Query processing system for XML

Iztok Savnik
Faculty of Mathematics, Science and Information Systems, University of Primorska

Abstract

The paper presents the implementation of query execution system Qios. It serves as a lightware system for the manipulation of XML data. Qios employs the relational technology for query processing. The main aim in the implementation is to provide a querying system that is easy to use and does not require any additional knowledge about the internal representation of data. The system provides robust and simple solutions for many design problems. We aimed to simplify the internal structures of query processors rooted in the design of relational and object-relational query processors. We propose efficient internal data structures for the representation of queries during all phases of query execution. The query optimization is based on dynamic programming and uses beam search to reduce the time complexity. The data structure for storing queries provides efficient representation of queries during the optimization process and the simple means to explore plan caching. Finally, main memory indices can be created on-the-fly to support the evaluation of queries.

Keywords

Database algebras, query languages, distributed databases, query optimization, Web query languages.

1 Introduction

The Internet contains large amount of different data sources accessible through ftp files, XML or HTML documents, and the wrappers around relational and object-relational database systems. The data available via such data sources may vary from simple lists of records, catalogs containing large amounts of flat tables, to complex data repositories including large conceptual schemata and tables composed of complex objects. Querying Web data sources poses several new problems such as uniform access to the data sources, integration of information provided by the data sources, and efficient manipulation of large data sets.

Our work focuses on the design of a robust and flexible query execution system for querying and integration of data from Web data sources. The design of query execution system Qios (Savnik, 2007) is based on the existing work on relational and object-relational query execution systems (Daniels, 1982; Graefe, 1993). We investigated the efficient and robust design of the internal structures of the query execution engine. In particular, we investigate the following implementation aspects of a query processing engine: query optimization for lightware XML databases including up to hundreds of thousands of objects, dynamic selection of indexes during the query evaluation, and efficient internal data structures of a query processing system.

2 Managing Information Spaces

Work on specific problems on the Web requires gathering and restructuring various types of data. For instance, creation of an overview of a specific scientific topic requires gathering the publications, references to Web pages, titles of books, as well as organizing the information about the topic. Similar examples might appear when user browse the Web in order to collect information about the products available on the market, business information about companies, information about a hobby, and the like.

The data gathered about the particular subject of interest forms the information space (Bryce, 1998). It comprises the local data loaded from Web data sources as well as the data stored at the data sources of interest. The information space forms a conceptual framework for inquiring about the topic of interest. To be able to create and manipulate an information space, the system for querying Web data sources has to provide the facilities for: exploring the schemata of data sources, loading data from data sources, the integration of the information from the data sources, and managing local conceptual schema of information gathered for the particular topic.

The data repository of the query processing system Qios is organized into databases which serve as working environments for collecting, browsing, querying and integration of XML data. The conceptual schemata of a database comprises a set of classes that can be manipulated by means of data model operations. The extension of classes include ground objects that can be queried in a similar manner to SQL databases. The queries can be expressed as the object algebra expressions (Savnik, 1999) that form a robust and easily understandable functional query language. The language is relationally complete: classes can be filtered, projected, joined and restructured. The algebraic operations allow for querying and integration of structured as well as semi-structured data.

The Qios console interface is used for issuing queries, browsing the databases, storing query results in Unix files and for the manipulation data residing in a file system. It is based on Unix style commands offering the complete access to the operating system facilities and hence a reliable and simple tool for the manipulation of files and other resources used during a session. Other interfaces to query processing kernel are allowed through the Qios programming language interface as well as through the server wrapper.

3 Implementation

Qios serves as the lightware kernel of a data manipulation server. The main aims in the design of Qios were to provide: capabilities to manipulate collections of data in a fast manner, various data manipulation functions from classical querying to data restructuring, and, the capabilities to organize, store and browse the data collections obtained from the Internet data sources on the local host. The system currently provides the interface for XML. The treatment of other data formats requires the addition of the interface routines for the conversion of data into an internal database format.

We investigated the efficient and robust design of the internal structures of the query execution engine. The algorithm used for the optimization is based on graph representation of query trees and query transformation rules. Query transformation is seen as a matching rule input tree against a query tree and than duplicating the rule output tree. Similarly, a query evaluation module uses the same structure for the implementation of scans. Finally, to pro-

vide efficient implementation of query optimization and evaluation queries are stored in a data structure called Mesh (Graefe, 1993). The data structure is refined to provide efficient access to the stored queries.

The query optimization algorithm is based on a version of dynamic programming called memoisation. The most promising results were obtained with an optimization algorithm which uses beam search for the exploration of the hypothesis space of equivalent queries. Further, we explore plan caching (Graefe, 2004) which speeds up significantly the subsequent execution of queries issued on the same domain.

