Query Processing System for Web Data Sources (Working Notes)

Iztok Savnik
Institut für informatik
Albert-Ludwigs-Universität Freiburg
Am Flughafen 17, 79110 Freiburg, Germany
savnik@informatik.uni-freiburg.de

July 2, 2000

Abstract

This report present some aspects of the design and implementation of the system for querying and integration of data from the Web data sources called `Qios` which is currently under development.

1 Introduction

The data sources on Web will include large amounts of various data such as for example the catalogs describing companies and their products. Querying and analysis of the data from Web data sources will require the capabilities to manipulate information represented in various data models, to integrate information from the different data sources, and, to process efficiently the queries. Moreover, the system needs to provide the data store with the facilities for organizing and browsing the results of querying and integration sessions.

The representation language used as the basis of the system for querying and the integration needs to be expressive enough to model the existing data representation formats. Primarily, we assume that the data is available in HTML or XML data representation formats accessible as Web files. However, because of the fast evolving technology we may expect other data models as well as the interfaces to the database management systems. Furthermore, the data model of the system must also serve as the basis for the representation and reasoning about the data collected from the data sources. The query processing facility of the system have to be efficient enough to process the queries on larger amounts of data in a reasonable response time. To provide fast query processing we use the existing techniques developed for relational and object-relational database management systems and adapt them for the new environment.

2 Architecture of system

The system for querying Web data sources have to provide flexible and easily extensible interface towards the heterogeneous data sources. Since many of the popular data representation formats provide the functionality to describe the same data in the numerous different forms, the interface must include the detailed descriptions of the data format (within the particular representation language) used by the data sources. For instance, the structure of the data available as HTML files have to be specified in order to be able to correctly interpret the data.

The query processing engine of the system for querying and integration of data from Web sources needs to fulfill number of requirements among which some of them are conflicting. The query processing engine have to be based on the expressive data model in order to be able to represent various formats as well as to be able to serve as the knowledge representation language for structuring and browsing the collected data. Further, the system have to be able to process larger amounts of data fast, therefore it has to include the functionality of the traditional database systems. Finally, the query processing engine has to be able to process queries on the intensional (metadata) databases.

The data model and the storage system have to serve as the basis of the knowledge representation system which allows a user to browse thru the data gathered in the previous sessions, and not just processing the queries. A collection of objects and classes obtained from the data sources and organized using a conceptual schema can be referred to as *information space*. The problems here include the design and the implementation of the knowledge framework which can serve as the basis for the organization of the gathered data, the user-interface providing the means for navigating thru the information space, as well as the reasoning facilities which can be used to derive (or, discovery) the relationships among the elements of the gathered data.

Finally, the Web data sources are distributed and the connection between the sources can vary significantly. The distribution of the data among the data sources depends on the organization of the data about the particular subject of user's interest. We can not assume the predefined type of the distribution which could help in the design of the query processor. Caching data from the Web sources and/or the query results, and the problem of the temporal inaccessibility of the data sources have to be considered.

2.1 Data model

The system is intended primarily for querying the existing data provided by the Web data sources. Therefore, it has to rely on the data modeling constructs defined by the data sources. The main representation languages used to represent the data from the Web data sources are XML and HTML.

The data model provided by `Qios` is based on the structural part of F-Logic. The objects stored by `Qios` storage manager are either classes or individual objects. Classes can include, besides the types, also the class properties which describe the properties of the classes them-self. Classes are in `Qios` treated in a similar manner as individual objects. We expect that the intensional databases will be as important as the extensional database for querying Web data sources. The uniform view of the intensional and extensional databases is discussed in [19].

The *basic* types of `Qios` are `string`, `number`, and the *object type*. The domain of a variable of object type is the extension of a class which name serves as the type identifier. The *constructed* type of `Qios` provides usual object abstraction by which the objects can be described by a number of either single valued or multivalued attributes (functions).

The identifiers of objects are in many cases defined by the data sources. For this reason, the external object identifiers are used in `Qios`. An external object identifier is defined by the identifier of the data source, the identifier of the database, and the identifier of the object in the frame of the database. All modules of the query processing engine manipulate the external identifiers.

2.2 System for querying and integration

`Qios` is a light-ware query processing system providing efficient manipulation of the data available from the Web data sources. The system is designed to serve as the tool for the integration of the information from Web data sources, efficient processing of queries posed on distributed data sources including larger volumes of data, and, also as the tool for organizing and browsing the data and knowledge collected from the data sources.

The system is designed to provide flexible and uniform interface towards the Web data sources. The interface to Web is based on the abstract access methods.

