Query Decomposition, Optimization and Processing in Multidatabase Systems

Cem Evrendilek Asuman Dogac

Software Research and Development Center
Scientific and Technical Research Council of Turkiye
Middle East Technical University (METU)
06531 Ankara Turkiye
email: asuman@srdc.metu.edu.tr

Extended Abstract

1. Introduction

One way of achieving interoperability among heterogeneous, federated DBMSs is through a multidatabase system that supports a single common data model and a single global query language on top of different types of existing systems. The global schema of a multidatabase system is the result of a schema integration of the schemas exported from the underlying databases, i.e., local databases. A global query language is used by users of the multidatabase system to specify queries against the global schema.

Three steps are necessary to process the global queries [MY 95]: First a global query is decomposed into subqueries such that the data needed by each subquery are available from one local database. Next each subquery is translated to a query or queries of the corresponding local database system and sent to a local database system for execution. Third the results returned by the subqueries are combined into the answer.

Multidatabase query optimization has been addressed in [CHO 94], [DKS 92], [LS 92], [LOG 92], [MY 95], [SC 94]. In [CHO 94] and [SC 94], a new method for schema and data integration in multidatabase systems is presented that gives the answer to the query as a set of sets representing the distinct intersections between the relations. Furthermore extensions to MSQL are presented along with a query processing strategy which reduces the amount of data transmitted. In [DKS 92] cost parameters for query optimization in heterogeneous DBMSs is estimated through database calibration. In [LS 92] the query optimization techniques developed for distributed database systems are adapted to the federated DBMS environment. In [LOG 92] it is claimed that global query optimization problem is fundamentally different from global query optimization in homogeneous distributed database systems. Furthermore major issues in the design of a multidatabase query optimizer are highlighted. In [MY 95] a survey of the previous research on multidatabase query optimization is presented.

In this paper we consider the optimization of query decomposition in case of data replication and the optimization of intersite joins, that is, the join of the results returned by the subqueries. The optimization algorithm presented for intersite joins can easily be generalized to any operation required for intersite query processing. The algorithm presented in this paper is distributed and takes the federated nature of the problem into account.

2. Optimizing query decomposition in case of data replication

Notice that if there is data replication in a multidatabase there may be more than one site where the subquery can be executed. In decomposing the query in case of data replication, we will try to balance the load distribution of the local DBMSs. Since we may not know about the local loads of the sites, most reasonable thing to do is to distribute the load of the global query evenly. If the loads of the sites are available, then the algorithm can be modified easily.

Then the query decomposition in multidatabases in case of data replication can be formulated as the following optimization problem: given an initial assignment of relations and fragments to the sites, distribute the modified and decomposed query to the sites such that the independent parallel execution time is optimized, i.e., the load distribution is balanced.

For this assignment problem we propose the following heuristic:

Given the weight w(qi, sj) which denotes the cost of executing subquery i at site j we need to find a mapping f:{qi}$\rightarrow${sj} where qi$\in${1,...,n} and sj$\in${1,...,m}.

This mapping gives the execution site of each subquery. The cost of such a mapping is:

$$\max_{sj \in \{1,...,m\}} \left(\sum_{qi:\,[f(qi)=sj]} w(qi, sj) \right)$$

Our objective is to find a minimum cost mapping. This is an NP Complete Problem (by reduction from an NP Complete problem 2-PARTIT. 2-PARTIT problem is, given m integers summing to 2*k, dividing this set of integers into 2 partitions such that each partition sums up to k). For this purpose we suggest a greedy heuristic algorithm that tries to achieve the least increment possible at each step as follows:

1. Calculate all w(qi, sj), that is, find the cost of processing each subquery at each site, taking into account the necessary transmission and conversion costs.

2. Find max {w(qi, sj)}. If there are more than one candidate for the maximum value, choose the subquery having the minimum w(qi, sj) among the candidate subqueries. Set subquery=qi, and among w(subquery,sj) choose sj such that w(subquery,sj) is minimum. Set site=sj.

3. Delete all w(subquery, j) for all sites. Increase w(i, site) values by w(subquery, site).

4. Go to Step 2 until all subqueries are assigned to the sites.

We will explain this algorithm through an example.

