Query Optimization in Multidatabase Systems: Solutions and Open Issues

Tadeusz Morzy
Poznan University of Technology
morzy@put.poznan.pl

Zbyszko Krolikowski
Poznan University of Technology
zbyszko@cs.put.poznan.pl

Abstract

Query optimization in multidatabase systems differs from that of distributed homogeneous database systems due to the lack of information about cost formulae of component local database systems as well as due to less control over local query execution. The paper provides an overview of query processing in multidatabase systems. It includes the architecture of a multidatabase system, query execution model and multidatabase query optimization. Some results are surveyed and some open issues are discussed.

1 Introduction

Due to historical reasons, various types of database systems are employed concurrently by different departments of a large organization. These systems may range from legacy hierarchical and network database systems, through relational database systems to object-oriented and object-relational database systems, and to systems that manage unstructured data such as multimedia database systems and information retrieval systems. Although each system was designed and developed independently, mainly to satisfy the needs of data management of its own department, the data it manages can be useful to other departments in the organization. Thus, there is an obvious need to have a global and integrated view of the data stored in these pre-existing database systems to support some global applications accessing data residing at more than one departmental system e.g. OLAP applications. This need for integrating information from various systems has motivated the design of *multidatabase systems* (*MDBS*) A multidatabase system is a database system implemented to connect distributed, autonomous and (possibly) heterogeneous database systems, called *local database systems* (*LDBSs*). The *MDBS* provides homogenizing layer on *LDBSs* and, hence, it provides users an integrated transactional access through a uniform interface to data residing at more than one *LDBS*.

Two main features of local systems in this environment make such an integration very difficult: autonomy and heterogeneity. Autonomy means that each local database system maintains autonomy and complete control over its own data, and allows only indirect access to it. Heterogeneity refers to the various types of component local database systems which may have different user interfaces, data models, query languages, and query optimization strategies.

As in traditional homogeneous distributed database systems (*DDBS*), query optimization is an important problem that has a great impact on the performance of a multidatabase system [8]. A user can issue a global query on an *MDBS* to retrieve data from several local databases. Due to location transparency of an *MDBS* the user does not need to know where the data is stored and how the result is processed. The *MDBS* hides not only discrepancies in user interfaces and query languages, but also optimization and execution details of global queries. How to efficiently process such a global query is the task of *multidatabase query optimizer* (*MQO*). Like traditional distributed database systems, the *MDBS* chooses the execution plan for a global query that is of minimum estimated cost. Although the basic query optimization paradigm remains unchanged, query processing in *MDBSs* is quite different and more difficult than that in homogeneous distributed systems.

From a query processing point of view, the major difference between an *MDBS* and *DDBS* is local autonomy of component local database systems. The effects of local autonomy are manifested in both optimization as well as execution aspects of query optimization. First of all, local information needed for global query optimization, such as local cost formulas (models),database statistics typically are not available to the *MQO*. Secondly, the *MDBS* has neither direct control over subqueries executed at *LDBSs*, nor direct access to internal data structures of *LDBSs*. Hence, the *MQO* cannot predict the execution time of a query submitted to *LDBS*. Consequently, commonly recognized optimization solutions for global query optimization in traditional distributed database systems are not appropriate for *MDBS*.

This paper has two objectives. One is to provide a brief survey of results obtained in the area of multidatabase query optimization. The second one is to provide some important issues that require further research and development.

The paper is organized as follows. In section 2 the multidatabase system architecture is presented. Section 3 describes the query execution model and the decomposition of global queries. In Section 4 the multidatabase query optimization problem is formulated. In Section 5 a number of techniques that can be used for global query optimization are briefly discussed. Section 6 contains brief description of few open issues that we propose as a research agenda for the near future.

