Query Optimization for Inter Document Relationships in XML Structured Document

Radha Senthilkumar, A. Kannan, D.Vimala, M. Bhuvaneswari
Department of Information Technology, MIT, Anna University, Chennai, India

The information published in the form of XML-complaint keeps growing. They are mainly used in web-based applications and have many simultaneous interactive users. Hence there should be an efficient and effective way to optimize the XML queries. Query optimization in the context of XML database is extremely challenging as compared with the other data models. The existing optimization algorithms focus only on the containment relationships with in a XML document. This paper explains the optimization of XPath for structured Inter-documents, which make use of referencing relationship among the different XML documents. The Inter-document concept is helpful in cases where it is beneficial to have several documents. A new method of schema level query optimization for structured Inter- document is being proposed herein. The results demonstrate better scalability, selectivity and reduction in execution time.

1. Introduction

Query optimization in XML is becoming more important as the use of XML in web applications increases rapidly. Optimization in the context of XML database is extremely challenging due to its high complexity, as compared with other data models. This high complexity renders a much enlarged search space for XML query optimization. The classical cost-based and heuristic-based approaches yield unacceptably low efficiency when applied to XML data. Much of the work on XML query optimization has been done by providing efficient supporting algorithms and indexing schemes [10]. In the earlier work [7], query optimization is done for containment relationship with in a single XML structured document. However, such an optimization gives rise to issues when multiple interrelated tables are used. In order to do query optimization it is necessary to identify the attributes of the each table, and the relationship between the attributes of the different tables. It becomes complex to handle such Inter-document relationships. In this paper, we explain the optimization of XPath queries for Inter-documents. Various XML properties are identified from schema files given as input. A comprehensive set of general equivalences with regard to XML documents and XML queries are developed. Based on these equivalences, a large set of deterministic algebraic transformation rules are developed for XML query optimization. These rules are applied based on the identified properties to obtain the optimized XPath expression [12]. The efficiency of the optimized queries is tested in a relational database [2][8].

2. Related work

Dunren [7] have proposed an Xpath with containment operators, which are optimized by performing deterministic algebraic transformation. This article extends the approach in [7] to be suitable for both containment and reference operators. Lore [13] is a DBMS originally designed for semi structured data and later migrated to XML based data model. This paper performs optimization at the logical level by applying deterministic algebraic transformation rules. In [9], a comparable strategy for exploiting a grammar specification for optimizing queries on semi-structured data is studied. In [9], efforts were made on how to make complete use of the available grammar to expand a given query. This article identifies transformation that introduces “guaranteed” improvements on query expressions from a heuristic point of view. In [4], path constraints were used to convey the semantics of semi structured data and applied to query optimization.

0-7695-3050-8/07 $25.00 © 2007 IEEE
DOI 10.1109/ICCIMA.2007.363

However, certain implicit knowledge about the structural connections within XML data like exclusivity, obligation and entrance location is of greater value for XML query optimization than those considered in [4]. In [1], the authors studied the minimization issue of general tree pattern queries, and both constraint dependent and constraint independent minimizations were addressed. However this article uses exclusivity and entrance location and considers the relationship between the tables for optimization, which was not there in [1]. In Wang [16], the notion of path is a rooted one. Our approach is more general and can identify more opportunities for reducing a path, since it does not need to be rooted. It explores other aspects of applying structure knowledge of XML data and related heuristics for query optimization.

3. The Proposed Optimization Techniques

The overall flow of steps involved in the optimization of Intra Document query processing [18] [19] is shown in figure 1. The optimization starts with the conversion of XPath to a PAT expression. The conversion is followed by three level of optimization, namely Normalization, Semantic Transformation and Simplification. The operations carried out in the above mentioned levels are listed in the Figure 1. The Semantic Transformation phase uses the three structural properties, Exclusivity, Obligation, Entrance Location [5] [3], which are extracted from the schema. The output of the three levels is the optimized PAT expression. This PAT expression is converted into an SQL statement for executing on a RDBMS.

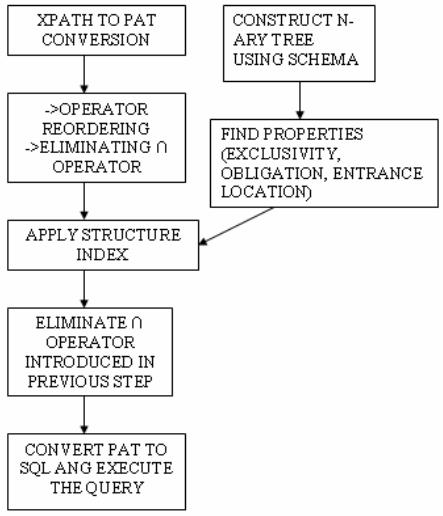

Figure 1. Overall Flow of Optimization

3.1 Tree Construction

Schema defines the structure of the document and the elements within it. An n-ary tree is constructed for each table in the document [6]. Each node in the tree represents the elements in the table and the edges give the structural relationship that exists among the nodes.