Assume there are three sites and the costs of executing the subqueries at those sites are as follows:

	Site 1	Site 2	Site 3
Subquery 1	w(1,1)=5	w(1,2)=7	w(1,3)=9
Subquery 2	w(2,1)=6	w(2,2)=8	w(2,3)=8
Subquery 3	w(3,1)=6	w(3,2)=7	w(3,3)=6
Subquery 4	w(4,1)=15	w(4,2)=13	w(3,4)=14

In the first iteration max {w(qi, sj)}=w(4,1)=15, subquery=4, minimum w(subquery,sj)= w(4,2)= 13, site=2, that is the subquery 4 will be executed at site 2 with cost 13. Delete subquery 4, increase all w(i,2) by 13. For the second iteration, we start with the following configuration:

	Site 1	Site 2	Site 3
Subquery 1	w(1,1)=5	w(1,2)=20	w(1,3)=9
Subquery 2	w(2,1)=6	w(2,2)=21	w(2,3)=8
Subquery 3	w(3,1)=6	w(3,2)=20	w(3,3)=6

In the second iteration max {w(qi, sj)}=w(2,2)=21, subquery=2, minimum w(subquery,sj)= w(2,1)= 6, site=1, that is the subquery 2 will be executed at site 1 with cost 6. Delete subquery 2, increase all w(i,1) by 6. For the third iteration, we start with the following configuration:

	Site 1	Site 2	Site 3
Subquery 1	w(1,1)=11	w(1,2)=20	w(1,3)=9
Subquery 3	w(3,1)=12	w(3,2)=20	w(3,3)=6

In the third iteration max {w(qi, sj)}=w(1,2)=w(3,2)=20. Since there are more than one candidate for the maximum value, choose the subquery having the minimum w(qi, sj)=w(3,3)=6 among the candidate subqueries 1 and 3. Then subquery=3, minimum w(subquery,sj)= w(3,3)= 6, site=3, that is the subquery 3 will be executed at site 3 with cost 6. And finally Subquery 1 will be executed at site 1. The algorithm in this example provided the optimal solution.

Notice that there are several other possible greedy heuristics for this problem.

3. Optimizing Intersite Joins in Multidatabases

In multidatabase systems because of the site autonomy, it is not possible to interfere with the execution of the queries at the related sites. The information vital for query optimization about the local database systems, such as statistical information on cardinalities and selectivities, and indices defined are not available to the global query optimizer. However by using the work described in [DKS 92] the appearance times of temporary results from heterogenous DBMSs can be estimated. The problem yet to be solved is, given the estimates of the appearance times of the temporary results, optimizing the processing of the query.

In optimizing the total execution time of queries the critical part of the process is the scheduling of the intersite joins. Since appearance times of partial results can be different the solution to the problem should take into account the appearance times in addition to the classical parameters of join ordering, i.e., the cost of the join operation and the join selectivity. Yet another parameter to consider is the conversion cost due to the heterogeneity.

We formulate this problem as follows: Let cost(i,i) = t_i denote the appearance time of the partial result from the ith site.

cost(i,i+1) denotes the minimum completion time of the join of ith and i+1st partial results.
cost(i,i+1)= $\max(t_i,t_{i+1})+jc_{i,i+1}$

where $jc_{i,i+1}$ denotes join cost. In this section we assume that a join is performed in the best possible way. How to process a join is explained in Section 4.

It should be noted that the join ordering is an NP-Complete problem [OL 90]. With the additional time parameter the problem does not change its nature. It follows from the definition of NP-Completeness. That is we can reduce the join ordering problem to the intersite join ordering problem in polynomial time by specifying the appearance time of partial results as zero.

So we propose the following recurrence relation as a heuristic solution to this problem which is briefly sketched in the following due to space considerations. The details are given in [ED 94].

cost(1,n)= $\min_i \{\max(\mathrm{cost}(1,i), \mathrm{cost}(i+1,n))+jc_{i,i+1}\}$ where $1\leq i <n$ where $jc_{i,i+1}$ denotes cost of processing an intersite join between the partial results ($R1 \bowtie R2 \bowtie \ldots \bowtie Ri$) and ($R_{i+1} \bowtie \ldots\ldots \bowtie Rn$).

