Efficient Evaluation of Query for XML Graph Based on Single Source Shortest Path

1Manmohan Singh, 2Chiranjeev Kumar
1,2Computer science & Engineering, Indian School of Mines Dhanbad, India
Email: 1mannohansinghnumber@gmail.com, 2Kumar.c.cse@ismdhanbad.ac.in

Abstract— XML (Extensible markup language) is widely accepted as the standard for data representation and e-data interchange. Therefore a large amount of data is stored in database. How to store bulk amount of data and fetch the expected data with an intelligent approach is the most common and important question. It is also known that optimizing query processing is one of the crucial tasks in case of database of large size. Numbers of techniques are designed to solve this open problem. One of effective way to access the query is XPath in XML database. XPath allows us to pick node(s) out of our XML tree. The expression argument is simply a string containing the nodes that we want to evaluate. The sequences that are returned by path expression have an inherent and stable order, which is called document order. In this technique, all nodes will get traversed to solve the expression. It will lead to more execution time. In this paper, we have used single source shortest path algorithm in alliance with topological sorting algorithm to solve the problem. Our technique will return less number of nodes in expression, and hence distance will be less. Resulting query can thus be evaluated more effectively leading to an increase in performance and also the storage space will reduce.

Keywords: Strongly connected component (SCC); Directed acyclic graph (DAG); DAG-SHORTEST-PATHS;

I. INTRODUCTION

XML is self-describing data format for representation of semi structured database. Recent advancement on XML, biological databases, scientific data, social network analysis, the Semantic Web, h Filtering [5], and many other emerging applications have sparked renewed interests of graph-structured databases (or simply graphs) and the related problems of data storage practices and query processing [16-17]. Besides being flexible, self describing or semi structured and extensible in nature, the structure of XML may vary from a flat regular data centric structure to a deep irregular document centric structure. XML database contains ancestor-descendant ("/" ) relationship that is generally known as hierarchical relationship. Computer can access quickly and efficiently in hierarchical data structure in comparison to non hierarchical data structure. ID/IDREF (identity reference) is an important and widely used feature in XML documents to reduce redundancy in database. Generally to eliminate redundant data we introduce ID/IDREF attributes [14].

ID/IDREF can be linked to primary key or foreign key in a relational database. Using ID/IDREF, each object can occur once under the document root with a unique ID. The ID/IDREF mechanism provided by XML DTD relies on a simple form of constraints to describe references [15].

One of the efficient way to make XML graph by using hierarchical relationship with ID/IDREF in the database. XML query can be evaluated as two approaches first approach which uses the Principal of graph traversal algorithm (1) Breadth First or (2) Depth First and the second approach can be by pre-compiling the transitive closure. These approaches are not suitable if XML graph contains many cycles and the database is large [1-3]. Since graph structure is too large, traversal of graph either DFS or BFS cursor have to traverse each node of XPath which is defined as parts of an XML document. We make directed acyclic graph by strongly connected component of a complex database and reduce path indexing of xml graph as well as storage space of xml graph. The strongly connected components, linear time algorithm computes the strongly connected components of directed graph G = (V, E) using two depth-first searches, one on G and one on G^1 [22]. Xpath traverses the node from root. This proposed approach also gives assistance in distributed system. If whole data is fragmented [19-21] into different sites, Xpath traverse from local root nodes and evaluate the query of local site in parallel. Thereafter we combine the individual results and obtain the final result, using this strategy we can reduce the execution time of query. There is no cycle and negative edge weight cycle can exist in DAG. Hence single source shortest paths problem for DAG can be solved more efficiently by using topological sort. Topological sort of DAG G = (V, E) is a linear ordering of its entire node such that if G has an edge (u, v), then u appears before v in the ordering. Total time of topological sort is O (V+E).

The rest of the paper is organized as follows. Section-2 reviews related work, fundamental theory of SCC and
reduced database. Section 3 reviews how to reduce the huge amount of XML database. Section 4 represents optimization of evaluation technique and applying single source shortest paths in DAG. Section 5 deals with conclusion and future work.

