GENERAL FRAMEWORK FOR OPTIMIZATION OF DISTRIBUTED QUERIES

Dr. Sunita M. Mahajan[1] and Ms.Vaishali P. Jadhav[2]

[1]Mumbai Education Trust, Reclaimation,Bandra, Mumbai ,India
sunitamm@gmail.com
[2]NMIMS University, Mumbai, India
vaishaliwadghare@gmail.com

ABSTRACT

The paper suggests the heuristic approach for selecting the optimal evaluation plan and Semi-join approach for reducing the communication cost. Calculating the cost of each evaluation plan of a query takes lots of computational efforts as well as time. Initially the heuristic approach is used to find best evaluation plan among various plans of a single query. By considering only joins, queries are divided into two: tree and cyclic. Semi-join is a useful tool to reduce the cost of joins. The communication cost in distributed database is mainly depending upon the amount of data transferred from one site to another site. Semi-join approach is sufficient to fully reduce the tree queries while semi-joins are not the full reducers for cyclic queries. Cyclic queries are therefore converted into tree queries using different approaches such as addition of vertices, elimination of vertices, decomposition of vertices and elimination of edges. After conversion of cyclic to tree query, semi-join approach can be used.

KEYWORDS

Heuristics, Query Optimization, Cost-Based Optimization, Tree query, Cyclic query

1. INTRODUCTION

A heuristic is a rule of thumb, or a good guide to follow when making decisions. Heuristic refers specifically to algorithms in computer science. Algorithms have importance which works well in certain situations even though they may perform inefficiently for uncommon situations.
Therefore, a heuristic is a problem solving technique may include running tests and getting results by trial and error. After testing more sample data, it becomes easier to create an efficient algorithm to process similar types of data. These algorithms are not always perfect, but work well most of the time. The goal of heuristics is to develop a simple process that generates accurate results in an acceptable amount of time.

In case of Query Optimization, it is impractical to search evaluation plans exhaustively, when the optimization of joins involves many relations. Optimizers use heuristics to find best evaluation plan in addition with an exhaustive algorithm which generates alternative join order plans and terminates when cost budget is exceed [1-3].

Our framework includes the heuristic rules for finding optimal evaluation plan among various plans of a single query. This reduces the computational cost of each evaluation plan. Then for distributed query, semi-join approach is used. The queries which are not forming cycles in the query graph are called tree queries. Semi-join approach reduces the communication cost by reducing the amount of data transfer from one site to another site. If the query graph forms the

DOI: 10.5121/ijdms.2012.4303

cycle, then query is a cyclic query. To make the use of semi-join approach, cyclic queries are converted into tree queries by using different approaches such as addition or elimination or decomposition of vertices, elimination of edges etc.

The approach of using semi-joins as reducers to process a distributed query has received a great deal of attention. This approach consists of the following three phases: (1) a local processing phase which involves all local processing such as selections and projections, (2) a semi-join reduction phase where a sequence of semi-joins is used to reduce the size of relations, and then, lessen the total communication cost required, and (3) a final processing phase in which all resulting relations are sent to the result site where the final query processing is performed. The problem of minimizing the communication cost for distributed query processing by a semi-join reduction approach has been shown as NP-hard.

The rest of the paper is organized as follows: Section 2 describes the related work. Section 3 describes the heuristic approach. Section 4 gives the tree query optimization. Section 5 describes the cyclic query optimization. Section 6 gives the concluding remarks.

2. RELATED WORK

There has been much work on distributed query processing and optimization. Most of the work has focused on minimizing the total communication cost for executing a single query by judiciously choosing the join order and possibly adding semi-join operators to the query plan [1-4].

Hevner and Yao considered simple queries and suggested an optimal semi-join algorithm. They also extended their algorithm to produce efficient reducers for general queries [5].

Kambayashi Y. and Yoshikawa M. suggested horizontal decomposition as the one of the way of converting cyclic query into tree query [6][7].

Bernstein and Chiu showed that semi-joins are full reducers for tree queries. Tree queries are those queries whose query graphs are trees. They also give a proof that cyclic queries cannot be answered by the semi-joins alone [8][9].

Goodmann N. And Shemuli O. give different approaches of transforming cyclic query into tree query [10][11].