The query evaluation is based on dynamic selection of the query evaluation plans. We use a simple strategy that exploits the large quantity of main memory which is lately provided by almost any personal computer. The main memory indices can be constructed on-the-fly to support query evaluation. The construction of indices is based on standard index selection rules available from any database textbook. For instance, hash-based index is used whenever a larger table has to be joined. In this way we achieve fast performance of a query processor for relatively large quantities of data. Furthermore, the user does not need to concern about the details of the query evaluation process.

4 Related work

Our work on object algebra has been influenced by the early functional query language FQL proposed by Buneman and Frankel (Buneman, 1979) and by some of its descendants, for instance, the functional database programming language FAD (Danforth, 1992). Algebra is by its nature a functional language, where the operations can be combined by the use of functional compositions and higher-order functions to form the language expressions. The presented object algebra can be treated as a generalization of FQL for the manipulation of objects. It subsumes the operations of FQL, i.e. operations extension, restriction, selection and composition.

Let us now present the work related to the implementation of the presented query execution system. Firstly, the implementation is closely related to the implementation of Query Algebra originally proposed by Shaw and Zdonik in (Shaw, 1990) and implemented by Mitchell (Mitchell,1993). In particular, we have used a similar representation of query expressions by means of query trees. Furthermore, the representation of query expressions in Qios is optimized by using single operation nodes and query trees during all phases of query processing.

The design of the query execution system was based on the design of the Exodus optimizer generator and its descendant Volcano (Graefe, 1993). The data structure MESH used in the Exodus query optimizator generator is improved by adding additional access paths. The data structure can be accessed through: unique identifier, normalized query expression, and equivalence class. The algorithm for query optimization is rooted in Graefe's work on the Volcano optimizer algorithm (Graefe, 1993). This algorithm uses top-down search guided by the possible „moves“ that are associated to a query node. The algorithm uses memorization to avoid repeated optimization of the same query. The search is restricted by the cost limit which is a parameter in optimization.

We implemented a lightware system able to manipulate middle size XML databases including up to some 100.000 records. This size is reasonable for most of the collections appearing on the Web as well as in our local data environments. One of the aims in the design

of Qios was to exploit this advantage. The design of Qios has the following salient features. A single data structure is used during the complete optimization and evaluation process. The central data structure of the query optimization Mesh is robust and simple to use. Rules are treated as query trees and rule matching and application procedures are based on graph (tree) algorithms.

The computation of a query execution plan that uses indices created during query evaluation has similar objectives: it is expected that the XML database will be of the above stated size and that the user is not informed about the existence of indices and the selected access paths during the execution. All decisions about the creation of main memory indices are done by the system.

5 Conclusions

The paper presents the implementation of an object algebra. The data model based on F-Logic provides a convenient environment for the representation of semi-structured as well as structured data. Algebra includes standard operations on sets which evolved from the relational and nested-relational algebras and the operations for querying the conceptual schemata. Object algebra is implemented in the query execution system Qios which is rooted in the architecture of the relational and object-relational query processors.

Bibliography

Bryce Allen, `Information space representation in interactive systems: relationship to spatial abilities', International Conference on Digital Libraries, 1998.

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

S.Danforth, P.Valduriez, `A FAD for Data Intensive Applications', IEEE Trans. on Know. and Data Eng., Vol.4, No.1, Feb. 1992

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.

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

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

G.Graefe, `Query processing', Slides, ICDE Influential Paper Award, 2005.

G.A.Mitchell, `Extensible Query Processing in an Object-Oriented Database', Ph.D. thesis, Brown University, 1993.

I.Savnik, `Algebra for distributed data sources', Dagstuhl Seminar „Multimedia Database Support for Digital Libraries", Schloss Dagstuhl, Germany, Aug 1999.

I.Savnik, Z.Tari, T.Mohoriè, `QAL: A Query Algebra of Complex Objects', Data & Knowledge Eng. Journal, North-Holland, Vol.30, No.1, 1999, pp.57-94.

I.Savnik, Z. Tari, `QIOS: Querying and Integration of Internet Data', http://www.famnit.upr.si/~savnik/qios/, FAMNIT, April 2007.

G.M.Shaw, S.B.Zdonik, `A Query Algebra for Object Oriented Databases', Proc. of IEEE Conf. on Data Engineering, 1990, pp.154-162.

Contact Information

Iztok Savnik
Faculty of Mathematics, Science and Information Systems
University of Primorska
Glagoljaška 8
5000 Koper, Slovenia