Nested XPath Query Optimization for XML Structured Document Database

Radha Senthilkumar[#1], G. B. Rakesh, N.Sasikala, M.Gowrishankar[2], A. Kannan[3]

[1,2] *Department of Information Technology, MIT Campus, Anna University*

[3]*Department of Computer Science Engineering, CEG Campus, Anna University*

Chennai, Tamil Nadu, India

[#] radhasenthil@annauniv.edu

***Abstract* - The XPath language is based on a tree representation of the XML document, and provides the ability to navigate around the tree, selecting nodes by a variety of criteria. Here an optimization plan is proposed, to unnest and optimize nested XPath query for XML Inter and Intra document relationship. Inter and Intra document relationships are techniques of implementing one to many relationship. We propose an enhanced variant of kappa join which is used for query unnesting. Further the deterministic optimization approach which exploits the structure of XML document is used to optimize the unnested query. Previous works have focused only on unnesting phase of nested query optimization for a containment and intra document relationship. This paper further extends the unnesting strategy with the deterministic optimization approach for Inter and intra document relationship. The optimization plan starts with unnesting of the query. Unnesting is performed by means of enhanced variant of kappa join taking into account the XML relationship. Unnested query is converted to a internal PAT(Pattern) representation. This PAT expression is optimized by deterministic transformation on queries using the structure knowledge of XML data and structure-related semantics. Then the optimized PAT expression is converted to XPath query. Finally both the normal and optimized query is executed in DB XML database to evaluate the execution time. The final results prove that the optimized query executes faster with better scalability, selectivity and reduction in execution time.**

I. INTRODUCTION

XML (eXtensible Markup Language) is a standardized format to store semi-structured documents. It has emerged as the predominant mechanism for representing and exchanging data. For example, it is used in document-centric applications for modeling and exchanging data in the financial (FIXML), chemical (CML), or biological (BIOML) sector. Moreover, web standards such as Web Services use XML for communicating with clients or among each other. Because of its widespread use and the large amounts of data that are represented in XML, the necessity arises for efficiently managing and storing XML data. This need led the major commercial database system vendors as well as many open source projects (e.g. eXist, MonetDB, MySQL) to integrate XML support in their systems. In order to query XML data, XPath and XQuery have been developed Both are declarative query languages and, hence, can benefit from powerful optimizations. Query Optimization is the function of Database Management Systems in which multiple query plans for satisfying a query are examined and a good query plan is database performance tuning is to minimize the response time of your queries and to make the best use of your server's resources by minimizing network traffic, disk I/O, and CPU time. Query optimization is one of the approaches to achieve this.

A. Need for Nested XPath Query Optimization

Query optimization has been typically used in database management systems as an effective way of accelerating the evaluation of posed queries. Query optimization in the case of XML database is challenging because of two reasons:

- First, as the repositories of structured documents tend to be large, document query evaluation is becoming unacceptably slow.
- XML data model is complex when compared to other data models.

The existing query optimization approaches which primarily focuses on simple Xpath queries cannot be applied to complex nested XPath queries as they evaluate the query in a nested fashion and are deeply nested. Hence, here we develop an optimization approach for efficient processing and optimization of complex nested XPath queries.

II. RELATED WORK

Brantner [7] proposed optimization techniques for Nested XPath query for intra document relationship. It presents algebraic equivalences to unnest nested XPath queries. This article extends the approach for inter document relationship. Matthias [9] proposed XPath queries involving comparison expression which are existentially quantified. In this article [9], the author has developed a new ternary operator called Kappa-join, for efficient evaluation of queries with existential quantification. Dunren [4] proposed XPath with containment operators which are optimized by performing deterministic algebraic transformation. Thorsten [15] gives a tour of Native database management system designed for storing and processing XML data. Lore [10] is a DBMS originally designed for semi structured data and later migrated to XML based data model. The Lore optimizer is cost based and does not perform logical level optimization. Brantner [8] presents a first complete translation of XPath into algebra. Brantner [11] presents a framework for handling nested queries that is based on unnesting the queries after having translated them into an algebra. It not only presents a collection of algebraic

equivalences, but also supplies a strategy on how to use them effectively. In [1] the author studied the minimization issue of general tree pattern queries, and both constraint dependent and constraint independent minimizations were addressed. In [18] the notion of a path is that it always starts from the root. We identify more opportunities for reducing a path, since it need not start from the root. In [12], the author has proposed a query optimization technique for inter document relationship and in [13] optimization technique for intra document relationship has been proposed. In this article we apply this technique with respect to a nested query.