A survey article by Yu and Chang reviews many query processing algorithms. These algorithms can be characterised by several properties including timing of the optimization, replication of data, cost of the data etc [18-20][22].

System R* uses dynamic programming to find an optimal solution to the distributed query processing problem. The algorithm considers both processing and communication costs to arrive at an optimal strategy. However it does not use semi-joins as one of the options for reducing the relations [21].

Chen and Yu give different approach of semi-joins. They are showing that the approach of semi-join as reducers can lead to substantially larger reduction on data transmission required. They developed an efficient heuristic approach to determine an effective sequence of semi-join and join reducers. Semi-joins whose execution will reduce the amount of data transmission required to perform a join sequence are termed beneficial semi-joins for that join sequence.

Beneficial semi-joins include the conventional profitable semi-joins and the gainful semi-joins that are not profitable themselves but become beneficial due to the inclusion of join reducer. This type of dependency between semi-join and join reducers complicates the identification of beneficial semi-joins and the ordering in the reducer sequence.

They first obtain a sequence of join reducers and map it into a join sequence tree. In light of the join sequence tree we derive important properties of beneficial semi-joins. These properties are then applied to develop an efficient algorithm to determine the beneficial semi-joins which can be inserted into the join sequence.

The approach of interleaving a join sequence with beneficial semi-joins is not only efficient but also effective in reducing the total amount of data transmission required to process distributed queries [23]

3. HEURISTIC APPROACH

There are many rules for transforming relational algebra operations into equivalent ones. In relational algebra, there are several definitions and theorems the query optimizer can use to transform the query. For instance, the definition of equivalent relations states that the set of attributes (domain) of each relation must be the same— because they are sets, the order does not matter. Here is a partial list of relational algebra theorems [1-2].

3.1. General Transformation Rules for Relational Algebra Operations

1. **Cascade of σ**: A conjunctive selection condition can be broken up into a cascade (sequence) of individual σ operations:

   $$\sigma_{c1 \text{ AND } c2 \text{ AND } \ldots \text{ AND } cn}(R) = \sigma_{c1}(\sigma_{c2}(\ldots(\sigma_{cn}(R))\ldots))$$

2. **Commutativity of σ**: The σ operation is commutative:

   $$\sigma_{c1}(\sigma_{c2}(R)) = \sigma_{c2}(\sigma_{c1}(R))$$

3. **Cascade of π:** In a cascade (sequence) of π operations, all but the last one can be ignored:

   $$\pi_{List1}(\pi_{List2}(\ldots(\pi_{Listn}(R))\ldots)) = \pi_{List1}(R)$$

4. **Commuting σ with π:** If the selection condition c involves only the attributes A1, ..., An in the projection list, the two operations can be commuted:

   $$\pi_{A1, A2, \ldots, An}(\sigma_{c}(R)) = \sigma_{c}(\pi_{A1, A2, \ldots, An}(R))$$

5. **Commutativity of ⋈(and x):** The operation is commutative as is the x operation:

   $$R \bowtie_C S = S \bowtie_C R; \quad R \times S = S \times R$$

6 **Commuting σ with ⋈ (or x):** If all the attributes in the selection condition c involve only the attributes of one of the relations being joined—say, R—the two operations can be commuted as follows:

a. $\sigma_c (R \bowtie S) = (\sigma_c (R)) \bowtie S$

Alternatively, if the selection condition c can be written as (c1 and c2), where condition c1 involves only the attributes of R and condition c2 involves only the attributes of S, the operations commute as follows:

b. $\sigma_c (R \bowtie S) = (\sigma_{c1} (R)) \bowtie (\sigma_{c2} (S))$

7. **Commuting π with ⋈ (or x):** Suppose that the projection list is L = {A1, ..., An, B1, ..., Bm}, where A1, ..., An are attributes of R and B1, ..., Bm are attributes of S. If the join condition c involves only attributes in L, the two operations can be commuted as follows:

a. $\pi_L (R \bowtie_C S) = (\pi_{A1, ..., An} (R)) \bowtie_C (\pi_{B1, ..., Bm} (S))$

b. If the join condition C contains additional attributes not in L, these must be added to the projection list, and a final π operation is needed.

8. **Commutativity of set operations:** The set operations U and ∩ are commutative but "–" is not.

