Presenting New Method to Optimize Query in Distributed Database System

Sajjad Baghernezhad
Department of Computer, Darab Branch, Islamic Azad University, Darab, Iran
sbaghernezhad@yahoo.com

ABSTRACT

Query optimization is one of the essential problems in centralized and distributed database. The data allocation to different sites is proposed in a distributed DMS(Database Management System) before a query in order to decrease, the next communicative costs namely an optimized bed production which is of 'NP' issues. In this article, it was attempted to examine both the methods to allocate data and produce optimized design in a distributed system and the space to query for query optimization in the distributed environment and show the need concerning optimization method in view of different aspects of optimization process. We install a new method for optimization in distributed database environment which indicates somehow our simple optimization design is executed relatively well until the database design is physical

KEYWORDS

Query Optimization, Distributed Database, Allocation, SQL

1 INTRODUCTION

In recent years the distributed database are used increasingly due to development concerning computer networks and database technology[1]. Distributed database system is dispersed physically but is centralized logically and is a composite of computer networks and database system; generally distributed database technology is the center of different researches such as general integration design, data exchange and query processing and optimization; the query optimization returns to the early distributed systems and recently many researches executed in relation to different potentials of data sources combination and costs model[2]. However, query optimization presents features in a developed distributed environment which changes considerably trade offs in the optimization process. The distributed query processors should take into consideration three essential principles to process and optimize users' query:

Necessary query processing: Processing query is the process to translate a query in a high level language such as SQL to a lower level language. It is possible to access the data and calculate in different sites in a distributed system in big scale so one of the essential goals of the distributed database systems development is to take into consideration some part of a query design in a distributed method to increase efficiency[3].

Necessary costs factors: Considering the tables are in the same place of a centralized database management system the query costs are measured on the base of a one-dimensional factor but in a distributed database it is necessary to control the costs logically by a database by dividing them in different dimensions; usually the response time and computation precision and accuracy are main factors to compute the query costs.

Necessary costs estimation: Considering the tables are among different sites in the distributed systems one of the main goals of distributed systems is to estimate the communicative cost among the sites; of course,

a centralized optimizer may not estimate accurately the operations' costs in many independent sites. In first section of this article we describe the query optimization and examine essential steps to optimize query in distributed databanks. In the second one we describe optimization architecture and define the problem and in the third one we describe how allocate the data to different sites and methods installing to produce the optimize design and in the following section we examine the proposed method to optimize query and finally we propose essential points for future researches.

2 DEFINING ARCHITECTURE AND RELATED PROBLEM

The most related sections of the system to optimize query are the proposers in independent sites and query optimizer among the devices (Figure). As the query optimizer may use a variety of optimization, algorithm in a centralized database system it is necessary to estimate the cost by essential sources or the fabricated proposers in a heterogeneous database system. The optimizer and relation proposers use two mechanisms:
1 – RFP (Request for proposition) where the optimizer uses one operation.
2 – Proposition by the proposers who estimate the cost (Figure 1).