2 Multidatabase System Architecture

We define a *MDBS* as a collection of sites $\mathcal{S} = \{S_1, \ldots S_n\}$ with component local database systems $LDBS_1, \ldots, LDBS_n$, autonomous and possibly heterogeneous, and a *front-end system* that supports a single common data model and a single global query language on the top of these component local systems [25]. Each *LDBS* may have its own data model and a local database schema that is the schema managed by the local database management system. Each *LDBS* provides access to its data through an API supporting SQL, OQL or other query languages. The main task of the front-end system is the management of the global schema and the processing of the global queries accessing data from multiple local databases. We assume that the front-end system is duplicated in each site and that each site is capable of accepting global queries and carry out the query optimization process. Summarizing, there are two levels of query processing capabilities: one at the local database level within each component database system, and the second one at the front-end level. The main advantage of this approach is that a single query can be issued to access data from multiple *LDBSs* in an integrated manner without affecting existing application programs written for component local databases.

The portion of the local database schema available at the *MDBS* level is called an *export schema*. Export schemas of *LDBSs* are imported and integrated into a *federated schema*. Integration process may require syntactic and semantic transformations of data (e.g. mapping to the exchange model, unification of names, unification of domains values). During the integration process, possible data and structural inconsistencies are resolved [16].

3 Global Queries and Query Execution Model

The global schema of a *MDBS* presents an integrated view of the sharable data from multiple *LDBSs*. As the result, a global query language (e.g. MSQL) can be used to issue queries, called *global queries*, to retrieve data stored in *LDBSs*. Due to autonomy of *LDBSs*, there is a clear distinction in a *MDBS* between *local queries* and *global queries*. A local query is a query on the local schema that accesses only data controlled by a single *LDBS*. It is locally submitted to and executed by the *LDBS*, without the control of the *MDBS*. A global query is a query on the federated schema that accesses data controlled by more than one *LDBS*. It is submitted to the front-end system of the *MDBS* that coordinates its execution by interacting with the *LDBSs*.

When a global query is submitted it is decomposed into a set of local subqueries that can be directly evaluated by *LDBS*. The results returned by subqueries may be connected by a set of global conditions that has to be enforced at the front-end of the multidatabase system through the execution of federated join operations. By a federated join operation we mean a join of a local intermediate result, produced by a local subquery, and an imported relation, i.e. the result of a subquery imported from another site. The decomposition of the global query can be performed according e.g. to the algorithm presented in [22, 25].

More formally a global query may be represented as a quadruple $\langle \mathcal{R}_Q, \mathcal{C}^l_Q, \mathcal{C}^g_Q, \mathcal{A}_Q \rangle$, where $\mathcal{R}_Q$ is the set of relations of the federated schema involved in the query, $\mathcal{C}^l_Q$ is a set of *local conditions*, i.e. each involving only attributes from a single site, $\mathcal{C}^g_Q$ is a set of *global conditions*, i.e. each involving attributes from several sites and $\mathcal{A}_Q$ is the set of attributes that are in the result of the query.

We illustrate the above notions using the following example.

Example 3.1: Let us consider the multidatabase of a company with several divisions, each manufacturing several products. Each product is made of several parts, each supplied by several suppliers. For each order placed by a customer, the product, the quantity, the price and date are specified. For processed orders the shipping date, the shipping cost and the name of the shipping company are maintained. The system keeps also track of customers addresses and credits. The database is managed by a MDBMS consisting of five autonomous *LDBSs*, namely, $LDBS_1$, $LDBS_2$, $LDBS_3$, $LDBS_4$ and $LDBS_5$, located at sites S_1, S_2, S_3, S_4, and S_5. The federated schema is given as the union of the following export schemas:

S_1: Product(pid, name, did∗)
Division(did, name, city)

S_2: Part(tid, name, pid∗, sid∗)
Supplier(sid, name, city)

S_3: Order(ordid, qty, date, price, pid∗, cid∗)

S_4: Shipping(shid, date, shcost, shcompany, ordid∗)

S_5: Customer(cid, name, city, credit)

Let us now consider the global query:

> *Find all orders placed by customers in Rome for products that use parts supplied by suppliers in Poznan, and that have been shipped within two weeks from the order, and with total price (including shipping cost) greater than the customer credit. Return order id, product name and customer name.*