9. **Associativity of ⋈ , x, U, and ∩:** These four operations are individually associative; that is, if q stands for any one of these four operations (throughout the expression), we have

a. $(R \; q \; S) \; q \; T = R \; q \; (S \; q \; T)$

10. **Commuting σ with set operations:** The σ operation commutes with U, ∩, and –. If q stands for any one of these three operations, we have

a. $\sigma_c (R \; q \; S) = (\sigma_c (R)) \; q \; (\sigma_c (S))$

11. **The π operation commutes with U:**

i. $\pi_L (R \; U \; S) = (\pi_L (R)) \; U \; (\pi_L (s))$

12. **Converting a (σ , X) sequence into ⋈:**

i. $\sigma_c (R \; X \; S) = (R \bowtie_C S)$

Figure 2: Outline of a Heuristic Algebraic Optimization Algorithm

The below algorithm utilizes some of the rules of transform an initial query tree into an optimized tree that is efficient to execute during heuristic optimization.

Input: Relational Algebra Query
Output: Best Evaluation Plan
Algorithm:
Step 1: Design the initial canonical tree of the query
Step 2: Move the SELECT operation down the query tree
Step 3: Apply more restrictive SELECT operation first. If the two relations are residing at same site, they will be handled first.
Step 4: Replace CARTESIAN PRODUCT and SELECT with JOIN operation
Step 5: Move PROJECT operations down the query tree

Figure 3: Heuristic Query Optimization

3.3. Motivating Example

Consider the following relations [4]
R1(Prof, Dept, Research_Area)
R2(Course, Professor, Text, Department)

Retrieve the professor who is teaching Soft Computing course which includes the text as Fuzzy in Computers department and whose research area is Neural Network.

Select Prof from R1,R2 WHERE Dept = Department AND Research_Area = "Neural Network" AND Course = "Soft Computing" AND Text ="Fuzzy" AND Department = "Computers" ;

To implement optimizer we have considered the Visual Studio 2008 and SQL Server 2008 R2.The above query in the relational algebra form is given as the input and the final query tree by using the heuristic rules is the output of the optimizer.

Following is the query tree converted using heuristic:

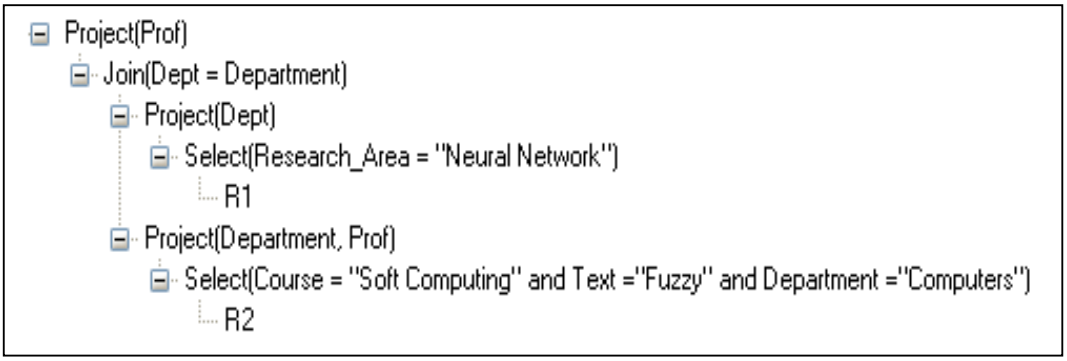

Figure 4: Heuristic Query Tree

4. COST ESTIMATION APPROACH

In case of distributed query, along with the heuristic rules we need to apply cost estimation methods for reducing the transmission cost and for better optimization. The cost estimation methods are different for different types of queries. By considering only joins, the distributed queries are divided into tree query and cyclic query.

4.1 Tree Query and Cyclic Query

Consider the join graph and query graph given below. The query graph which forms the cycle is the graph of cyclic query and the query graph which does not form the cycle is tree query graph. We will consider the optimization of both the queries [4][5][6].

