A Novel Approach of Query Optimization for Distributed Database Systems

Deepak Sukheja [1], Umesh Kumar Singh[2]
1: Priyatam Institute of Technology and Management
Indore 452009, India

.
2: Institute of Computer Science, Vikram University
Ujjain, India

Abstract
Query optimization in distributed databases explicitly needed in many aspects of the optimization process, often making it imperative for the optimizer to consult underlying data sources while doing cost based optimization. This is not only increases the cost of optimization, but also changes the trade-offs involved in the optimization process significantly. The leading cost in this optimization process is the "cost of costing" that traditionally has been considered insignificant. The optimizer can only afford a few rounds of messages to the under- lying data sources and he nce the optimization techniques in this environment must be geared toward gathering all the required cost information with minimal communication.
In this paper, we explore the design and s earch space for a que ry optimizer in distributed environment and de monstrate the need for this optimization approach in various aspects of the optimization process. This work present minimum-communication cost query cost variants of various query optimization techniques, and di scuss trade-offs in their performance in the present development. We have implemented a nov el optimization approach in the distributed database environment, somewhat unexpectedly, indicate that a simple two-phase optimization scheme performs fairly well as long as the physical database design is known to the optimizer, though more determined algorithms are required.

Keyword: Query Optimization, Query Optimization approach, Query processing in DDBMS

Introduction

The need for distributed database services has increased dramatically in present working environment. Within enterprises, IT infrastructures are often decentralized as a result of mergers, acquisitions, and specialized corporate applications, resulting in deployment of large distributed databases. Possibly more significantly, the Internet and intranet have enabled new enterprise ventures including *Business-to-Business Net Markets* (or *Hubs*) [1, 2], who's business hinges on federating thousands of decentralized catalogs and other databases.
In general, distributed database technology has been the subject of multiple research thrusts, including schema integration [3, 4], data transformation [2], as well as distributed query processing and optimization. The query optimization work goes back as far as the early distributed database systems (R*, SDD-1, Distributed Ingres [5, 7]), and the lot of research's recently has been focused on l inking data sources of various capabilities and cost models [8]. However, query optimization in the broad distributed environment presents peculiarities that change the trade-offs in the optimization process quite significantly. Distributed query processors need to consider three basic requirements:
Need of Query Processing: I n a large scale distributed system, both data access and

computation can be carried out at various sites. For global efficiency, it is beneficial to consider assigning portions of a query plan in arbitrary distributed ways. In fact, this has been one of the major motivations for development of distributed database systems.

Need of Cost Factors: In a centralized DBMS, query execution "cost" is a single dimensional factor measured in conceptual units. In a distributed database, costs must be dividing into multiple dimensions under the control of single logical database. One proposal for a universal cost metric is hard currency, but typically there are other costs that are valuable to expose orthogonally, including response time, data freshness, and accuracy of computations [9, 10].

Need of Cost Estimation: This work is motivated by the necessity of the cost estimation in the part of the query optimizer from the optimization process. Apart from of the number of cost dimensions factor, a centralized optimizer cannot accurately estimate the costs of operations at many autonomous sites. Z. G. Ives and A. Tomasic perposed middleware systems in [8, 11] address this problem by involving site specific wrappers in the optimization process, but they do not consider the cost of communicating with these wrappers. This cost is not significant in these systems because the wrappers typically reside in the same address space as the optimizer. But in general, the execution costs may also depend on transient system issues including inter communication cost between two sites [9]. Therefore cost estimation process must be distributed in a manner reflective of the query processing, with cost estimates being provided by the sites that would be doing the work. However, to the best of our knowledge, complete cost estimation, which requires the optimizer to communicate with the sites merely to find the cost of an operation, has not been studied before. In such a scenario, communication may become the dominant cost in the query optimization process. The high *cost of costing* raises a number of new design challenges, and adds additional factors to the complexity of distributed query optimization.