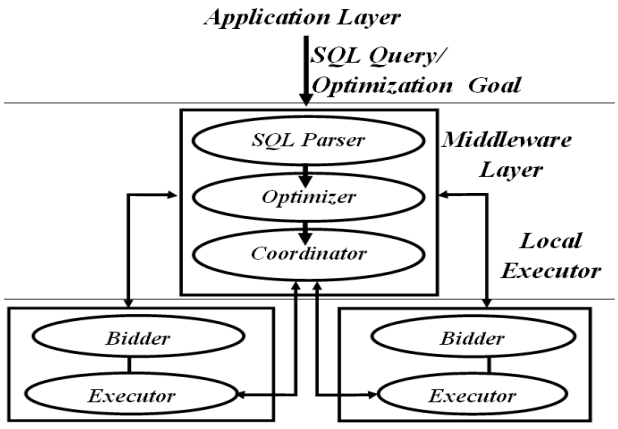

Figure 1. System architecture

3 THE DISTRIBUTED QUERY OPTIMIZATION PROBLEM

The most important issue is related to communicative cost among different sites in the distributed databanks. Contrary to centralized databanks the most important cost is related to time and the memory necessary to execute a query. The distributed query optimization problem is to find an execution design special to the user's query to achieve the goal proposed by the user; such goal may be a function formed by many variables such as response time, total execution cost and data accuracy; for simplicity we focus on two of them: execution time and total execution cost. The optimization issue may be proposed in two general phases in distributed databanks and each one may be examined separately; this section includes the data allocation in different sites to decrease communicative costs or minimize the exchanges in a distributed system and second section is related to produce executive optimized design for a query[4]. A query with link tendency formed of some links may be executed in different designs which has different executive cost but the same result; in continuation we try to describe in detail the two sections

3.1 Data allocation

The most important cost in the executive requests in a distributed database system is the cost for the cost of the data transfer achieved by a request from different site to a site where the request is executed. The data allocation is to define a measure from the frameworks in other sites to decrease total costs appeared during the execution of a collection of the requests. Having executed this method the time mean to execute the request which is very important in a usual distributed group and multimedia database system decreases; however, the problem related to NP data allocation method continues. The execution cost related to the

request depends on the request and data situation. In view of specialty the data situation in a request defines the amount of the data transfer in processing the request so when someone encounters the data allocation it had better he (she) improves the data. Having defined the collection of the requests achieved from the pieces of the data the pieces are allocated to the sites from the database to decrease the total cost of the data transfer for processing requests. The methods proposed in the field are described. It should be noted that a general form for databank, the method to allocate data and some limits are taken into consideration in some algorithms and some algorithms are stated, but we consider their general state in which the tables of a bank may be divided between several sites. Perhaps there are some copies of a table in several sites or each site may include only one table; with such presupposition we deal with proposed algorithms.

3.1.1 Genetic algorithm

This method select a primary group from division possibilities and enters into genetic cycle as chromosomes shown in an array frame as following algorithm[5]:

```
(1) Initialize population. Each
individual of the
population is a concatenation of the
binary
representations of the initial random
allocation of
each data fragment.
(2) Evaluate population.
(3) no of generation = 0
(4) WHILE no of generation < MAX
GENERATION DO
(5) Select individuals for next
population.
(6) Perform crossover and mutation for
the selected
individuals.
(7) Evaluate population.
(8) no of generation ++;
(9) ENDWHILE
(10) Determine final allocation by
selecting the fittest
individual. If the final allocation is
not feasible, then
consider each over-allocated site to
migrate the data
fragments to other sites so that the
increase in cost is
the minimum.
```

Figure 2. Genetic algorithm

3.1.2 Algorithm to query randomly beside

Main principle in a side searching method is to create a primary solution with medial quality; then based on neighbor defined before it selects a rapid solution in the searching space and tests if it is a better solution or not. If the new solution is better, it accepts its method and begin to query in new neighbor space; otherwise, it selects another solution. The method stop querying after some social steps or the solution stops after passing some stable steps. The quality of querying solution in neighbor space depends on creating neighbor solution; this method is defined to allocate the data as follows:

```
(1) Use Divisive-Clustering [19] to
find an initial
allocation Initial Alloc;
(2) Best Alloc = Initial Alloc;
(3) New Alloc = Best Alloc; iteration
= 0;
REPEAT
(4) searchstep = 0; counter = 0;
REPEAT
(5) Randomly select two sites from New
Alloc;
(6) Randomly select two data fragments
from each
site;
(7) Exchange the two data fragments;
(8) IF cost is reduced THEN
adopt the exchange and set counter to
0;
ELSE otherwise undo it and increment
counter;
UNTIL ++searchstep > MAXSTEP OR
counter >
MARGIN;
```

```
(9) IF cost(New Alloc) < cost(Best
Alloc) THEN
Best Alloc = New Alloc;
(10) Randomly exchange two data
fragments from
two randomly selected distinct
sites from New Alloc; /* Probabilistic
jump */
UNTIL iteration > MAXITERATION;
```