III. Proposed Optimization Technique

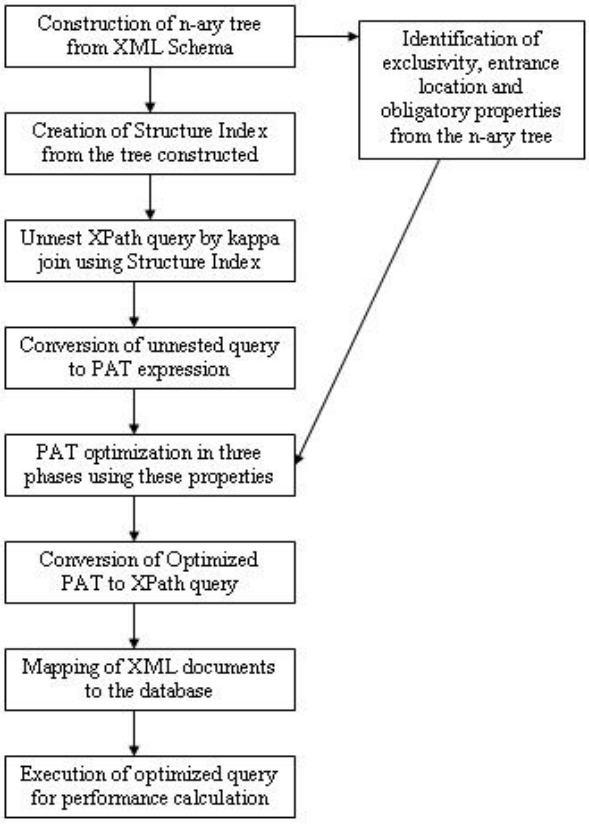

Fig.1. System Design of Nested Query Optimization

A. Unnesting Phase

Nested XPath query is unnested by means of enhanced variant of kappa join [9]. Normal evaluation of nested XPath queries executes the steps in the order of their occurrences. In this case, the inner predicates are evaluated unnecessarily for every context resulting from the outer expression. Hence the nested queries are divided into dependent and independent expression.

Every expression in XPath is evaluated with respect to a context. *Dependent expression* is expression which depends on local context and *independent expression* depends only on global context.

Consider the sample nested query

Nested XPath query:

/mondial/country/border[@country=//country[@name='Austria']/@id]@length.

Unnested Query: //country[@name='Austria]/@id

/mondial/country/border[@country=output of I query]/@length

B. XPath to PAT Conversion

The XPath expression is converted to an internal PAT expression [14] for further optimization. XPath is converted to PAT so that optimization can be performed more efficiently. PAT expression is an algebraic representation of the expression. The following algorithm is used for XPath to PAT conversion.

```
Parse-xpath (xpath-query)
    Get the xpath token from the xpath query
    Stack-push (tokens)
    Convert-pat (stack)
Convert-pat (stack)
    While stack
     do
          Element=stack.pop()
          if element contains "/" and element is at the end of the stack
        then
          Convert element as σ (element)
              if element contain '@'
                then
                    Convert element as σ attr (element)
              if element contains function()
                then
                    Element=getnext element ()
    Convert element as σ function (attr) (element)
```

Concatenate the converted elements and print the PAT expression

C. PAT Optimization

Now the converted PAT expression is optimized further by the deterministic optimization approach([4],[12],[13]). Deterministic means, transformation once carried out will bring in definite improvement on the input query expression. Optimization is carried out in three phases namely Normalization, Semantic transformation and Simplification. All the queries are not optimized in all the three phases. It varies from query to query. In our approach the main optimization phase is Semantic Transformation.

1) Construction of n-ary Tree: XML documents are validated using XML Schema. It defines the structure of XML document. The XML schema is taken as input to construct the n-ary tree([3],[13]). An n-ary tree is constructed for each table in the document. The node representation is given below

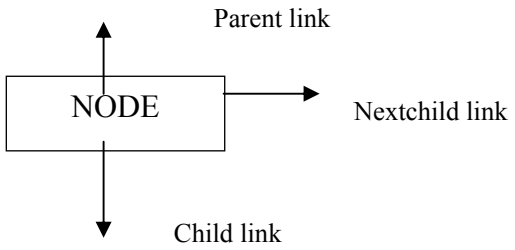

Fig. 2. Node representation of an element in XML document

2) Identification of Structural Properties: From the tree constructed various properties existent in the XML documents [4] are found out. The three such properties are:

- Exclusivity
- Entrance Locations
- Obligation

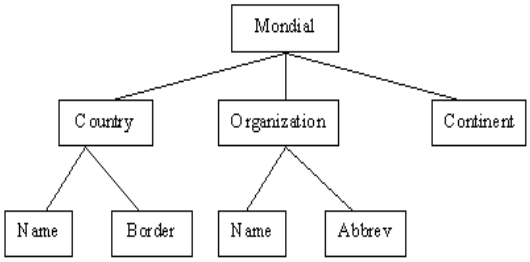

Fig. 3. Tree representation of Mondial Database

Exclusivity: Element Ej is said to be exclusively contained in Ei if all the paths from Ej to the root node passes through Ei. For e.g. name is exclusively contained in country.