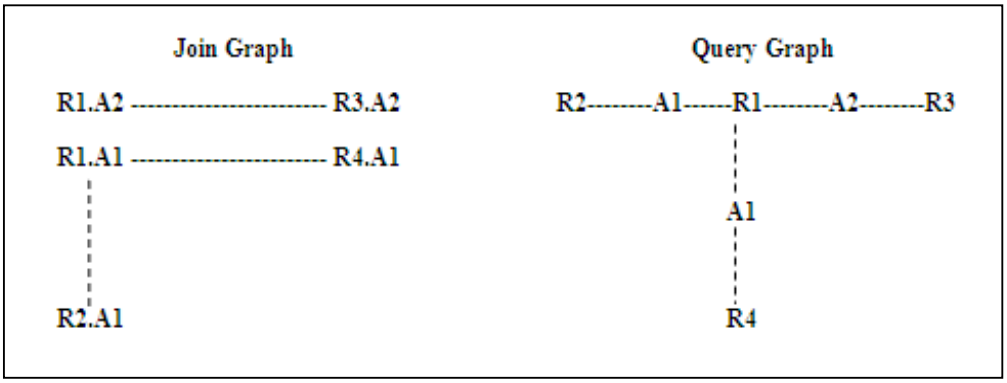

Figure 5: Tree Query Example

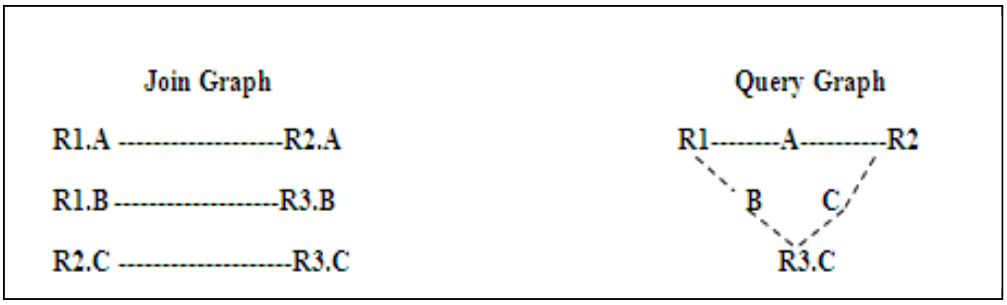

Figure 6: Cyclic Query Example

4.2 Tree Query Optimization

Since the amount of data transferred affects the cost of a strategy, we should attempt to reduce it. A promising approach for optimizing tree queries is to make use of semi-joins. A semi-join from relation R2 to relation R1 on attribute A is denoted by **R2---B--->R1** where R2 is the sending relation , R1 is the reduced relation and B is the joining attribute. It can be obtained by joining R1 and R2 on attribute B, then projecting the resulting relation on the schema of R1. The semi-join operation is denoted as $\mathbf{R}_{i+1} \ltimes \mathbf{R}_i$. For semi-join operation, we will consider the selectivity of relation on join attribute [7][8][9][10].

Selectivity: Let Pi be the probability that a value in attribute A appears in Ri, i = 1, 2. Pi is called the selectivity of Ri on attribute A. It is given by **| Ri| / |A|.**

4.2.1. Reduction of simple tree queries using semi-join operation

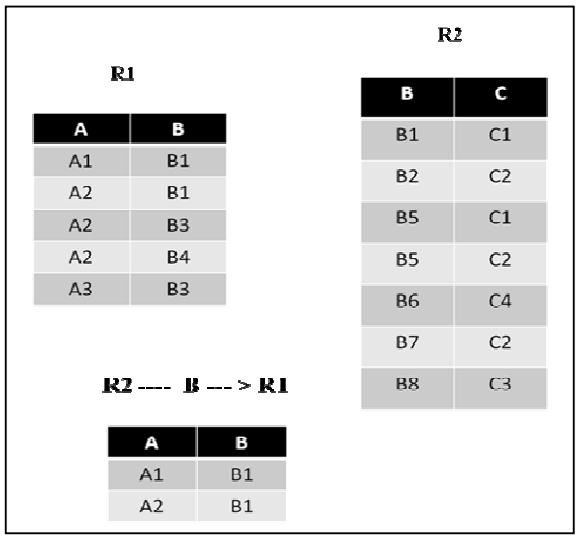

Figure 7. Semi-Join Operation

A simple query consists of only one joining attribute. Since the simple query is a tree query, all the relations in the query may be reduced by using semi-join operation [11][12][13]. We are considering two strategies for implementing Semi-joins.

