Distributed View Expansion in Composable Mediators

Timour Katchaounov, Vanja Josifovski and Tore Risch

Uppsala Database Laboratory,
Uppsala University, Sweden
first_name.last_name@dis.uu.se

Abstract. Data integration on a large scale poses complexity and performance problems. To alleviate the complexity problem we use a modular approach where many heterogeneous and distributed data sources are integrated through composable mediators. Distributed mediators are defined as object-oriented (OO) views defined in terms of views in other sub-mediators or data sources. In order to minimize the performance penalty of the modular approach we have developed a distributed expansion strategy for OO views where view definitions are selectively imported from sub-mediators. Our performance analysis has shown that the distributed view expansion can support modularity through distributed and composable mediators with little overhead.

1 Introduction

There has been substantial interest in using the mediator/wrapper approach for integrating heterogeneous data [9, 21, 7, 19, 4]. Most mediator systems integrate data through a single mediator server accessing one or several data sources through a number of 'wrapper' interfaces that translate data to a common data model (CDM). However, one of the original goals for mediator architectures [22] was that each mediator should be a relatively simple modular abstraction of the integration of some particular kind of data. Larger systems of mediators would then be defined through these primitive mediators by composing new mediators in terms of other mediators and data sources. Different mediator servers distributed on the network would define different logical views of data. Such a modular logical composition of mediators allows to overcome complexity problems of data integration on a large scale with many data sources and mediators involved. However, very few projects have used a distributed mediator architecture, e.g. [15], and there is little work on implementation issues of distributed mediators.

This paper investigates query processing in a distributed mediator system, AMOS II [20], where distributed mediators are composed as object-oriented (OO) views in terms of views in other sub-mediators or data sources. The views make the distributed mediators appear to the user as a single virtual database consisting of a number of types (classes) and functions (methods, attributes). However, unlike regular OO systems the extents of these types and functions

are not explicitly stored in a database but are derived, through an OO multi-database query language, from data in the underlying data sources and other OO mediators [5, 10, 12]. Even though such an architecture addresses the complexity problems of data integration it also has some performance problems. Unlike distributed databases, the distributed mediators do not have any central schema and each mediator server has only limited knowledge about the structure of other mediators. This makes it difficult to find an optimal distributed query execution plan. In our approach the distributed mediator servers communicate with other known mediator servers to import some of the schema information, such as some OO view definitions. Exchanging OO view definitions in a composable mediator framework poses new problems compared to relational distributed databases as information about unknown user-defined types has to pass through intermediate servers. In order to deal with this problem we have developed a technique to process queries when incomplete type information is available, to be described in an upcoming work.

In [11] we described how to decompose distributed queries and then rebalance the decomposed query execution plans to minimize the communication overhead by generating an optimized data flow pattern between the distributed mediator servers. In that strategy the sub-mediators did not export their view definitions but only executed queries and provided query costing information. Such a strategy can be suboptimal when there are more than two distributed mediator layers. In this paper the importance is analyzed of a method based on distributed selective view expansion (DSVE) to minimize the penalty of several mediator server levels. The combination of DSVE, query decomposition and rebalancing is shown to significantly improve query performance. The method can drastically reduce query execution time when information from several hidden sub-mediators can be combined. A performance study of this case shows execution time improvement between 20 and 144 times for a test query selectivity varying between 1 and 0.01 [13]. Our measurements also show that savings in time are achieved in every component of the mediator composition (mediators, network, datasources). The performance improvements are due to more selective queries, smaller data flows between the servers, and fewer servers involved in the data exchange.

As our research platform we use the AMOS II mediator database system [20]. The core of AMOS II is an extensible and distributed main-memory OO DBMS. For more details about the architecture, query language and data integration capabilities the interested reader is referred to [5, 10, 12, 20]

2 Distributed Selective View Expansion

The following two subsections first describe the mechanism for view definition exchange and expansion in a hierarchy of AMOS II servers. After that a new heuristic based approach to selectively perform view expansion is proposed. Due to space limitations only an outline of the algorithms is given, while a detailed description can be found in [13].

2.1 Basic View Expansion Algorithm

Distributed view expansion is implemented as an extension to the query processor of AMOS II. Figure 1 illustrates the extended query processing with distributed view expansion. The view expansion is placed after the original query has been decomposed into distributed subqueries over the participating mediators. This improves query compilation time by minimizing the sizes of the query expressions. The grouped predicates are compiled and rewritten together at the coordinating mediator. In the distributed view expansion phase (the grayshaded boxes in Figure 1) the client mediator sends a view expansion request to each server where a subquery is to be executed and an expanded view definition of the subquery is retrieved. The subqueries are communicated between the servers in the form of declarative expressions [14]. At the server accepting the subquery expansion request, the subquery processing starts with query transformations and continues in the same manner as with other queries until the distributed selective view expansion phase. In this phase subqueries that contain themselves subqueries to other AMOS II mediators are selectively expanded. The process might span several levels of mediators, and, it terminates according to the strategy described in the next subsection.