Entrance locations: Element E is an entrance location for elements E1 and E2 if in any document complying with the schema, all paths from an element E1 to an element E2 passes through an element E. For e.g. country is the entrance location for mondial and name.

Obligation: Element Ei obligatorily contains element Ej if each element Ei has to contain an element Ej in all the documents complying with the schema. For e.g. mondial obligatorily contains country.

These properties are used to reduce the time taken to process a query by reducing the number of operations and thereby improving the efficiency of performance.

1) *Normalization:* This phase achieves three main goals

- Isolate different types of operators at different level
- Completely eliminate the ∩ operator from the expression
- More selective operations are pushed down.

This phase has its own set of rules which is based on enforcing DTD constraints, operator reordering etc.

2) *Semantic Transformation:* This phase identifies the opportunities of applying the structure indices. When an index operator is used in the query it reintroduces the ∩ operator which is again eliminated in the third phase. It identifies the structural properties of the XML document for query optimization.

3) *Simplification:* In the second phase new PAT operators especially intersection and index operators are introduced. In order to remove them we apply the simplification rules.

IV. ILLUSTRATION

A sample Query is taken and the overall optimization approach is applied to it.

Sample query:

Select the length of the border of the country Austria.

Nested XPath query:

/mondial/country/border[@country=//country[@name='Austria']/@id]/@length

Unnested query:

s1= //country[@name='Austria']/@id

s2= /mondial/country/border[@country=s1]/@length

PAT conversion:

id< $\$_{name,'Austria'}$ (country)

length< $\$_{country,s1}$(border)<country<mondial

PAT optimization:

By applying (E1<E2) => (E1<E3) if E2 is an entrance location for E1 and E3 and a structure index available for E1 and E3.

After the rule is applied, the XPath query becomes

/mondial/border[@country=s1]/@length

V. EXPERIMENTAL RESULTS

We conducted a series of experiments to test the proposed optimization technique. This experiment was conducted to prove the efficiency and effectiveness of the approach. So we chose two databases to conduct the experiments and performance results were tabulated and shown using graphs.

A. Benchmarking database

For experimentation purpose we chose a benchmarking database for the Inter document relationship and Intra document relationship. Mondial database [21] is used as the dataset. The Mondial database has been compiled from geographical web data sources. It provides a comprehensive example for XML and is a case study for information extraction and integration.

B. Sample Nested Queries

Q1./mondial/country/border[@country=//country[@name='Austria']/@id]/@length

Q2./mondial/country/province[@country=//country[@name='Austria']/@id] /@population

Q3./mondial/organization/members[@country=/mondial/country
[@name=”Austria”]/@id]/@type

Q4./mondial/country/city/located_at[@water=//lake[@name=’LakeSkutari’]/@id]@type

Q5./mondial/country/city[@province=//province[@name=
’Vienna’]/@id] /@id

C. Experimental Environment

The characteristics of the database chosen for our experiment are given above. We performed our experiment on a single system LINUX platform. The CPU is a Intel Pentium IV processor of 3.2 GHz speed with 512 MB RAM. We used DB XML database to load our XML documents and execute the query for performance calculation.

D. Performance Results

The designed sample queries were executed for both Inter and Intra Document Relationship and the performance results were calculated. The results showed reduction in execution time with better scalability and selectivity.

E. Intra Document Relationship

TABLE I
COMPARISON OF EXECUTION TIMES (IN SECS) OF THE QUERIES FOR INTRA DOCUMENT RELATIONSHIP

Query	Q1	Q2	Q3	Q4	Q5
Unoptimized	5.495	9.823	17.622	11.73	8.73
Optimized	0.62	0.97	1.83	1.1	0.9

1) Execution Time Graph: From the table and the graph, it is clear that the execution time of the optimized queries have been reduced considerable irrespective of the nesting level. For certain queries, the execution time remains unchanged.