Strategy 1: We will consider the relations which are in the ascending order of their size

Strategy 2: We will consider the relation which is to be reduced, not in the ascending order but in the last semi-join operation.

Consider the qualification of a simple query with ***n*** join clauses as ***for i = 1 to n , R_i. A = R_{i+1}.A***
Consider relations R1, R2, R3, R4 and joining attribute as A. The relations are in the ascending order of their join attribute. Consider simple query is (R1.A = R2.A) and (R2.A=R3.A) and (R3.A = R4.A). Their Corresponding sizes are S1, S2, S3, and S4.Their corresponding selectivities are SEL1, SEL2, SEL3, and SEL4. The Cost of initiating a message is C_0. The cardinality is C (Ri.A).

Strategy I:

Semi-join Operation will be as follows:
We will consider the sequence of semi-joins $R_{i+1} \ltimes Ri$ where i is indexed from 1 to 4.
STEP 1:
$R2 \ltimes R1$
Cost 1= C_0 + S1
C (R2.A) = S2 * SEL1

STEP 2:
$R3 \ltimes R2$
Cost 2= C_0 + C (R2.A)
C (R3.A) = S3* SEL2 *SEL1

STEP 3:
R4 ⋉ R3
Cost 3= C_0 +C (R3.A)
C (R4.A) = S4 * SEL3 *SEL2 *SEL1

Moving the result to resultant site
Cost 4= C_0 +C (R4.A)

Therefore, Total Cost = Cost1 +Cost2 +Cost 3+Cost 4

Strategy II:

Relation R2 is to be reducible so we consider R2 at the end of semi join operation

STEP 1:
R3 ⋉ R1
Cost 1= C (R1.A)
C (R3.A) = S3 * SEL1

STEP 2:
R4 ⋉ R3
Cost 2= C (R3.A)
C (R4.A) = S4 * SEL3 *SEL1

STEP 3:
R2 ⋉ R4
Cost 3= C (R4.A)
C (R2.A) = S2 * SEL4 *SEL3*SEL1

Therefore, Total Cost = Cost1+Cost2+Cost3

After comparing the total costs obtained by strategy I and II, select the strategy having the lowest cost.

4.2.2. Motivating Example

Consider the following example for illustration of both the strategies:

Illustration : Consider the simple query with R1 , R2,R3 and R4 as relations and A as their joining attribute. Result should be at the site of relation 2.Size and Selectivity of each relation is given below:

	Size	Selectivity
R1.A	**10000**	**0.1**
R2.A	**20000**	**0.2**
R3.A	**30000**	**0.3**
R4.A	**40000**	**0.4**

Figure 8. Relation with size and selectivity

Strategy I

Consider transmission cost =10. It is added to each calculation of cost.

We will consider the sequence of semi-joins $R_{i+1} \ltimes Ri$ where I is indexed from 1 to 4.

R2 $\ltimes$ R1
Cost = 10010
C(R2.A) = 20000 * 0.1 = 2000

R3 $\ltimes$ R2
Cost = 2010
C(R3.A) = 30000 * 0.2 *0.1 = 600

R4 $\ltimes$ R3
Cost = 610
C(R4.A) = 40000* 0.3 *0.2 *0.1 = 240

Moving the result to site of relation 2
Cost = 250

Therefore, Total Cost = 10010 +2010+610+250 = 12880 ---------- Strategy I

Strategy II:

Relation R2 is to be reducible so we consider R2 at the end of semi join operation

R3 $\ltimes$ R1
Cost = 10010
C(R3.A) = 30000 * 0.1 = 3000

R4 $\ltimes$ R3
Cost = 3010
C(R4.A) = 40000 * 0.3 *0.1 = 1200

R2 $\ltimes$ R4
Cost = 1210

C(R2.A) = 20000 * 0.4 *0.3*0.1 = 240

Therefore, Total Cost = 10010+3010+1210 = 14230 ---------- Strategy II
Comparing the total costs obtained by strategy I and II, and selecting strategy I.
So, **Total Cost = 12880**

4.3 Cyclic Query Optimization

There is no semi-join sequence which can reduce all relations in a cyclic query. So we cannot use semi-join strategies for cyclic query optimization. For processing a cyclic query, we first transform it into a tree query and then a tree query optimization procedure is applied. There are many ways of transformation from cyclic queries to tree queries. Some of the techniques are given below[10][14][15][16]:

4.3.1 Minimization of Vertices

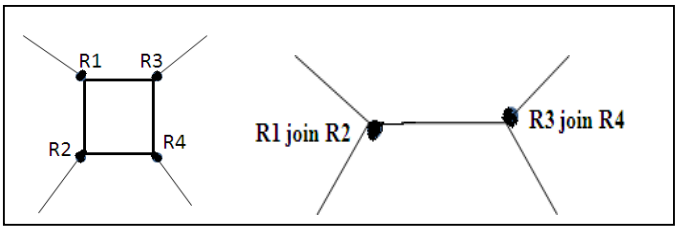

Figure 9. Removal of cycle using minimization of vertices

Pseudo- Code:
Consider a graph having vertices as relations Ri, Rj, Rk
Check whether vertices of relations forms a cyclic graph
If (Cyclic Graph)
Start transformation
Merge the vertices of relations such that resulting vertex is connected to every vertex which is connected to vi or vj or vk
End Transformation
Else
Graph is not a cyclic graph

4.3.2 Technique 2: Minimization of Edges

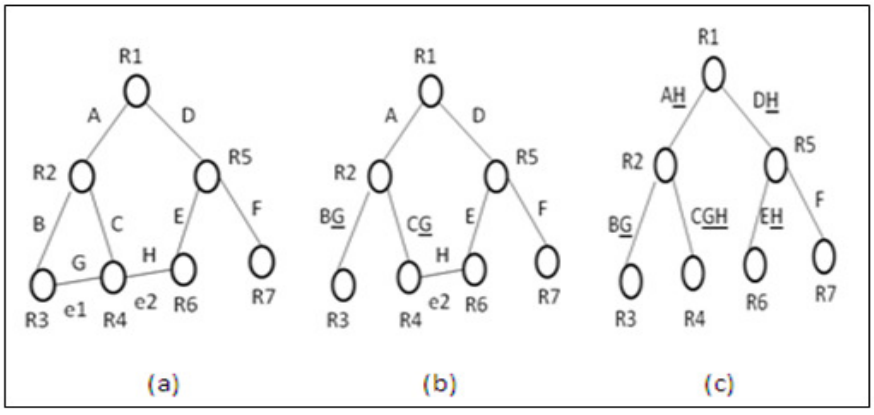

Figure 10. Removal of cycle using Attribute Addition Approach.

Pseudo-Code:
Consider relations Ri and Rj are joined on attribute A
Check one of the values common to Ri.A and Rj.A
If (common value exists)
Let common value be a
Replace the join on this value by two selection operations
Ri [A = a] and Rj [A =a].
Repeat till all A-values from Ri and Rj are over

4.3.3 Technique 3: Addition of Vertices

Instead of adding attributes we can convert queries by adding extra relations. After adding relation, apply following transformation to the graph.

Elementary Transformation: In a query graph, vertices acts as relations and edges are the joining attributes of two vertices.Consider three vertices as Vi, Vj, Vk and their edges be Y1, Y2 and Y3.If there is no edge between Vi and Vk, Y3 is assumed to be Ø. We can change the label Y3 to any Y4 satisfying the following condition, where $Z = E1 \cap E2 \neq \emptyset$. $E3 - Z \subseteq Y4 \subseteq Y3 \cup Z$. Here we assume $Y1 \neq \emptyset$. and $Y2 \neq \emptyset$. If Y3-Z is Ø, Y4 can be Ø. In such a case, the edge is removed. If Y3 is originally Ø, we can add edge e_{ik} whose label is a subset of Z.

Consider the following relations:
R1(ABC), R2(ADC), R3(EADB).The corresponding query graph forms the cycle, but addition of new relation R4(ACDB) removes the cycle.