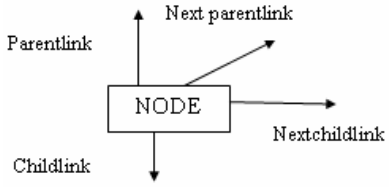

Figure.2. Node Representation for an element in XML document

The node structure is given in Figure 2. Each node has four pointers. First link is the *childlink* which points to the child node of the node. Second link is the *Nextchildlink* which points to the next sibling of that node. Third link is the *parentlink* which points to the parent of the node. The fourth link is the *Nextparentlink* which points to the other parent of its child.

3.2 XPath to PAT

XQuery and XPath are expression languages. These two are based on a data model that consists of a tree structure of various nodes. XPath expression specifies a pattern that selects a set of XML nodes. XPath query is represented in PAT [14] so that algebraic transformation can be performed. XPath queries are transformed into an equivalent PAT algebraic expression to perform optimization by applying certain rules.

3.3 Optimization of Input PAT Expression

A comprehensive set of general equivalences with regard to XML documents and XML queries are developed. Based on these equivalences, a large set of deterministic algebraic transformation rules are developed for XML query optimization. The PAT expression that is obtained is optimized, by applying the set of rules that are deterministic. The rules are applied based on the properties identified from the tree. All transformation rules in this approach are intrinsically deterministic as, once applied, a transformation brings in definite improvement on its input query. The optimization is done in three phases.

3.3.1 Normalization:

During this phase, the algebraic properties of the PAT algebra are exploited to isolate different types of operators at different levels, eliminate the ∩ operator from the query expression and push down the more selective operations. The complete elimination of the ∩ operator from a PAT expression is beneficial for applying algebraic transformations on PAT expressions. Pushing the $ operators to the bottom level can simplify the identification of potential indices. The selection operators are the most beneficial ones to evaluate first. In most of the formatted PAT expressions, the selection operations are specified more early and the set operators are typically applied to the selection results. So it is expected that most incoming queries are in a format similar to that. The normalization phase also performs obvious simplifications, like transforming "*E U E*" to a single *E*, to make the subsequent semantic optimization more efficient.

3.3.2 Semantic Transformation:

The primary goal of this phase is to identify the opportunities of applying potential structure indices which is a short cut path between any two nodes. This semantic transformation phase focuses on exploiting the opportunities of using an available structure index or enabling such an opportunity to applying a structure index. This phase applies the rules to the queries involving the elements that satisfy certain properties as already explained. These properties are identified from the tree constructed.

3.3.3 Simplification

Semantic transformations typically bring in new sub expressions, including new PAT operators, in particular the index and intersection operators. Hence, a further run of the simplification rules of Phase 1 is desired, and additional simplification rules are needed to handle the combination of the index operator with the ∩ operator. The third phase is intended

to carry out a bottom-up cleaning on the output of semantic optimization. The rules that enforce explicit constraints during Phase 1 are excluded because semantic transformation does not normally introduce new inconsistencies. The rules that reorder operators are not needed either, as the semantic phase does not change the main structure of a query. The above three phases are applied continuously till a final optimized query is reached. The optimized PAT expression obtained through the above optimization process results in reduced number of operators when compared to the original non-optimized PAT expression.

3.3.4 Optimized PAT to XPath

The optimized PAT expression is converted to its equivalent XPath expression for mapping to SQL. As the given input is XPath query, the PAT expression after optimization is converted back to XPath so that the query can be directly executed to obtain the results using stylesheets. Apart from this, it is also easy to map the XPath queries to SQL than mapping PAT to SQL using the algorithm as will be explained.

3.3.5 Mapping XML to Relational

The XML document is mapped to the corresponding relational database following the algorithm described [15][17]. Relational database is used to show the effect of query optimization. The original XPath query and the optimized XPath query are mapped to their respective SQL queries for testing the performance in a relational database Oracle [10]. To clearly show the time differences in the execution times of the non-optimized and optimized XPath queries they are mapped to equivalent SQL queries based on the algorithm.

4 SIMULATION AND RESULTS

The experiments were conducted on a system which runs Windows XP and the CPU is an Intel 2.3 GHz processor with 1 GB RAM. The performance results with Databases are presented below. Three types of databases are used to show the performance results of the optimization approach. The first database [Database 1] uses two XML documents having less number of levels. Thus the complexity involved in the optimization is less and not much of difference is noticed in the execution time for the optimized and non-optimized queries. The second database [Database 2] is complex than the first in that it contains two XML documents, which have large number of levels. As the levels are larger, XPath queries written for these documents involve larger path length. The path length gets reduced for these types of XML documents using the optimization approach explained. Time taken for execution of both the optimized and non-optimized queries has a clear difference, which is depicted in Fig.4. The third database [Database 3] is used for the purpose of showing the applicability of this approach for more than two documents.