The system can be easily upgraded to provide the access to the data sources which use a novel representation language. The interface is constraint to operate on the collections of objects which is the primary data structure used for the organization of data sources.

The kernel of query processing system is based on the algebra of objects [21]. It is defined to be flexible in the sense that it allows the extensibility of the system with the new operations and the query transformation rules. The currently implemented operations are the relational operations `select`, `project`, `join` and the operation `copy`.

3 Implementation

The system for querying and the integration of the data from the Web data sources `Qios` is implemented using Perl programming environment. The access to the Web data sources and the manipulation of HTML and XML documents is realized using the Perl modules `libnet`, `libwww`, `XML-Parse` and `HTML-Parse` modules available from CPAN repository [5]. In this section we overview the implementation of the main modules of `Qios`.

3.1 Storage manager

The storage manager is based on the structural part of F-Logic [14]. Object is an entity which has an identity and the number of functions (attributes) which can be either single-valued or multi-valued. Only the complete objects can be manipulated and the identifiers are seen as the names of objects. The storage manager implements objects and their functions as in the database systems – they are represented as tuples. The multivalued attributes are represented with lists.

The class objects are represented in the same manner as the ordinary objects as regards the implementation of the identifiers and the attributes [19]. However, their implementation includes only few differences as regards the involvement of objects in the ISA partially ordered set of objects. The class extension is realized as in the hierarchical database model by using the double linked lists which include all members of a class and its subclasses.

In general, objects can be accessed either directly by object identifiers, or using access methods. The access to the objects using oids is realized using the associative array. The currently available access methods for class objects are sequential scan and index scan thru class members or class instances [19]. The hash-based indexes [17] are used for the implementation of the index scan. The access methods are realized as iterators.

3.2 Optimizer

The main structure of the optimizer based on the data structure Mesh [10, 11]. The optimizer organizes queries into the equivalence classes. Each equivalence class is realized as an object which includes reference to the best query of the class, the price of this query and the list of queries belonging to the equivalence class. The equivalence classes are indexed by the normalized query expressions represented as text strings. Finally, each query includes a reference to the equivalence class to which it belongs.

The query optimizer is based on query rewrite rules that can be dynamically read or deleted to/from the system. The optimization algorithm is a version of the dynamic programming algorithm for the query optimization [10]. It optimizes the queries in the top-down manner: the sub-queries are optimized first starting with the left-deep queries and at the end the alternatives of the complete query are evaluated. The algorithm provides possibilities to control the portion of the search space that is investigated by the optimizer.

3.3 Query evaluation

The queries are interpreted as iterator trees. The query executor does not include the separate physical optimization phase since most of the work (in comparison to the relational and object-relational query optimizers) involved in the optimization of physical operations is realized as the part of the previously presented optimizer. The main reason for this is the small number of alternative physical operations used by the system. The choice of the physical operation used for query execution is defined during the creation of the iterator tree of a query.

The selection of the access methods for the (materialized) classes includes only sequential scan and index scan based on an attribute. The join can be realized using either the nested-loops or index nested-loops algorithms. The right-deep branches of query tree are handled in each case by the materialization of the result and then using the index based access in the case that the hash-based index can be applied. Similarly, the select operation can use the index when the predicate has appropriate structure.

3.4 Information spaces

To provide the environment for the interactive querying and integration of the data from Web sources the concept of the *information space* is used [7]. The information space is defined by the set of interrelated classes and their instances. Physically it is realized as a class including the name space, that is, the oids and the addresses of all objects which have external oids. Objects can include the references to objects from other information spaces. All objects created during the interactive sessions are generated within the current information space.

4 Conclusions

The system for querying and the integration of information from the Web data sources is presented in this document. The first part of the document presents the architecture and some of the reasons for the decisions in the design of the complete system. The second part of the report describes the main aspects of the implementation.

The following work on the implementation of `Qios` needs to be completed. First, the user interface of the system will be improved to provide a user-friendly interactive environment for querying and integration. Next, while the optimization of the queries for the distributed servers is completed, there is more work on the implementation of the query evaluation system. Finally, the error reporting and (internal) error recovery mechanisms have to be completed.

References

[1] S. Abiteboul, C. Beeri, 'On the Power of the Languages For the Manipulation of Complex Objects', Verso Report No.4, INRIA, France, Dec. 1993.

[2] C. Beeri, Y. Tzaban, 'SAL: An algebra for semistructured data and XML', Informal Proc. of Workshop on The Web and Databases, ACM SIGMOD, June 1999.