Referring to the schema above this would be expressed in SQL as:

```
SELECT O.ordid, Pd.name, C.name
 FROM Product Pd, Customer C, Part P.,
      Order O, Shipping Sh, Supplier S
 WHERE S.city = 'Poznan'  AND  S.sid = P.sid
 AND  P.pid = Pd.pid  AND  O.pid = Pd.pid
 AND  C.city = 'Rome' AND  O.cid = C.cid
 AND  Sh.ordid = O.ordid
 AND O.date + 14 < Sh.date
 AND  C.credit < O.price + Sh.cost
```

□

4 Global Query Optimization

In general, global query optimization process is decomposed into two steps. In the first step a global query is decomposed into a set of subqueries; in the second step the results returned by subqueries are connected through a set of federated join operations. At the global level (i.e. at the front-end level), the execution process of a global query is specified by a query execution plan (*QEP*) that defines a partial order in the execution of the federated join operations. Thus, the global query optimization problem consists in selecting the best *QEP*, i.e. the best partial order of the federated join operations. More formally, the global optimization problem may be formulated as follows. Given a global query $\mathcal{Q}$, the execution space $\mathcal{X}_Q$, consisting of all possible execution plans $\mathcal{P}$ for $\mathcal{Q}$, and a cost function $C(\mathcal{P})$ that associates an execution cost to each $\mathcal{P}$. We may then formulate the global query optimization problem as finding in the search space $\mathcal{X}_Q$ the execution plan(s) with the least execution cost: $min_{\mathcal{P}\in\mathcal{X}_Q} C(\mathcal{P})$.

We distinguish two optimization levels: *global optimization*: selecting the optimal *global* execution plan; *local optimization*: selecting the best *local* execution plans, i.e. the best way to perform each federated-join in the global *QEP* in the local site where it must be executed.

Both levels are strictly interconnected. Possible approaches to the query processing and optimization in a multidatabase system differ from each other on how both levels cooperate.

5 Solutions

The problem of multidatabase query optimization has recently attracted a number of contributions [1, 6, 7, 17, 18, 19, 20, 21, 22, 26, 27, 28]. There are several classifications of possible approaches for the processing and optimization of global queries in a multidatabase system [25].

In general, two approaches to the multidatabase query optimization problem are eligible [23]. The first one is based on applying traditional distributed query optimization techniques to generate the "optimal" query execution plan; the second one is based on the idea of delegating the evaluation of the execution cost of the elementary steps of a global query execution plan to *LDBSs* where computations are performed.

The applicability of the first approach depends on the ability to find approximate cost models for autonomous *LDBSs* – how to compute a cost of a local query? There are several proposals of new techniques of extracting or estimating local cost models [23]: (1) calibrated cost model, (2) query sampling cost model, (3) extensible cost model, etc. We describe very briefly two techniques for estimating the cost of local queries: *calibration method* and *sampling method*. The first technique was proposed by Du *et al.* in [6]. The idea is to develop a generic cost model for a *LDBS*, and, then, to use a synthetically created database and to run a set of queries against this database, to deduce the coefficients in cost formulas for the underlying *LDBS*. The values of coefficients reflect most system-dependent factors such as hardware speed, operating system and *LDBS* configurations. Moreover, the cost formulas include a system weight factor to reflect different load characteristics. Instead of using cost formulas based on physical parameters, authors developed a set of cost formulas based on logical parameters such as cardinality and selectivity. The results of experiments show that the estimated costs-based calibrated data and the actual costs measured at run-time are within 10 percent. However, this approach has some shortcomings [26]. The fundamental problem is that it may not be possible (or allowed) to create the calibrating database at some local sites.

Another technique for estimating the costs of local queries, called query sampling method, was proposed by Zhu and Larson in [26, 27, 28]. This technique tries to use sample queries to estimate cost parameters of local cost formulas. The idea of the query sampling method consists in grouping all possible queries to each *LDBS* into classes, and running a sample query workload for each class. The costs of sample queries are used to derive a cost formula for the queries in the query class. To estimate the coefficients of cost formulas the statistical procedure based on multiple regression analysis is applied. The accuracy of the estimation is closely related to how queries are classified into classes.

