A simple cascade of two-way joins need to do twice map reduce.

For three-way triangle join we can join R and S first or S and T first or T and R first and the MapReduce algorithm is like the two-way join in project one. However, we want to minimize the communication cost, so we need to compare the size of R, S and T. Suppose that the relation R,S and T have sizes r, s and t, if (r⋈s) less than (s⋈t), we join R nad S first and create the intermediate join R ⋈ S. In the second round, we just need to join the rest of relation use the algorithm like two-way join in project one.