Figure 3. Algorithm querying randomly in neighborhood

3.2 Producing design to execute optimally and related works

In this section we examine the methods producing optimal design to execute a query. The methods producing an optimal design are sorted in two groups based on cost and rule; in the method based on rule essentially the findings are considered from a design in the best link graph and there is no space for a vaster space so this method is rapid and mostly there is no other better one and the algorithms find a better design after one execution. In the method based on cost the base is to apply statistic relation in the estimations and costs; it has query space and uses competency methods for query and finds the best design for little relations and there is no possibility to find design for many relations

3.2.1 Link graph

The query optimization methods are based on that if each database may be considered as a link graph in a way that each node shows a table and each edge shows the relation between the tables, both groups of the algorithms operate by virtue of the link graph(Figure 2)[6].

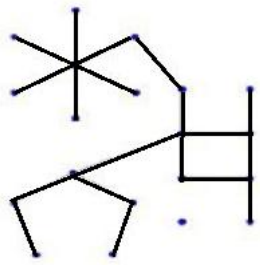

Figure 4. Joint graph.

Algorithms based on rule are the methods with low flexibility and often low efficiency and mostly no optimal design is achieved, but considering they select a design as an optimal design in one execution they are rapid. For instance, we may mention Prim and Kruskal algorithms; these methods are executable only for little databases with limited capacity and practically are not very efficient. In the algorithms based on cost the statistical relations and data in the system catalog are used to estimate costs, etc. And the query space is exponential and competency methods are used for query. The algorithms based on cost are sorted in two definitive and non-definitive groups. In the former algorithm is only a comprehensive and dynamic one and may not reply the great questions, but the non-definitive ones query for a graph whose nodes are alternative executive designs and may be used to reply the question; each node has a cost and the algorithm is to find a node with its least costs[7].

3.2.2 Dynamic programming algorithm

The advantage of this algorithm is to create the best possible design, but its time complexity is multi-phrases and its space complexity is not appropriate to complicated queries; specially in a distributive system the dynamic programming complexity is expensive for many queries[8]; one of the developed states of the dynamic programming algorithm is repetitive dynamic programming creating designs as good as dynamic programming algorithm for and complicated simple queries not available in the dynamic programming. Dynamic programming algorithm is shown for query optimization in algorithm No. 3 operating from down to up and creates more complicated design substructures by simpler ones.

```
INPUTS rels "List of relations to be
joined"
```

```
OUTPUT pt "Processing Tree"
partialsolutions := {All scans for all
attributes
involved}"Remove all elements from
partialsolutions
with equivalent, lower-cost
alternative"
FOR i := 2 .. |rels|
FOR all pt in partialsolutions
FOR all R in rels such that R not in
pt := pt ∞ R
END
END
"Remove all elements from
partialsolutions with
equivalent, lower-cost alternative"
END
RETURN "Arbitrary element from
partialsolutions"
```

Figure 5. Algorithm dynamic programming

3.2.3 Composite evolution optimization algorithm

Composite evolution optimization algorithm is created and used by composing genetic algorithm, learner's automata ,composing gene and chromosome concepts. The important property of composite evolution algorithm is its resistance against replies' superficial changes. Auto-restoration, reproduction, fine and reward are the composite algorithm features. Contrary to classic genetic algorithms the binary coding or natural overlay exposition are not used in the composite evolution algorithm. Composite evolution algorithm has higher efficiency than the genetic one.

```
Function query optimization (query)
Create the initial population CM1…
CMn;
EvalFitness();
While (Not (Stop Condition)) do
NewCM1 = CM with minimum Value of
Cost;
For i = 1 to n do
Select CM1; Select CM2;
If (Random > PC) then
Crossover (CM1, CM2);
End If
If (Random > PM) then Mutation (CM1);
Mutation (CM2);
End If
NewCMi+1 = CM1;
NewCMi+2 = CM2;
i=i+2;
End For
For i = 0 to n do
CMi = NewCMi;
For i=1 to 4
u = Random *n;
If (costu(CM.LAi)<MeanCost) then
Reward(CM.LAi , u )
Else Penalize(CM.LAi , u );
End If
End For
End For
EvalFitness();
```

Figure 6. Compositive evolution algorithm

4 HOW TO OPTIMIZE THE PROPOSED QUERY

The suppositions to discuss about algorithm and optimization technique proposed in this article are as follows:

Precise statistics: We suppose that there are precise statistics about cardinality and choice; such data may are collected from the standard protocols with permission to query from host database.