In contrast to the above approach, focusing on techniques of cost estimation of local query processing, in [22, 23] a new approach for multidatabase query optimization based on the cooperation of the *LDBSs* in the evaluation of the execution cost of a global query execution plan is proposed. The main idea of the approach is to delegate the evaluation of the processing of each elementary step of a global query to *LDBSs* where the computation would be performed. All *LDBSs* perform this evaluation process in parallel, and cooperate in sending to each other the cost estimates of the partial execution plans. It means that the total execution cost of a global query execution plan is computed incrementally through the cooperation of all sites involved in the query. Each site evaluates the cost of its part of the computation and the characteristics of the result, and eventually sends this information to sites that may use it for further computation. All sites cooperate, but each acts autonomously. The main advantages of this approach is that it does not require any assumption on local execution models and optimization strategies and preserves completely the local autonomy of sites.

There are also some other papers studying the multidatabase query optimization problem [7, 17, 18, 19, 20]. In [18, 19] the architecture of a multidatabase query optimizer and the discussion of problems arising from the heterogeneity and local autonomy of component *LDBSs* were presented and discussed. In [17], Lee *et al.* considered the *MQO* problem for replicated *MDBS*. In [7] the problem of reducing multidatabase query response time by tree balancing was considered. The authors propose the two-phase optimization strategy that first produces a left deep join tree query execution plan which is optimal with respect to total time. To produce this plan a traditional cost-based optimization algorithm is employed. Then, the plan is improved with respect to response time using tree transformations. A class of basic tree transformations and algorithms that effectively apply these transformations to balance a a left deep join tree query execution plan were presented. The tree transformation algorithms try to reduce the response time by finding cost reducing bushy trees. However, since they start with a left deep join tree, they may miss the least costly bushy trees.

6 Future Research Agenda

Query processing in multidatabase systems has been studied for more than 15 years. A lot of work has been done already, however, still much work remains. We list a few research topics that in our opinion are important and challenging.

6.1 Integration of Structured and Semistructured Data

Traditional multidatabase systems manage structured data. However, there is huge amount of data around that are not well structured, e.g. the content of the Web and other online data stores in which data are in the form of textual data, picture, video, image, and voice. The inclusion of the Web and online stores with semistructured data as components of a multidatabase system is one of the most challenging research topic now.

The Web itself is one large federated system. There is a huge demand for database technology to automate the creation, management, and querying the Web content. Different aspects of the problem are investigated under several research projects [10]: STRUDEL system (AT&T Research), TSIMMIS project (Stanford University), Infomaster, KOMET (University of Karlsruhe), Junglee, Araneus (University of Rome), YAT (INRIA, France), WebOQL (University of Toronto). The common theme of these projects is the declarative management of the content and structure of Web sites. Other group of research projects address also the problem of query languages for semistructured data: STRUQL, UnQL, LOREL, W3QS, ULIXES, DISCO - Distributed Information Search COmponents, POQL - Path Object Query Language (INRIA - Verso Group) and STARTS.

Only a few proposals have addressed the problem of query optimization and evaluation [4, 9, 11, 14]. As in relational systems, query optimization is done at multiple levels. Given a query, it is first translated into relational expression over set finite automata. At this level, factorization of common prefixes of regular expressions can be applied. The automaton representation is translated into the second-level representation, from which *QEP* are obtained. In searching for the best *QEP* the optimizer investigates a space of execution plans taking into account also the capabilities of data sources (i.e. data access with limited patterns).

6.2 Scalable Federated Systems

In the near future we can expect federated database systems of 1000 or more sites. We have to make it easy to integrate the information in these databases. There are several challenges in building scalable federated systems. One important issue concerns query optimizers that has to deal with federated systems of this size. A multidatabase query optimizer cannot simply construct a static optimal plan for a global query. First, some sites may be unavailable; second, even the best plan can behave arbitrarily badly in the presence of unexpected data transfer delays due to changing loads in sites and in the network. Moreover, there may be several replicas at various sites, and the quality of them may vary. Therefore, an optimizer must be able to deal with this quality of service issue. Another problem is how to decide that objects in two different databases refer to the same object in the real world (it is simply impossible to ensure uniform keys across 1000 databases). For all this reasons, the traditional static-cost-based approach to global query optimization seems inappropriate because it is unable to adapt in response to changing loads, unexpected delays or unavailability of sites.