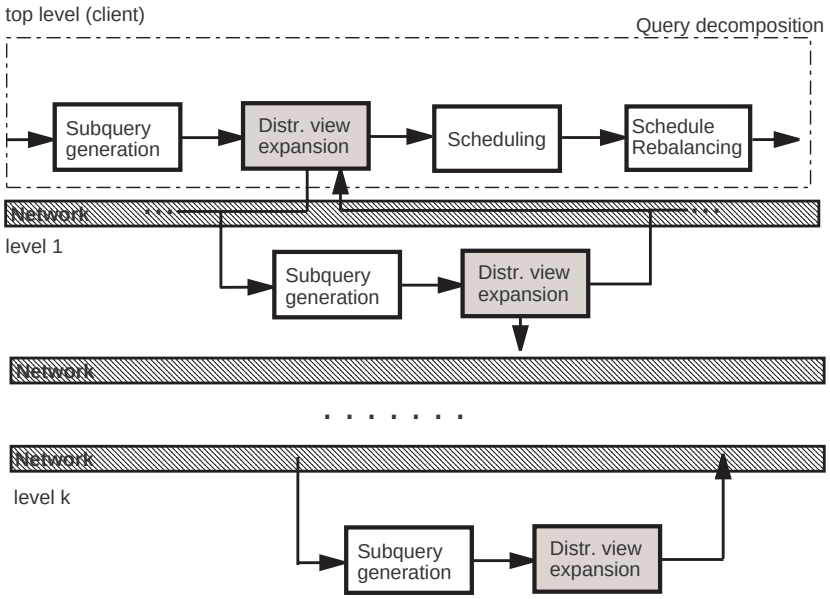

Fig. 1. Distributed selective view expansion process

After collecting the definitions of the remote subqueries selected for expansion, the next phase in the DSVE combines again all these expressions into

one in order to be reoptimized by the subsequent optimization phases. Next a query scheduler generates distributed execution plan. In a final step this plan is rebalanced by a distributed algorithm [11].

The benefits of the proposed view expansion are the following:

1. Calculus based rewrites can be performed at the client to eliminate overlap in the calculation among the different servers.
2. Each subquery to be executed at another AMOS II server could be expanded into expressions derived from multiple AMOS II servers. These expressions might in turn have sub-expressions that are executed at a common AMOS II server. Putting them together in a single predicate may increase it's selectivity and allows for optimizations that eliminate overlap or achieve a better execution strategy.
3. The expanded expressions returned from the DSVE may contain predicates that could be combined/replicated with predicates at different data sources where they can act as selections, reducing the query execution time and the intermediate result sizes.
4. A richer space of data flow patterns will be considered by the query scheduler.

2.2 Controlling the view expansion process

While the view expansion can eliminate redundant computation and data shipment, it may also introduce extra costs in the optimization. Therefore we turn our attention to the problem of deciding when to perform expansion of the views and how to control the depth of DSVE.

An exhaustive cost-based algorithm determining which views to expand would need to fully compile the query for each combination of the views at the intermediate nodes, thus resulting in prohibitively long compilation times. To reduce the complexity of this problem each node makes a decision if a view is to be expanded based only on locally available data - the number of sub-subqueries of each subquery.

Whenever a multidatabase (sub)query is to be processed by the query decomposer, before DSVE is performed, a budget based decision procedure is used to control the depth of view expansion. The main goal of this procedure is to favor view expansion in cases of deep mediator hierarchies, and prohibit view expansion explosion when the number of direct sub-mediators is too large.

The query site starts with some initial budget. Whenever a view expansion request is to be sent, part of the budget is sent along with that request. The receiving site starts it's compilation process with the received part of the initial budget and proceeds in the same manner. The view expansion process stops when all of the initial budget is distributed among some of the participating mediators.

3 Related Work

This work is related to work on query optimization in distributed databases and mediators. Distributed databases [18, 3, 8, 1] have complete global schemas de-

scribing on what sites different (fractions of) tables are located, while distributed mediators do not have complete knowledge of meta-data from all mediators and data sources. Full expansion of all possible views in a distributed system with many nodes may be very costly. By contrast selective view expansion allows compositions of very many servers. In [16] a view expansion strategy for the System R* distributed database is briefly mentioned but not evaluated.

Mediator systems are usually not distributed (e.g. [9, 21, 17]) and thus do not use our strategies. In [4] it is indicated that a distributed mediation framework is a promising research direction without reporting any results. The DIOM system [19] is also a distributed mediator system using distributed query scheduling similar to our decomposer. However, no distributed view expansion is reported.

To our knowledge no work has addressed problems related to the use of OO features and more specifically user-defined types in a distributed mediator system.