Relation costs: We suppose the relation costs are almost stable during optimization and query execution and the optimizer may meet the sustained relation costs in data transfer between two related sites.

There is no tube line throughout sites: We suppose that there is no tube line among the query operators throughout the sites; generally we divide all optimization algorithms in three steps:

1 – Selecting designs' substructures meeting the cost and preparation of the requests for proposals.

2 – Sending the message for the proposers of the request cost.

3 – Estimating costs for the designs and designs' substructures; if possible, to decide how to execute the design for query and if necessary, repeating the steps 2 and 3.

It is clear that we should try to minimize the number of the steps 2 and 3. Considering step 2 includes relation with some expense our proposed algorithm has tried to minimize the restoration from a great collection of data. Our proposed algorithm searches for all probable designs for query by using 'Up to down' method and optimal rule in the least time. However, algorithm finishes the work on due time and guarantees to find the optimal design for query execution,it is possible to divide the proposed algorithm in four steps as follows:

Step 1: Catalog of all joints and possible multiple joints which may be defined as a basic relation and an intermediate relation without production Cartesian multiplication.

Step 2: Creating a proposal request for the joints and estimating step 1 to scan basic tables.

Step 3: The request cots from the proposers for the joint and scan operation. If the entrance relations are the intermediate tables, only the single joint costs is requested for each joint with supposing the entrance costs had been estimated before.

Step 4: Estimating the designs' costs and related substructures as return by dynamic programming and finding optimal design for query.

Suppose the bank with relations designs as follows:

Branch (branch_name,branch_city,assets).
Client (customer_name,customer_street, customer_city).
Loan (loan_number,branch_name,amount).
Customer (customer_name,loan_number)
Account (account_number, branch_name, balance).
Deposit (customer_name,account_number).

We may distribute tables among three sites:

Site 1: Branch.
Site 2: Customer, customer, deposit.
Site 3: Loan, account (Figure 7).

Central data lexicon includes data related to the tables in the sites and defined designs for the tables. Now suppose following query:

```
SELECT customer_name, loan_number,
amount
FROM borrower , loan
WHERE borrower.loan_number =
Loan.loan_number
AND branch_name = 'Perryridge' and
amount > 1200;
```

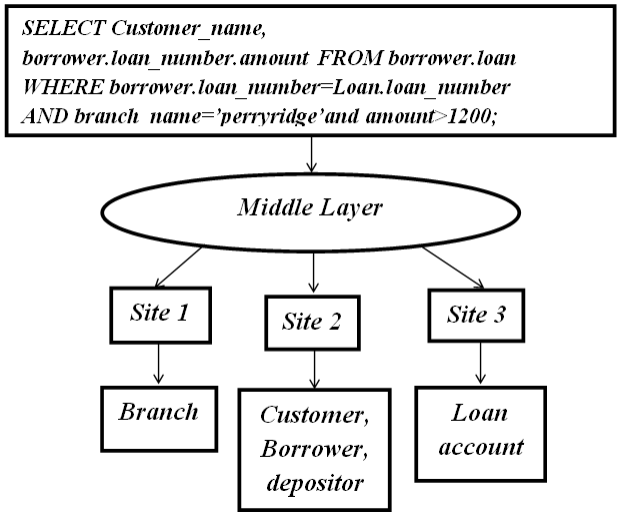

Figure 7. Proposed database

Step 1: In this step the catalog refers to central lexicon to define the sites for special query and creates 'N' SubSelects, SubWheres and fubForms in which there are 'N' sites; then SelectItems enter in SubSelect(n) and FormItems from the distributed sites in SubForms (n) (Figure 8).