Contributions of the Paper

In this paper, consider a large space of distributed query optimizer design alternatives and argu the need for taking into consideration the high "cost of optimization" in heterogeneous environment. Accordingly, its present minimum communication cost factors of various well-known optimization approaches and presents an effective algorithms for query optimization in the distributed database environments.

2. Architecture and Problem Definition

For query optimization purposes, the most relevant parts of the system are the query optimizer in the middleware, and the bidders at the fundamental sites (Figure 1). As in a centralized database system, the query optimizer could use a variety of different optimization algorithms, but the heterogeneous and distributed database system requires that the cost estimates be made by the original data sources or by the bidders. The optimizer and the bidder communicate through use of two mechanisms: (1) Request for Bid (RFB) that the optimizer uses to request cost of an operation, and (2) Bid through which a bidder makes cost estimates.

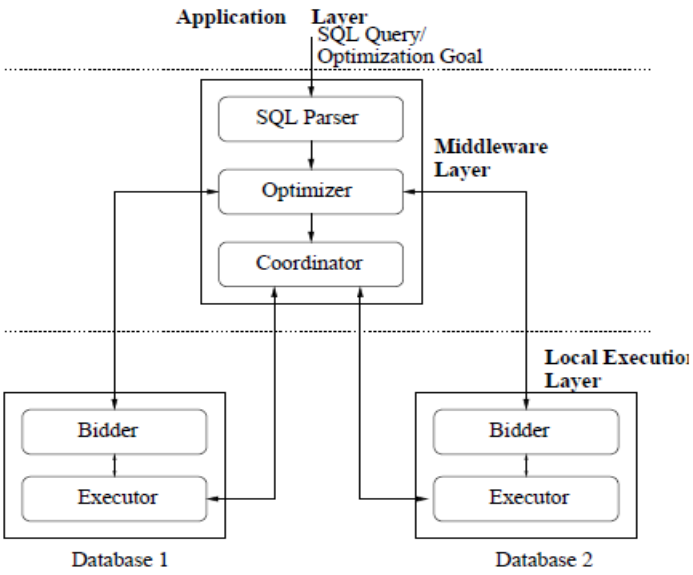


Fig1. System Architecture

The Distributed Query Optimization Problem

The distributed query optimization problem is to find an execution plan for a user specified query that satisfies an optimization goal provided by the user, this goal may be a function of many variables, including response time, total execution cost, accuracy and staleness of the data. For simplicity,

we concentrate on two of these factors, response time and total execution cost , though it is fairly easy to extend these to include other factors, assuming they can be easily estimated. Since we assume that the only information we have about the costs of operations is through the interface to the bidders, the optimization problem has to be restated as optimizing over the cost information exported by the bidders. Before describing the adaptations of the known query optimization algorithms to take into account the high cost of optimization, we will discuss two important issues that affect the optimization cost in this framework significantly.

Query Optimization Approach

To discuss the optimization technique and algorithm in this paper, it will make the following assumptions: **Accurate Statistics:** We assume that statistics regarding the cardinalities and the selectivity are available. This information can be collected through standard protocols that allow querying the host database about statistics, or by caching statistics from before query executions. **Communication Costs:** We assume that communication costs remain roughly constant for the duration of optimization and execution of the query, and that the optimizer can estimate the communication costs incurred in data transfer between any two sites involved in the query. **No Pipelining Across Sites:** We assume that there is no pipelining of data among query operators across sites.

In general, all optimization algorithms break into three steps:

Step 1: Choose subplans that require cost estimates and prepare the requests for bids.

Step 2: Send messages to the bidders requesting costs.

Step 3: Calculate the costs for plans/subplans. If possible, decide on an execution plan for the query, other- wise, repeat steps 2 and 3.

Clearly we should try to minimize the number of repetitions of steps 2 and 3, since step 2 involves expensive communication.