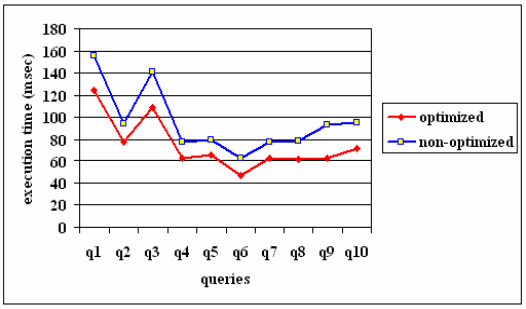

Figure 3 Performance result for Database 2

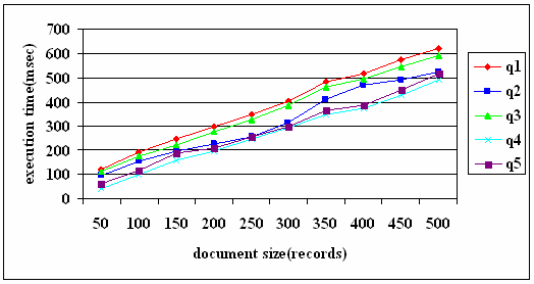

Figure 4 scaling effect of optimized queries with Database 2

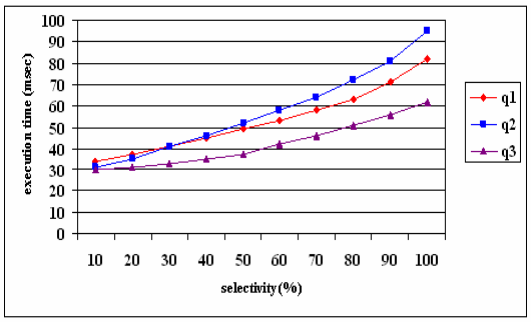

Figure.5 Effect of selectivity on optimized queries with Database 3

5 Conclusion and Future work

The proposed technique proved efficient when implemented and tested in terms of Execution time, scalability and selectivity. Further, this approach operates at the logical level and is therefore independent of any storage model. Hence, the optimizers developed based on this approach can be easily adapted to a broad range of XML data/information server to achieve faster query optimization.

6 References

[1] Amer-Yahia. S, Cho. S, Lakshmanan. L, and Srivastava. D, "Minimization of tree pattern queries", Proceeding of ACM Conf. on Management of Data, 2001.

[2] Bohannon P, Freire J, Roy P, Simeon J, "From XML Schema to relations: A cost-based approach to XML Storage", 18th International conference on Data Engineering, 2002.

[3] Bohm K, Aberer K.T, Ozsu M, Gayer K, "Query Optimization for structured documents based on knowledge on the document type definiton", IEEE International forum on Research and Technology Advances in Digital Libraries, 1998.

[4] Buneman. P,Fan. W, Weinstein . S, " Query Optimization for semistructured data using path constrints in determinist data model", In Procedding of DBPL, 1999

[5] Chan C.Y, Felber P, Garofalakis M, Rastogi R, "Efficient filtereing of XML documents with XPath expression", International Conference on Data Engineering, 2002.

[6] Chung. T-S, Kim. H-J, "XML query processing using document type definitions", Journal of Systems and Software, 2002.

[7] Dunren Che, Karl Aberer, Tamer Ozsu M, "Query Optimization in XML Structured document databses", The VLDB Journal , 2006.

[8] Fernandez M, Tan W, Suciu, "Trading between Relations and XML", 9th International World Wide Web conference, 2000.

[9] Fernandez. M.F, Suciu. D, " Optimizing regular path expressions using graph schemas" In Proceeding for the fourteenth International conference on Data engineering, February, 1998.

[10] Flesca. S, Furfaro. F, Masciari. E, " On minimization of Xpath Queries", VLDB, 2003.

[11] Florescu D, Kossmann D, "Storing and Querying XML data using an RDBMS", IEEE, 1999

[12] Kwong A, Gertz M, "Schema-based optimization of XPath expression", Technical report, Dept. of Computer Science, Univeristy of California, 2001.

[13] McHugh. J, Abiteboult. S, Goldman. R, Quass. D, Widom. J, " Lore: A Datbase Management System for semistructured data", ACM SIGMOD RECORD, 1997.

[14] Salminen A, Tompa F.W, "PAT expressions: An algebra for text search", Acta Linguistica Hungarica, 1994.

[15] Shanmugasundaram, J., Tufte, K., He, G., Zhang, C., DeWitt,D., Naughton, J. "Relational databases for querying XML documents:Limitations and Opportunities",VLDB(1999)

[16] Wang. G, Liu. M, "Query Processing and optimization for regular path expressions", In proceedings of CaiSE, 2003.

[17] Zhang C, Naughton J.F, DeWitt D. J, Luo Q, Lohman G.M, " On Supporting Containment Queries in Relational database management systems", ACM SIGMOD (2001).

[18] XML-Managing Data Exchange/ The one-to-many relationship (http://en.wikibooks.org/wiki/XML_-_Managing_Data_Exchange/The_one-to-many_relationship).

[19] www.developer.com/xml/article.php/1575731