[3] R.J. Brachman, J.G. Schmolze, 'An Overview of the KL-ONE Knowledge Representation System', Cognitive Science, Vol.9, No.2, 1985

[4] P. Buneman, R.E. Frankel, 'FQL- A Functional Query Language', Proc. of the ACM Conf. on Management of Data, ACM SIGMOD, 1979.

[5] *CPAN: Comprehensive Perl Archive Network*, [http://www.perl.com/CPAN/], Jan 2000.

[6] D. Daniels, P. Selinger, L. Haas, B. Lindsay, C. Mohan, A. Walker, P. Wilms, 'An introduction to distributed query compilation in R*', IBM Research Report RJ3497 (41354), June 1982.

[7] R. Davis, W. Neviett, M.A. Foltz, *Information Architecture*, [http://www.infoarch.ai.mit.edu/], Apr 2000.

[8] J.C. Freytag, 'A rule-based view of query optimization', Proc. of ACM Conf. on Management of data, 1987.

[9] G. Graefe, 'Query Evaluation Techniques for Large Databases', *ACM Comp. Surveys*, Vol.25, No.2, June 1993, pp. 73-170.

[10] G. Graefe, R.L. Cole, D.L. Davidson, W.J. McKenna, R.H. Wolniewicz, 'Extensible query optimization and parallel execution in Volcano', In Freytag, Vossen, Maier (eds.), *Query processing for advanced database applications*, Morgan-Kaufman, 1994, p.305.

[11] G. Graefe, W. McKenna, 'The Volcano optimizer generator: extensibility and efficient search', Proc. of IEEE Conf. on Data Engineering, April 1993, p.209.

[12] Y.E. Ioannidis, Y.C. Kang, 'Leeft-deep vs. bushy trees: an analysis of strategy spaces and its implications for query optimization', Proc. of the ACM Conf. on Management of Data, 1991, pp.168-177.

[13] M. Jarke, J. Koch, 'Query optimization in database systems', *ACM Comp. Surveys*, Vol.16, No.2, June 1984.

M. Kifer, W. Kim, Y. Sagiv, 'Querying Object-Oriented Databases', Proc. of the ACM Conf. on Management of Data, 1992.

[14] M. Kifer, G. Lausen, J. Wu, 'Logical Foundations of Object-Oriented and Frame-Based Languages', *Journal of ACM*, Vol.42, No.4, July 1995, pp.741-843.

[15] B. Ludëscher, R. Himmeröder, G. Lausen, W. May, and C. Schlepphorst, 'Managing Semistructured Data with FLORID: A Deductive Object-Oriented Perspective', *Information Systems*, 23 (8), Special Issue on Semistructured Data, 1998, pp. 589-612.

[16] G.A. Mitchell, ‘Extensible Query Processing in an Object-Oriented Database’, Ph.D. thesis, Brown University, 1993.

[17] R. Ramakrishna, J. Gehrke, *Database Management Systems*, McGraw-Hill, 2000.

[18] I. Savnik, Z. Tari, T. Mohoric, ‘Using schema information for querying databases’, Working Notes of KRDB’95 Workshop: Knowledge Representation Meets Databases, Bielefeld, 1995.

[19] I. Savnik, T. Mohoric, Z. Tari, ’Unifying Schema and Instance Levels of Object-Oriented Databases’, Technical Report, Faculty of computer and information sciences, University of Ljubljana, April, 1997.

[20] I. Savnik, ‘Algebra for distributed data sources’, Dagstuhl Seminar ”Multimedia Database Support for Digital Libraries”, Schloss Dagstuhl, Germany, Aug 1999.

[21] I. Savnik, ‘Algebra for Distributed Data Sources on Web’, To appear in *Proc. of Int. Conf. on Flexible query answering systems*, Springer, 2000.

[22] A. Tomasic, L. Raschid, P. Valduriez, ‘Scaling Heterogeneous Databases and the Design of Disco’, Technical Report RR-2704, Inria, 1995.

[23] S.L. Vandenberg, ‘Algebras for Object-Oriented Query Languages’, Ph.D. thesis, Technical Report No. 1161, University of Wisconsin, July 1993.

[24] M. Stonebraker, L.A. Rowe, M. Hirohama, ’The Implementation of Postgres’, IEEE Transactions on Knowledge and Data Engineering, March 1990, vol.2, (no.1):125-42.

[25] Z. Tari, W. Cheng, K. Yetongnon, I. Savnik, ’Towards Cooperative Databases: The Distributed Object Kernel Approach’, Proc. of Parallel and Distributed Computing Systems Conf., Dijon, France, 1996.