This computation can be visualized as the computation of the upper triangle of an n by n matrix M where M(i,j) denotes the minimum time it takes to compute the intersite joins for partial results i through j, that is, M(i,j) = cost(i,j). Notice that this matrix is symmetric because we assume that the join order and the join method is priori decided as explained in Section 4. Therefore calculating the upper triangle is enough. The order for computation is given as follows:

```
for f=1 to n-1 do
        for i=1 to n-f do
                compute M(i,i+f);
```

Notice that this algorithm will compute the elements diagonal wise so that the recomputation of the previous results are avoided.

The complexity of this computation is:

$$\sum_{i=1}^{n-1} i*(n-i) = n*(n-1)*(n+1)/6$$

that is $O(n^3)$.

To provide an insight to the algorithm we will provide a simple example where the details such as intersite join calculations, appearance time estimations and database integration issues are omitted.

Consider the following three databases stored in heterogeneous systems:

Stu_Regis_Office(stu_id, name, address, status) at site 1

Library(stu_no, stu_name, book_borrowed) at site 2

CEng_dept(id, name, course, grade) at site 3

and the following query: Find MS students of the CEng department who borrowed more than 10 books from the library and who obtained an A in the Database course.

This query is trivially decomposed as follows:

Q1. At site 1: Find MS students
Q2. At site 2: Find the students who borrowed more than 10 books from the library
Q3. At site 3: Find the students of the CEng department who obtained an A in the Database course.

The result requires Q= Q1⋈Q2⋈Q3 to be computed. The diagonal entries in the following table denote the appearance times of the partial results at the respective sites.

subquery results	1	2	3
1	cost(1,1)=2 msecs.	cost(1,2)	cost(1,3)
2		cost(2,2)=5 msecs.	cost(2,3)
3			cost(3,3)= 3 msecs.

Now we will calculate cost(1,2), cost(2,3), cost (1,3) in this order. Assume $jc_{1,2}$=3 msecs., $jc_{2,3}$=2 msecs., $jc_{(1,2),3}$= 2 msecs. and $jc_{1,(2,3)}$= 2 msecs. are the best possible intersite join times found as explained in Section 4.

cost (1,2) = max{cost(1,1), cost(2,2)}+ $jc_{1,2}$= 8 msecs.
cost (2,3)= max{cost(2,2), cost(3,3)}+$jc_{2,3}$= 7 msecs.
And finally cost(1,3)=min{(max{cost(1,2), cost(3,3)}+$jc_{(1,2),3}$), (max {cost(1,1), cost(2,3)}+$jc_{1,(2,3)}$)}
= min {10, 9}= 9 msecs.
and thus the join scheduling is Q= Q1⋈(Q2⋈Q3).

4. Performing a join in a heterogeneous environment

In Section 3 it is assumed that a join (R1⋈R2) is performed in a best possible way. In this section we will consider the possible ways of performing a join in a heterogeneous environment. The parameters involved in an intersite join ($R1 \bowtie R2$) are:

conv_cost(R1): Conversion cost of R1

conv_cost(R2): Conversion cost of R2

commun_cost(R1): Communication cost of R1

commun_cost (R2): Communication cost of R2

proc1($R1 \bowtie R2$): The cost of processing ($R1 \bowtie R2$) at site 1

proc2 ($R1 \bowtie R2$): The cost of processing ($R1 \bowtie R2$) at site 2

conv_cost_result: The cost of converting the result into the canonical data model

The possible ways of performing this join can be as follows: after converting R1 send R1 to site 2 (or the opposite), perform the join ($R1 \bowtie R2$) at site 1 through semi-join (or the opposite), send R1 to site 2 and perform the conversion at site 2 (or the opposite). Depending on system characteristics or on new research results like the one given in [SC 94] this list can be extended or may be reduced by considering the system characteristics.

Each of these strategies will determine the values of the related parameters defined above. The join strategy to be chosen is the one which gives the minimum cost where cost is defined as the summation of the related parameters.