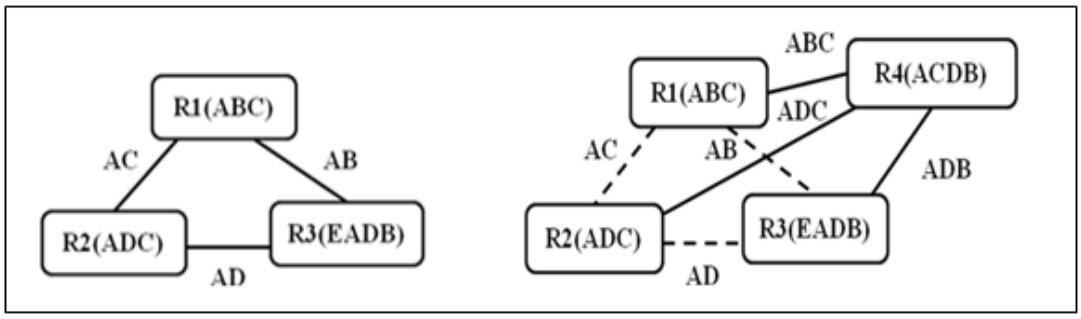

Figure 11. Removal of cycle using addition of vertices

Pseudo-Code:
Add vertices corresponding to the extra relations to the query graph.
Add appropriate edges.
Apply the elementary transformation to the resulting graph
Apply operations to compute the instances of the extra relations.

4.3.4 Technique 4: Decomposition of Vertices

To remove the cycle from a graph, we can divide decompose the relation horizontally, or if a relation satisfies a functional or multi-valued dependency, relation can be vertically decomposed [6][16][19][20].

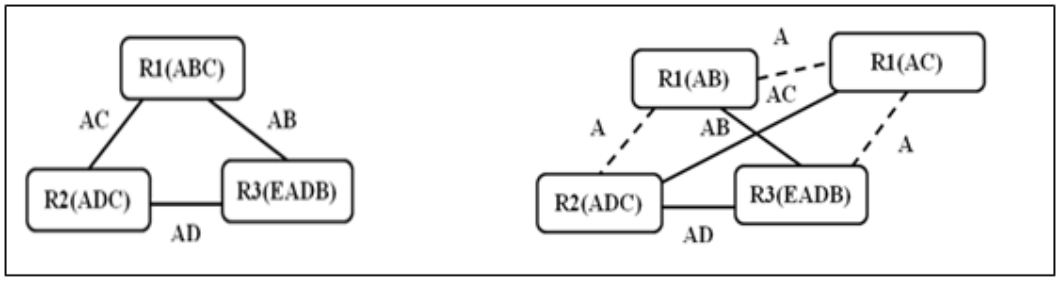

Figure 12. Removal of vertices using vertical decomposition of a relation

Consider relation R(ABC) and the functional dependency FD : A → B is required in relation R.The given relation R is not satisfing the required functional dependency.Horizontal decomposition R1 and R2 of relation R satisfies the Functional Dependency.

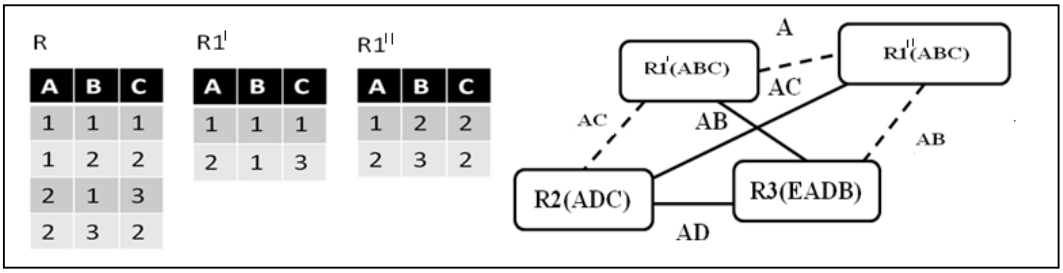

Figure 13. Removal of cycle using horizontal decomposition of a relation

5. CONCLUSION

By using the heuristic approach, we can reduce the cost of query optimization. As we are considering join of distributed queries, queries are divided into tree queries and cyclic queries. Semi-joins are full reducers for tree queries. The conversion of cyclic queries into tree queries uses different approaches such as addition of vertices, elimination of vertices, decomposition of relations and elimination of edges etc. Horizontal and vertical decomposition of relations is the best method for conversion of cyclic queries into tree queries.