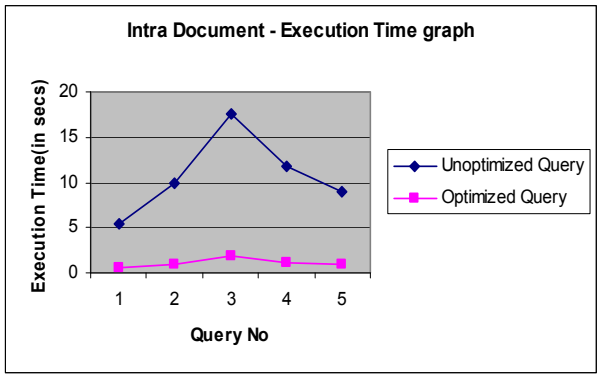

Fig.4. Execution Time (in secs) graph for Intra Document Relationship

2) Scalability Graph: This graph is used to demonstrate that the execution time of the queries increases linearly as the document size is increased.

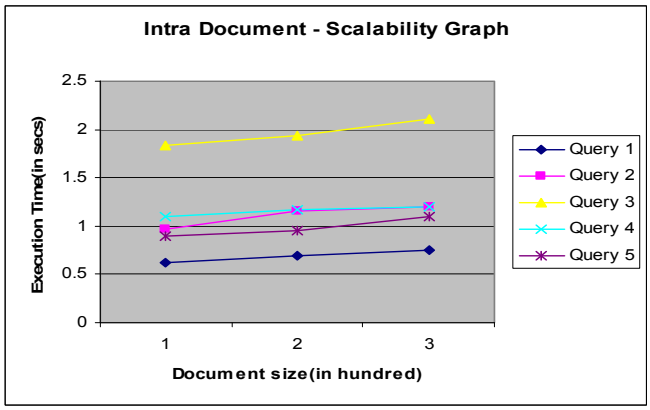

Fig.5. Scalability graph for Intra Document Relationship

3) Selectivity Graph: Selectivity plays a major role in the execution performance as it affects the time to execution of the queries when more number of records are to be retrieved i.e., when the selectivity percentage is high. The graph below shows quasi exponential effect for selectivity of queries for Intra Document relationship

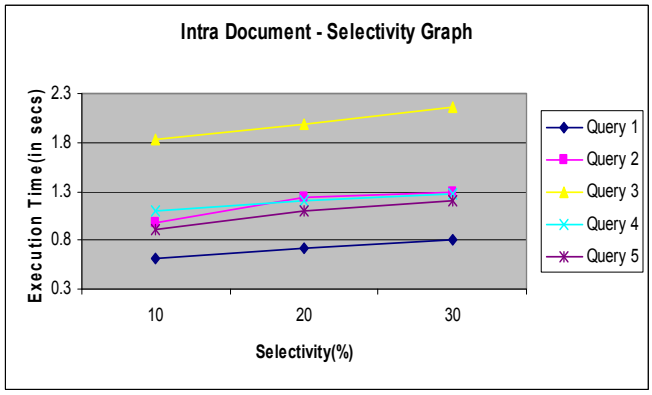

Fig.6. Selectivity graph for Intra Document Relationship

F. Inter Document Relationship

TABLE II
COMPARISON OF THE EXECUTION TIME(IN SECS) OF THE QUERIES FOR INTRA DOCUMENT RELATIONSHIP

Query	Q1	Q2	Q3	Q4	Q5
Unoptimized	5.495	11.56	19.63	12.11	9.94
Optimized	0.62	0.97	1.83	1.1	0.9

1) Execution Time Graph: From the table and the graph, it is clear that the execution time of the optimized queries have been reduced considerable irrespective of the nesting level. For certain queries, the execution time remains unchanged.

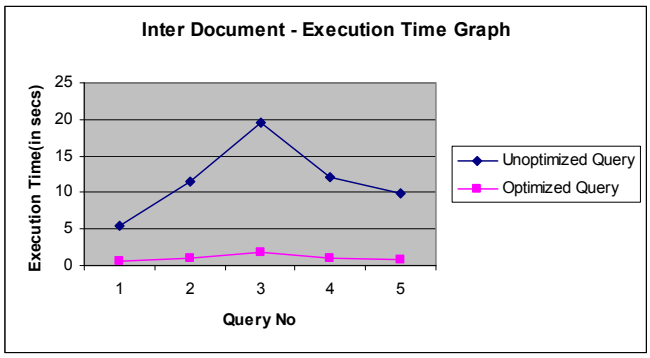

Fig.7. Execution Time (in secs) graph for Inter Document Relationship

2) Scalability Graph: This graph is used to demonstrate that the execution time of the queries increases linearly as the document size of any one of the documents is increased.