As an example the cost of performing a join by converting R1 and sending R1 to site 2 is

conv_cost(R1) + commun_cost(R1) + proc2 ($R1 \bowtie R2$) + conv_cost_result.

It should be noted that the conversion cost parameter defined above includes entity identification [CHO 94] and generalized attribute derivation [LS 92]. Conversion cost may be a dominating factor in the overall cost. Distributed nature of our algorithm by exploiting parallelism partially compensates the conversion overhead.

Up to this point we have assumed that t_i is given, however t_i is an estimation. Even after calibrating the individual databases, due to local workload of the sites, those estimations may change. Therefore it is necessary to minimize the cost of rescheduling when t_i's change.
One approach would be not to reschedule at all. In this case it is necessary to guarantee that the algorithm will behave optimal on the average. Let p_i be the probability that the ith partial result will appear at time t_i. In order to guarantee this behaviour the expected value of t_i, $E(t_i)$, must be used in the algorithm.

$E(t_i) = \Gamma_i = p_i t_i + (1 - p_i) * C_i$
where C_i is a constant obtained from synthetic database calibration.

Lemma Γ_i gives the average estimated time for t_i with O(1) for a Ci chosen appropriately.

If we take Ci as the average value of estimated query evaluation time different from t_i through database calibration procedure then the Lemma is proved by the definition of the expected value of t_i , $E(t_i)$.

Corollary Algorithm run with Γ_i 's gives the optimal on the average.

Another approach is rescheduling. The cost of rescheduling when t_i changes is the cost of recomputing M(k,j) from k=1 to i and j= i to n.

5. Conclusions

The generalization of the second phase of the algorithm (optimization of the intersite joins) so that all operations required in intersite query processing is taken into account, is a future work. We also plan to extend this work by incorporating the loads of the sites to the overall optimization process when such an information is available.

References

[CHO 94] E. I. Chong, "Query Optimization in Distributed Database Systems and Multidatabase Systems", Ph.D Thesis, Northwestern University, 1994.

[DKS 92] Du, W., Krishnamurthy, R., Shan, M-C., "Query Optimization in Heterogeneous DBMS", Proc. of the 18th Int'l Conf. Very Large Data Bases, August 1992.

[DOBS 94] Dogac, A., Ozsu, T., Biliris, A., Sellis, T., Advances in Object-Oriented Database Systems, Springer-Verlag, 1994.

[ED 94] Evrendilek, C., Dogac, A., "Query Decomposition, Optimization and Processing in Multidatabase Systems", Technical Report MD-2, Software R&D Center, METU, Dec. 1994.

[LS 92] E-P. Lim and J. Srivastava. Query optimization/processing in federated database systems. Technical Report 92-68, Dept. of Comp. Sc., University of Minnesota.

[LST 91] H. Lu, M.-C. Shan, and K.-L. Tan. Optimization of Multiway Join Queries for Parallel Execution. In Proc. of 17th Int. Conf. on Very Large Data Bases, pages 549-560, 1991.

[LOG 92] H. Lu, B-C Ooi, C-H Goh. On Global Multidatabase Query Optimization, Sigmod Record, Vol.21, No. 4, December 1992.

[MY 95] W. Meng, C. Yu, "Query Processing in Multidatabase Systems", in Modern Database Systems (Edtr. Won Kim), ACM Press 1995.

[Ono 90] Ono, K., Lohman, G. M., "Measuring the Complexity of Join Enumeration in Query Optimization", Proc. of Intl. Conf. on Very Large Databases, 1990.

[SL 90] A. P. Sheth and J. A. Larson. Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases. ACM Computing Surveys, 22(3): 183-236, September 1990.

[SC 94] P. Scheurmann and E. Chong, "Role-based Query Processing in Multidatabase Systems", in Proc. of EDBT, 1994.

[SRL 93] L. Suardi, M. Rusinkiewicz, W. Litwin. Execution of Extended Multidatabase SQL. In Proc. of 9th Int. Conf. on Data Engineering, 1993.

[YC 84] C. Yu and C. Chang. Distributed Query Processing. In ACM Computing Surveys Vol. 16, pp. 399-433, 1984.