REFERENCES

[1] Abraham Silberschatz, Hank Korth and S. Sudarshan. Database system concepts, 5th Edition. McGraw-Hill, 2006

[2] Ramez Elmasri and Shamkant B. Navathe. Fundamentals of database systems, Fifth edition. Pearson Education ,2009.

[3] http://www.techterms.com/definition/heuristic

[4] Won Kim,David S. Reiner,Don s. Batory. Query processing in database systems. Springer Verlag, Berlin Heidelberg New York Tokyo.

[5] Aper PMG, Hevner AR,Yao SB, "Optimization algorithms for distributed queries", IEEE Transactions on software Engineering, SE-9.1, January 1983, 57-68.

[6] Kambayashi Y, Yoshikawa M, "Query processing utilizing dependencies and horizontal decomposition", *Proc ACM SIGMOD conference*, San Jose, May 1983, 55-57.

[7] Kambayashi Y, Yoshikawa M, "Query processing for distributed databse using generalized semi-joins", *Proc. ACM SIGMOD International Conference on Management of Data*, June 1982, 151-160.

[8] Bernstein PA, Chiu DM, "Using semi-joins to solve relational queries", *Journal of ACM*, 28,1 ,January 1981, 25-40.

[9] Bernstein PA, Goodman N, "Power of natural Semi-joins", *SIAM Journal of Computing* , 10, 4, November 1981, 751-771.

[10] Goodman N,Shmueli O, "Transforming cyclic schemas into trees", *ACM SIGACT Symposium on Principles of Database Systems,* March 1982, 49-54.

[11] Goodman N,Shmueli O," Tree Queries : a simple class of relational queries", *ACM Transactions on Database systems*, 7,4, December 1982, 653-677.

[12] Gouda MG, Dayal UD, "Optimal semijoin schedules for query processing in local distributed database sytems", *Proc. ACM SIGMOD Conference* , Ann Arbor, 1981, 164-175.

[13] Goodman N, Shmueil O, "The tree property is fundamental for query processing", *Proc. ACM SIGACT-SIGMOD Symposium on Principles of Database Systems,* March 1982, 40-48.

[14] Fagin R, "Acylic Database schemes(of various degrees): a painless introduction", *IBM Research Report,* RJ3800, February 1983.

[15] Epstein R, Stonebraker M, Wong E, "Distributed query processing in a relational database system", *Proc. ACM SIG MOD Conference,* Austin, TX, May 1978, 169-180.

[16] Wong E, Youseffi K, "Decomposition – a strategy for query processing", *ACM Transactions on Database systems*, 1, 3 ,September 1976, 223-241.

[17] Yao SB, "Optimization of query evaluation algorithms", ACM Transactions Database Systems,4,2,June 1979,133-155.

[18] Yu CT, Lam K et al, "Distributed query optimization for tree queries", Dept. of Information Engineering, University of Illinois at Chicago Circle, October 1979.

[19] Yu CT, Chang CC, " On the design of a query processing strategy in a distributed database environment", Proc. ACM SIGMOD Conference, San Jose, 1983, 30-39.

[20] Yu CT, Chang CC et al, "Two surprising results in processing simple queries in distributed databases", IEEE COMPSAC, November 1982.

[21] Berstein, P.A. etal. "Query Processing in a system for distributed databases", ACM Trans. Database System Vol.6, No.4, Dec 1981.

[22] C.T. Yu and C.C. Chang, "Distributed Query Processing," ACM Computing Surveys, 16(4), 1984.

[23] http://www.csd.uoc.gr/~hy460/pdf/chenyu.pdf

Authors

Dr. Sunita M. Mahajan is currently Principal, Institute of Computer Science, MET, Mumbai. She worked in Bhabha Atomic Research Centre for 31 years. She obtained Ph.D. in parallel processing in 1997 from SNDT Women University and M.Sc. degree from Mumbai University in physics in 1966. She is a member of Indian Women Scientists Association, Vashi. Her research areas are parallel processing, distributed computing, data mining and grid computing.

Mrs. Vaishali P Jadhav is an Assistant professor in St Francis Institute of Technology, Borivali. She is a research scholar in NMIMS University, Mumbai. She obtained Master's degree in Computer Engineering from Thadomal Shahani College of Engineering, Mumbai. She is a member of IEEE and ISTE. Her research areas are database management, advanced databases, distributed computing, operating systems and artificial intelligence.