The proposed algorithm has tried to minimize the retrieval of large datasets. The cost measure that we have considered is the size of the retrieved result sets. The proposed algorithm searches all possible plans for the query, using top – down approach and the principle of optimality to cut subplans as early as possible. However the algorithm is finish it works in reasonable time in exponential nature and it is guaranteed to find the optimal plan for executing the query. Proposed optimization algorithms break into foure steps and these steps are as follows:

Step 1: Catalog all feasible joins and multi-joins. (A *feasible* relation is defined as either a base relation or an intermediate relation that can be generated without Cartesian product; a *feasible* join is defined to be a join of two or more feasible relations that does not involve a Cartesian product).

Step 2: Create bid requests for the joins and compute step 1 for scans on the base tables.

Step 3: Request costs from the bidders for these join and scan operations. If input relations are intermediate table in that case for each join, only request the cost of performing that individual join and assuming that the input relations have already been computed.

Step 4: Calculate the costs for plans/subplans recursively using classical dynamic programming (partial order dynamic programming if multidimensional costs are desired) and find the optimal plan for the query.

Consider a banking enterprise (given in the text book) having the following relation schemas:

branch (branch_name,branch_city,assets)

customer (customer_name, customer_street, customer_city)

loan (loan_number,branch_name,amount)

borrower (customer_name,loan_number)

account (account_number, branch_name, balance)

depositor (customer_name,account_number)

We could distribute the tables among 3 sites:

Site 1: branch

Site 2: customer, borrower, depositor

Site 3: loan, account

There will be a **central data dictionary** which will contain the information regarding which tables are in which sites and the schema definition of the tables.

Now consider a query:

```
select  customer_name, loan_number, amount
from borrower , loanl
where borrower.loan_number =
Loan..loan_number and
branch_name = 'Perryridge' and amount ≥ 1200;
```

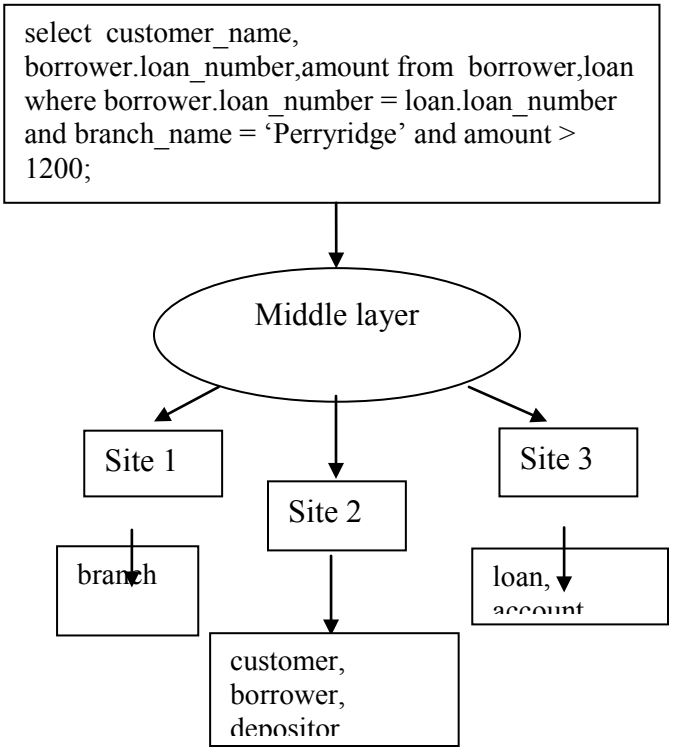
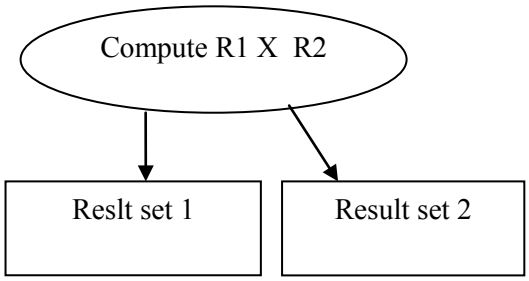


Fig 2.