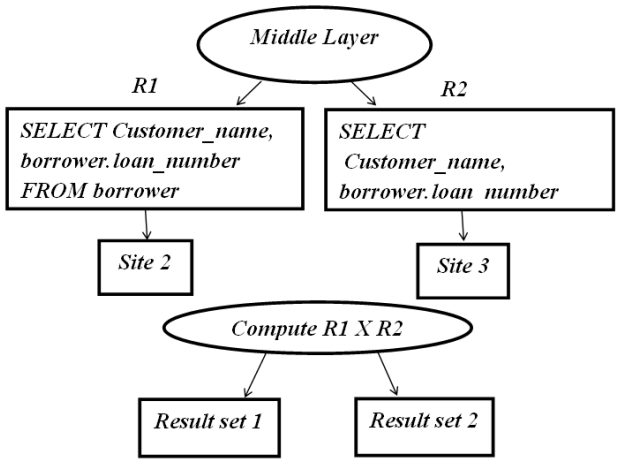

Figure 8. Step 1

Step 2: In this step we select each item from WhereItem list. If related element belongs to

that site completely, the element relates to SubWhere and if related element does not belong to a site completely, the element is located in final new list of Where (Element analysis related to the operation); then it searches the tables features one by one and finds the sites containing the features and if it does not include the features, the related SubSelect includes them. This step is one of the key steps in this algorithm; the example may describe it for us better(Figure 9).

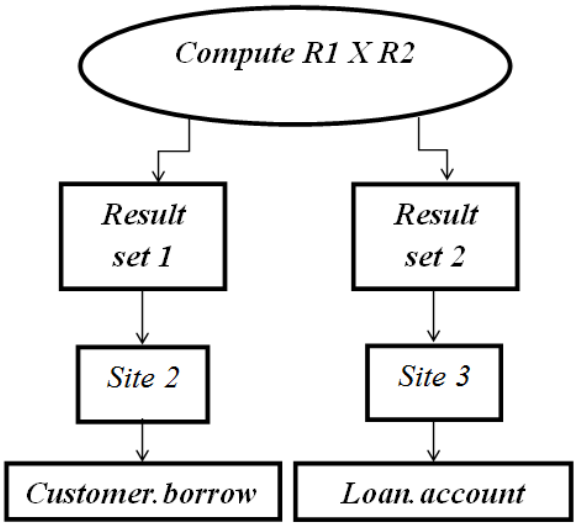

Figure 9. Step 2

Step 3: In this step we produce a SubQuery from SubSelect, SubWhere and SunForm lists.

```
FinalSelect = SelectItems
FinalFrom = {R1,R2…..,Rn}
```

Where:

n = Number of the sites.

Rn = The result of the collection achieved from 'n'; in this step we create query from FinalFrom, FinalSelect and FinalWhere lists.

Design execution:

```
SUBQUERY[1] =NIL
---------------------------------
SUBQUERY[2]:
SELECT
CUSTOMER_NAME, B.LOAN_NUMBER
FROM BORROWER ;
---------------------------------
SUBQUERY[3]:
SELECT AMOUNT, L.LOAN_NUMBER
FROM LOAN
WHERE BRANCH_NAME = 'PERRYRIDGE'
AND AMOUNT>1200 ;
---------------------------------
FINALQUERY :
SELECT
CUSTOMER_NAME,B.LOAN_NUMBER ,
AMOUNT
FROM R2, R3
WHERE
B.LOAN_NUMBER = LOAN.LOAN_NUMBER
```

Where:

R2 and R1 are the result from related parallel sites and final query in the middle layer, respectively(Figure 10).