Some solutions to the above mentioned problems already exist. The problem of unexpected data transfer delays was considered intensively by Amsaleg et al. in [1, 2, 24]. To address the problem query scrambling approach has been proposed. Scrambling modifies a query execution plan on-the-fly in response to unexpected delays that can arise due to communication problems in obtaining partial results. The plan is modified when a significant idling arise so that progress can be made on other parts of the plan. Query scrambling reacts in two ways: (1) by rescheduling the query execution plan (leave the plan unchanged but evaluate different operators), or (2) by operator synthesis (modify the plan by removing, adding or rearranging operators). The problem of matching objects across different data sources was analyzed by Cohen in [5]. To solve the problem a new logic called WHIRL is proposed. WHIRL retains the original names and reasons about the similarity of pairs of names using statistical measures of document similarity that have been developed in the information retrieval community. The answer to a user's query is a set of tuples. From the point of view of query optimization we deal with the new optimization problem – a query execution plan contains only cartesian products. The proposed solution is based on an $A*$ search strategy for solution tuples.

6.3 Efficiency and Quality of Answers

Traditionally, database systems have been designed based on the idea that for a given query there is a single correct answer. However, it is neither practical nor desirable to enforce this requirement in some environment. For example, a user might issue a query to a 1000-site federated database such as: "FIND THE AVERAGE SALARY OF PROFESSORS WORKING AT POLISH UNIVERSITIES". The user is not likely to get the exact result to the query within a reasonable time, if ever. Several factors such as heterogeneity, availability, usage cost, data quality, etc., impose new constraints on a multidatabase system. Therefore, it would be more reasonable to consider a query processing as an accumulation process. A database system should produce a "rough" answer as quickly as possible, and then, refine it over time until user decides that the result is good enough.

Allowing a flexible approach to query answer quality raises a number of interesting and challenging opportunities for optimization. Of course, this requires new techniques of data estimation, data delivery and new user interfaces. Recently, this problem has attracted a number of contributions [3, 12, 13, 15].

References

[1] Amsaleg, L., Franklin, M. J., Tomasic, A., Urhan, T., *Scrambling Query Plans to Cope With Unexpected Delays*, Proc. of the 4th Int. Conf. on Parallel and Distributed Information Systems, Miami Beach (Florida), 1996, IEEE Computer Society Press, p. 208-219.

[2] Amsaleg, L., Franklin, M. J., Tomasic, A., *Dynamic query operator scheduling for wide-area remote access*, Journal of Distributed and Parallel Databases, 6,(3), 1998, pp. 3-15.

[3] Barbara, D., et al., *The New Jersey data reduction report*, IEEE Data Eng. Bulletin, 20(4),1997, pp. 3-45.

[4] Buneman, P., et al., *A query language and optimization techniques for unstructured data*, Proc. of ACM-SIGMOD , 1996, pp. 505-516.

[5] Cohen, W. W., *Integration of heterogeneous databases without common domains using queries based on textual similarity*, Proc. of ACM-SIGMOD, Seattle, USA, 1998, pp. 201-212.

[6] Du, W, et al., *Query optimization in heterogeneous DBMS*, Proc. of the 18th VLDB Conference, Vancouver, 1992, pp. 277-291,

[7] Du, W., Shan, M.-C., and Dayal, U., *Reducing multidatabase query response time by tree balancing*, Proc. of ACM-SIGMOD, San Jose, USA, 1995, pp. 293-303.

[8] Evredilek, C., et al., *Multidatabase query optimization*, Journal of Distributed and Parallel Databases, vol. 5, No. 1, 1997.

[9] Fernandez, M., Florescu, D., Levy, A., and Suciu, D., *A query language for a web-site management system*, SIGMOD Record, 26(3), 1997, pp. 4-11.