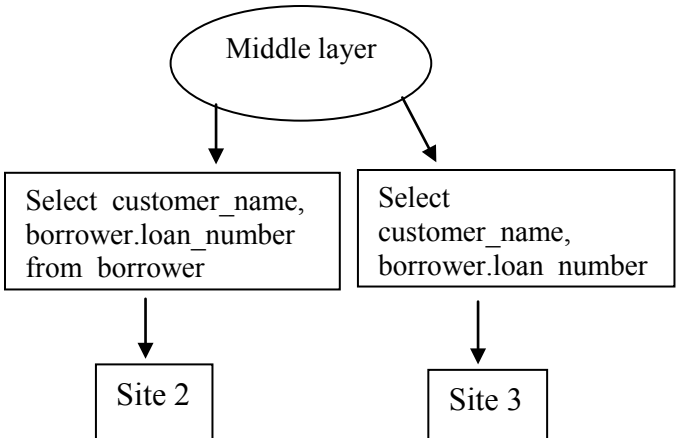
Step 1: Look at the From clause and refer catalog for the central data dictionary to determine which sites are to be considered for this particular query. Create 'n' number of SubSelect, SubFrom and SubWhere lists where n: no. of sites Distribute the SelectItems into the SubSelect(n) lists, FromItems into the SubFrom(n) lists.

R1 R2

Fig 3.



Step 2: Pick each item from the Where-Items list. If that clause belongs completely to a site, then include that clause into the corresponding SubWhere list. If a clause does not belong completely to one site then put that clause into a new list Final Where. Parse that clause to get the operands. Look for attributes of tables in individual operands. Find the sites to which these attributes belong to. If these attributes are not included in the respective Sub Select then include them. This step is one of the key steps in this algorithm. It is not intuitive and the example would help us understand it.



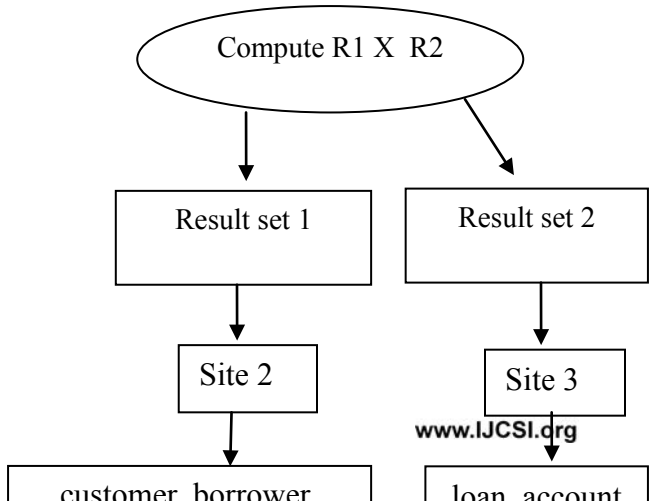


Fig 4.

Step 3: Using the SubSelect, SubFrom and SubWhere lists' generate the subquery.
FinalSelect = SelectItems
FinalFrom = {R1,R2…..,Rn) where n : no of sites and rn stands for the result set obtained from the nth site. Generate the final query from the FinalSelect, FinalFrom and FinalWhere lists.
Execution Plan:
SUBQUERY[1] =NIL
SUBQUERY[2] :
SELECT
CUSTOMER_NAME,B.LOAN_NUMBER
FROM BORROWER ;

SUBQUERY[3] :
SELECT AMOUNT, L.LOAN_NUMBER
FROM LOAN L
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 R1, R2 are result sets obtained from sites 2 and 3 respectively.