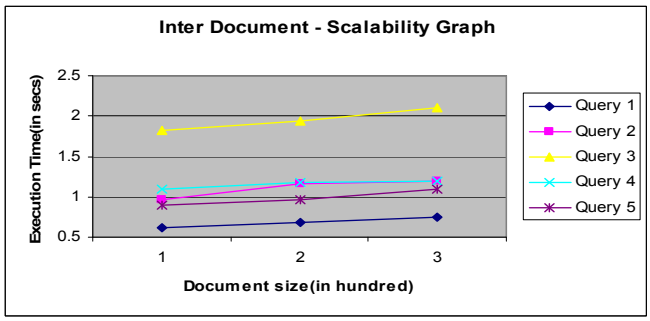

Fig.8. Scalability graph for Inter Document Relationship

3) Selectiviy Graph: The graph below shows quasi exponential effect for selectivity of queries for Inter Document relationship.

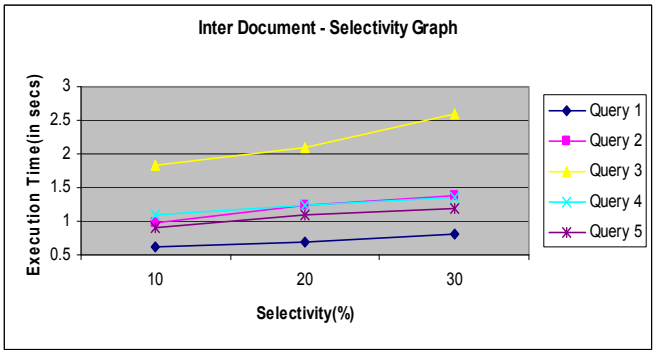

Fig.9. Selectivity graph for Inter Document Relationship

VI. Conclusion

In this paper, the proposed optimization technique was proved efficient for a subclass of XPath queries. We chose only a class of Conjunctive correlated queries and applied the optimization technique. This approach can be further extended to disjuntive correlated queries and a whole set of XPath queries. Only a subset of PAT algebra operators was used in the conversion process. In the deterministic optimization phases, semantic transformation phase was exploited efficiently to optimize the queries. Experiment results prove improved efficiency when applied to both Inter and Intra document relationship. With respect to the execution time, it was reduced to a reasonable amount and regarding the scalability factor, the execution time varies linearly as the size of the document was increased in terms of hundred. So the final optimized query was obtained in two phases of unnesting and applying the deterministic optimization approach to the unnested query. Therefore the technique was proved efficient when implemented and tested in terms of execution time and scalability for both Inter and Intra document relationship.

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] 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.

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

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

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

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

[7] M. Brantner, C-C. Kanne, G. Moerkotte and S. Helmer.Algebraic optimization of Nested XPath Expresssion. In *Proc. ICDEConference*, Atlanta,page 128, 2006.

[8] M. Brantner, C-C. Kanne, S. Helmer, and G. Moerkotte. Full-fledged Algebraic XPath Processing in Natix. In *Proc. ICDEConference*, pages 705–716, 2005.

[9] M. Brantner, S. Helmer, C-C. Kanne and G. Moerkotte. Kappa-join: Efficient Execution of Existential Quantification in XML Query languages

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

[11] N. May, S. Helmer, and G. Moerkotte. Strategies for Query Unnesting in XML Databases. Technical report, University of Mannheim, 2005.

[12] Radha Senthilkumar, A. Kannan, M. Bhuvaneswari, "Query Optimization for Inter Document Relationships in XML Structured Document," *iccima*, pp. 25-32, International Conference on Computational Intelligence and Multimedia Applications - Vol.2 (ICCIMA 2007), 2007

[13] Radha Senthilkumar, A. Kannan, V. Prasanna, P.Hindumathi , "Query Optimization for Intra Document Relationships in XML Structured Document", International conference on open-source systems & Technologies (ICOSST 2007)

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

[15] Thorsten Fiebig, Sven Helmer, Carl-Christian Kanne, Guido Moerkotte, Julia Neumann, Robert Schiele, and Till Westmann. Anatomy of a Native XML base management system. *VLDB Journal*, 2003.

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

[17] www.w3.org/TR/xpath

[18] www.w3schools.com/xpath/default.asp

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

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

[21] www.informatik.uni-freiburg.de/~may/lopix/lopix-mondial.html