[10] Florescu, D., Levy, A., *Current issues in data integration - tutorial*, 2nd Int. Symp. on Advances in Databases and Information Systems, ADBIS'98, Poznan, 1998.

[11] Florescu, D., et al., *Query optimization in the presence of limited access patterns*, Proc. of ACM-SIGMOD, Philadelphia, 1999, pp. 311-322.

[12] Gibbons, P. B., Matias, Y., *New sampling-based summary statistics for improving approximate query answers*, Proc. of ACM-SIGMOD, Seattle, USA, 1998, pp. 331-342.

[13] Hellerstein, J. M., Haas, P. J., and Wang, H. J., *Online aggregation*, Proc. of ACM-SIGMOD, Tucson, USA, 1997, pp. 171-182.

[14] Ives, Z. G., et al. *An adaptive query execution system for data integration*, Proc. of ACM-SIGMOD, Philadelphia, 1999, pp. 299-310.

[15] Jonsson, B. T., Franklin, M., and Srivastava, D., *Interaction of query evaluation and buffer management for information retrieval*, Proc. of ACM-SIGMOD, Seattle, USA, 1998, pp. 118-129.

[16] Kim, W., Choi, I., Gala, S., and Scheevel, M., *On resolving schematic heterogeneity in multidatabase systems*, in *Modern Database Systems* [ed. W. Kim], Addison-Wesley Pub. Co., 1995, pp. 521-550.

[17] Lee, C., Chen, C-J., and Lu, H., *An aspect of query optimization in multidatabase systems*, SIGMOD Record, vol. 24, No. 3, 1995, pp. 28-33.

[18] Lu, W, et al., *On global query optimization in multidatabase systems*, Proc. of 2nd Int. Workshop on Research Issues on Data Eng., Tempe, 1992, pp. 217-227,

[19] Lu, W, et al., *Multidatabase query optimization: issues and solutions*, Proc. of 3rd Int. Workshop on Research Issues on Data Eng., Vienna, 1993, pp. 137-143,

[20] Meng, W., and Yu, C., *Query optimization in multidatabase systems*, in *Modern Database Systems* [ed. W. Kim], Addison-Wesley Pub. Co., 1995, pp. 551-572.

[21] F. Ozcan, F., Nural, S., Koksal, P., Evrendilek, C., Dogac, A., *Dynamic query optimization in multidatabases*, Bulletin of the TC on Data Engineering, vol. 20, No. 3, September 1997, p. 38 - 45

[22] Salza, S., Barone, G., and Morzy, T., *Distributed query optimization in loosely coupled multidatabase systems*, Proc. of 5th Int. Conf. on Database Theory, Prague, 1995, pp. 40-53.

[23] Salza, S., Barone, G., and Morzy, T., *A distributed algorithm for global query optimization in multidatabase systems*, Proc. of the 2nd Int. Symp. on Advances in Databases and Information Systems, ADBIS'98, LNCS 1475, pp. 95-106.

[24] Urhan, T., Franklin, M. J., Amsaleg, L., *Cost based query scrambling for initial delays*, Proc. of ACM-SIGMOD, USA, 1998, pp. 130-141.

[25] Yu, C. T., Meng, W., *Principles of Database Query Processing for Advanced Applications*, Morgan Kaufmann Pub.1998.

[26] Zhu, Q, Larson, P-A, *A query sampling method for estimating local cost parameters in a multidatabase system*, Proc. of 10th Int. Conf. on Data Eng., Houston, 1994, pp. 144-153.

[27] Zhu, Q, Larson, P-A, *Building Regression Cost Models for Multidatabase Systems*, Proc. of the 4th Int. Conf. on Parallel and Distributed Information Systems, Miami Beach (Florida), 1996, IEEE Computer Society Press, p. 220-231.

[28] Zhu, Q, Larson, P-A, *Solving local cost estimation problem for global query optimization in multidatabase systems*, Journal of Distributed and Parallel Databases, vol. 6, No. 4, 1998, pp. 373-420.