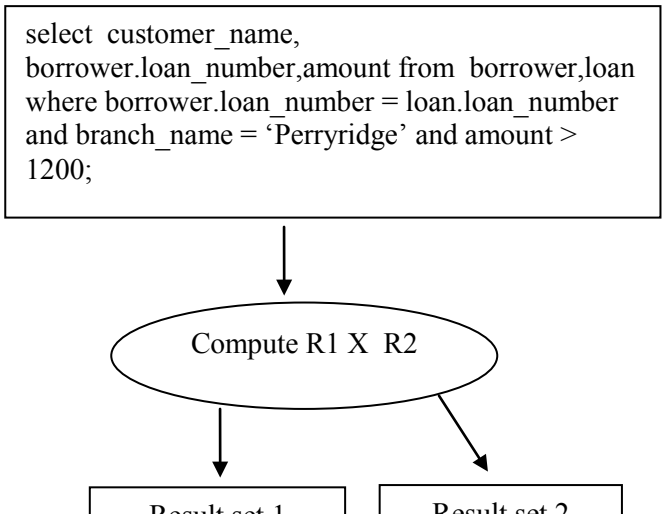


Figure 5.

Step 4: Execute the sub queries at the respective sites parallel (intra-query parallelism) and the final query at the middle layer.

Conclusion

A presented approach of query optimization is very useful for distributed database systems. To computations of cost from the optimization process, the optimizer must consult the data sources involved in an operation to find the cost of that operation. The mentioned analytical process indicate that, in many cases, especially when the physical database design is known to the optimizer, this query optimization algorithm works very well. In absence of such information, more aggressive optimization techniques must be used.

References

[1]Net market makers inc. http://www.netmarketmakers.com, 1999.

[2] L. Knight. "the e-market maker revolution", dataquest inc. www.netmarketmakers.com/documents/perspective1.pdf, 1999.

[3] *M. Lenzerini, and S.B. Navathe. A comparative analysis* of methodologies for *database scheme integration. ACM Computing Surveys*, 15(4):323–. 364

[4*R. J. Miller, L. M. Haas, and M . Hernández*, "*Schema Mapping as Query. Discovery," in VLDB, 2000*, pp. 77–88.

[5] L. M. *Haas*, P. G. *Selinger*, *E*. *Bertino*, *D*. *Daniels*, B. G. *Lindsay*, G. M.. *Lohman*, *Y*. *Masunaga*, *C*. *Mohan*, *P*. *Ng*, P. F. *Wilms*, and R. A. *Yost*. *R**: A *research project on distributed relational dbms*. *IEEE* Database Eng. Bull.,. 5(4):28–32.

[7] *P. A. Bernstein*, *N*. *Goodman*, *E*. *Wong*, *C. L. Reeve, and J. B. R. . Jr*., "*Query processing in a system for distributed databases* (*SDD-1*),". *TODS*, vol. 6, no. 4, *1981*. [15] R. S. Epstein, M. Stonebraker, **...** Databases Where Relations Are Hash Partitioned"," *TODS*, vol. 16, no. 2, pp. 279–308.

[8] Z. G. Ives, D. Florescu, M. Friedman, A. Y. Levy, and D. S.Weld. An adaptive query execution system for data integration.In *SIGMOD*, 1999.

[9] R. Avnur, J.M. Hellerstein, B. Lo, C. Olston, B. Raman, V. Raman, T. Roth, and K. Wylie, "CONTROL: Continuous Output and Navigation Technology with Refinement On-Line", in Proc. SIGMOD Conference, 1998, pp.567-569.

[10] C.Olston and J. Widom. Offering a precision-performance tradeoff for aggregation queries over replicated data. vldb.org/conf /2000/pp. 144.

[11] A. Tomasic, R. Amouroux, P. Bonnet, O. Kapitskaia, H. Naacke, and L. Raschid. The distributed information search component (DISCO) and the world wide web. In IEEE 1998, Volume: 10 Issue: 5 pp. 808-823.