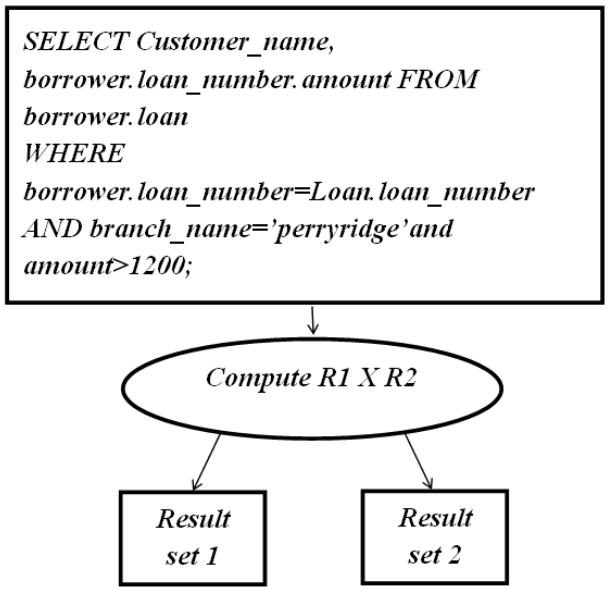

Figure 10. Design execution

Step 4: It includes queries substructure execution in related parallel sites and final query in the middle layer.

5 COMPARISON and CONCLUSION

Query optimization in distributed databank are examined from the views allocating data and producing optimal execution design, but considering both discussions are of NP there is no definitive reply for them; meanwhile, none of the presented algorithms produce optimal reply for all problems and have favorable

results only for some problems with special features; for example, the techniques querying for dynamic programming are appropriate to a little amount of queries, but such methods are not appropriate when the number of the relations in the query increases because high memory and process are used. The query optimization method for the query is very useful to distributed database systems. The optimizer should consult with the sources of the data involved in finding the operation cost to estimate the optimization process cost. In view of the cost concerning executive designs and number of the defined joints the proposed algorithm gives better results than composite evolution algorithm. The algorithms comparison indicates the proposed algorithm is better than the composite evolution algorithm. Having used this algorithm it is possible to achieve the reply more rapidly and prevent to trap algorithm in local minimums; so it can be said that the proposed algorithm is a more appropriate method to solve the problems of distributed database queries. The mentioned optimization process indicates in many cases specially when our database design is physical the query optimizer algorithm works well, but if we have not such data, we should use more aggressive optimization techniques

6 REFERENCES

[1] C. Shahabi, L. Khan, D. Mcleod. "A probe based technique to optimize join queries in distributed internet bases, Knowledge and Information Systems". Computer Science and Information Technology (ICCSIT), 3rd IEEE International Conference on, Volume 8. 2002.

[2] Z. G. Ives, D. Florescu, M. Friedman, A. Y. Levy, and D. S.Weld. "An adaptive query execution system for data integration.In SIGMOD", International Journal on Intelligent and Cooperative Information Systems,(6) 2/3:99–130, June 2009.

[3] C.Olston and J. Widom. "Offering a precisionperformance tradeoff for aggregation queries over replicated data". vldb.org/conf /2005/pp. 144.

[4] T.Oliveria ,"Evolutionary Query Optimization for Heterogeneous Distributed Database Systems", Engineering and Technology Volume33 September 2010.

[5] I. Ahmad, K. Karlapalem,and Y. Kwok, Siu-Kai So "Evolutionary Algorithms for Allocating Data in Distributed Database Systems", Distributed and Parallel Databases, January 2002, Volume 11, Issue 1, pp 5-32.

[6] S. Chaudhuri, "An Overview of Query Optimization in Relational Systems", PODS '98 Proceedings of the seventeenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems. 1998.

[7] M. Yannis, E. Ioannidis. "Query Optimization", ACM Computing Surveys (CSUR). Volume 28 Issue 1, March 1996

[8] M. Steinbrunn, G. Moerkotte, Alfons Kemper, "Heuristic and Randomized Optimization for the Join Ordering Problem". The VLDB Journal, The International Journal on Very Large Data Bases. Volume 6 Issue 3, August 2012.