4 Summary and Future Work

We described a distributed view expansion technique in composable mediators. Our results show that even in a simple mediator composition this approach leads to significant performance improvements, compared to a "black-box" approach to distributed query optimization. The main contribution of this work is that it shows that OO mediators may be logically composed to solve integration problems with very little execution overhead.

Though not addressed in this paper, we have also investigated performance problems in more complex composition scenarios. Preliminary results show that DSVE will be beneficial to use to discover optimal data flow in a non-homogeneous network with different communication speeds between the mediator nodes.

An issue not addressed in our current work is compilation time of large distributed queries. Our current experience shows that while mediator compositions of less than 10 mediators works reasonable with our framework, larger systems of distributed mediators require scalable and distributed compilation techniques.

References

1. P. Apers, A. Hevner and S. Yao: Optimization Algorithms for Distributed Queries. *IEEE Transactions on Software Engineering*, 9(1), 1983
2. O. Bukhres, A. Elmagarmid (eds.): *Object-Oriented Multidatabase Systems*, Pretince Hall, 1996.
3. D. Daniels et al.: An Introduction to Distributed Query Compilation in R*. In H. Schneider (ed): *Distributed Data Bases*, North-Holland, 1982
4. W. Du and M. Shan: Query Processing in Pegasus, In O. Bukhres, A. Elmagarmid (eds.): *Object-Oriented Multidatabase Systems*, Pretince Hall, Englewood Cliffs, 1996.
5. G. Fahl, T. Risch: Query Processing over Object Views of Relational Data. *The VLDB Journal*, Springer, 6(4), November 1997.
6. S. Flodin, T. Risch: Processing Object-Oriented Queries with Invertible Late Bound Functions, *21st Conf. on Very Large Databases (VLDB'95)*, Zurich, Switzerland, 1995

7. H.Garcia-Molina, et al: The TSIMMIS Approach to Mediation: Data Models and Languages. *Intelligent Information Systems (JIIS)*, Kluwer, 8(2), 1997
8. N. Goodman, P. Bernstein, E. Wong, C. Reeve and J. Rothnie: Query Processing in SDD-1: A System for Distributed Databases. *ACM Transactions on Database Systems (TODS)*, 6(4), 1981
9. L. Haas, D. Kossmann, E. Wimmers, J. Yang: Optimizing Queries across Diverse Data Sources. *23th Intl. Conf. on Very Large Databases (VLDB'97)*, Athens, Greece, 1997
10. V.Josifovski and T.Risch: Functional Query Optimization over Object-Oriented Views for Data Integration, *Intelligent Information Systems (JIIS)* Vol. 12, No. 2/3, Kluwer, 1999.
11. V.Josifovski, T.Katchaounov, T.Risch: *Optimizing Queries in Distributed and Composable Mediators*, *4th Conference on Cooperative Information Systems*, CoopIS'99, Edinburgh, Scotland, Sept. 1999.
12. V.Josifovski, T.Risch: Integrating Heterogeneous Overlapping Databases through Object-Oriented Transformations, *25th Conf. on Very Large Databases (VLDB'99)*, Edinburgh, Scotland, Sept. 1999.
13. T. Katchaounov, V. Josifovski, T. Risch: Distributed View Expansion in Composable Mediators, *Research Report 2000:2*, Uppsala University, Department of Information Science, 2000.
14. W. Litwin and T. Risch: Main Memory Oriented Optimization of OO Queries using Typed Datalog with Foreign Predicates. *IEEE Transactions on Knowledge and Data Engineering*, 4(6), 1992
15. L.Liu, C.Pu: An Adaptive Object-Oriented Approach to Integration and Access of Heterogeneous Information Sources, *Distributed and Parallel Databases*, Kluwer, 5(2), April 1997.
16. G.Lohman, C.Mohan, L.Haas, D.Daniels, B.Lindsay: Query Procesing in R*, in W.King, D.S.Reiner, D.S.Batory (eds.): *Query Processing in Database Systems*, Springer Verlag, 1985.
17. S. Nural, P. Koksal, F. Ozcan, A. Dogac: Query Decomposition and Processing in Multidatabase Systems. *OODBMS Symposium of the European Joint Conference on Engineering Systems Design and Analysis*, Montpellier, July 1996.
18. M.T.Özsu, P.Valduriez: *Principles of Distributed Database Systems*, Prentice Hall, 1999.
19. K.Richine: *Distributed Query Scheduling in DIOM*, Tech. report TR97-03, Computer Science Dept., University of Alberta, 1997.
20. T.Risch, V.Josifovski, T. Katchaounov: *AMOS II Concepts*, available at *http://www.dis.uu.se/~udbl/amos/doc/*, 2000.
21. A. Tomasic, L. Raschid, P. Valduriez: Scaling Access to Heterogeneous Data Sources with DISCO. *IEEE Transactions on Knowledge and Date Engineering*, 10(5), 1998
22. G. Wiederhold: Mediators in the Architecture of Future Information Systems, *IEEE Computer*, 25(3), Mar. 1992.