II. RELATED WORK

There are many approaches available for retrieval of desired nodes by Xpath [6]. Numbering and labeling scheme that are typically used to avoid exhaustive traversing of documents for query processing [10]. Many of the proposed labels focus on the simplified case tree like (rather than graph-like) data and simple path expressions (like, movie/director/name, “/”) are the parent-child axis. Path indexing is one of the techniques for Xpath process [11-13]. Path indexing creates path summary from root node to desired node. Data Guide (Gold man widom, 1997) indexes each raw data path to facilitate the evaluation of path expression. T indexed (Milo and Sucio, 1993) select path based on templates. D (k) indexes select the most frequently data used however these approaches suffer from large size of path indexing. Let us suppose that the Xpath expression is like /book [title]/chapter/section/figure. We have traverse all downwords path starting from book element. In this way cursor has to track all paths to find out where there exists any immediate chapter element. Next, it traverses down all other element nodes one by one until it reaches the chapter element and so on for comparison. For the next set of matches, it needs to backtrack to its previous visited book element and start the search again. This situation occurs when query contain edges with ancestor-descendant relationship. Any expression in square brackets that filters a step in a path expression is a predicate. Numerical predicates are the easiest to understand, since they test for the location of an element in a sequence according to what gets returned by depth-first traversal of the tree.

Other technique is probabilistic based model [7] which extract the information may have some form of certainty or probability associated with it. To manage probabilistic data, there has been a lot of work on supporting probabilistic data in relational database (Deay and sarkar, 1996; Eiter at Al. 2000, fuhr rolleke, 1997; Lakshmanan et al., 1997). They considered the modeling of uncertain data in relational database and there have been some issues such as the unit with which probabilities will be associated and whether the resulting database is to remain in the first normal form.

III. REDUCING XML GRAPH

According to needs, size of data increase (like Tree bank database [8]) and to store whole data storage space will be increases. Existing-life example of XML graph of a database of the internet movie database (IMDB) with links is shown below as fig 3.1. The general characteristics of this data are as follows. The center file of the IMDB [2-3] has a list of movies, each with a unique identifier. The actors of those movies are listed with their roles in a distinct file. All directors are listed in an independent file, with a number of important producers, writers, and cinematographers.

The IMDB is used because it was identified as a highly cyclic database likely to stress the path-indexing algorithms. To fulfill this requirement (storage space less) one of the easiest way is to apply the strongly connected component algorithm [9] on XML database. It gives the output a portion of nodes into a disjoint set so that each set defines a strongly connected component graph. This means output of this graph does not contain cycle. Now, the generation of linked XML documents from this data. A small subset of movies and all people (actor, directors, etc.) associated with these movies is randomly chosen. One XML document for each movie is generated and an XLink to the actors and the director for the underlying movie is added.

![XML Graph of internet movie database](image1)

![Reduced xml graph by applying SCC](image2)
The portion of the used database is organized around movie elements and elements for classes of people who appear in movie credits, for example, actor, director, composer, etc., as well as a wide variety of information about movies. Cyclist arises since each movie element is serviced as ID references and has pointers to individuals who acted in the movie, and each element representing an individual pointer to the movies in which she or he acted. This datasets consists of 40,211 nodes and 44,349 edges, among which 2718 are of type IDREF.

This means that the optimization techniques based on the principle of the SCC, reduce of the size of DAG graph more than 30% compared with the original XML graph.

IV. OPTIMIZATION OF EVALUATING TECHNIQUE

To efficiently query xml documents, there is a need of efficient indexing structure and best labeling scheme with query processing algorithm. Labeling scheme identifies the relationship between two nodes. When xml graph is huge there can be several paths from one node to others. In xml graph we have applied the strongly component algorithm and output is direct acyclic graph (DAG). To provide better labeling scheme, if root node store the shortest path from root to desired nodes, there is no requirement to track longer path. Suppose xpath query is /movie/movie/actor[3]/actor[1] or query is /movie/movie/actor[3]/actor[1]/name[1]. These two queries is equal to /movie/actor[1] and /movie/actor[1]/name.

4.1 SINGLE SOURCE SHORTEST PATHS IN DAG

We know that query/movie/movie/actor[3]/actor[1] is equivalent to movie/actor[1]. User always request the query according to its own analysis and does not know which is shortest path. XML database management system should be able to decide which the shortest path is. This expert system can be achieved by some intellectual algorithms. Our technique proposes how we can make such system. Firstly we have reduced the nodes of graph by applying strongly connected component algorithm and further we can reduce the graph by DAG-SHORTEST-PATHS. As we have shown in graph there are so many no of the xpath. Once we get the output of the DAG-SHORTEST-PATH we store it permanently. Analyzing the query xml database management system justified the root node and destination node. According to query, cursor traverses from root node to destination node. If node is reachable [23], xml database system management returns the query result otherwise it returns the NULL value.

By relaxing the edges of weighted (weight of each is 1) DAG (directed acyclic graph) \( G = (V, E) \) according to topological sort of its vertices, we can compute shortest paths from a single source in \( \Theta (V+E) \) time. Here we assume each edge has weight is one. This means increases the no of edge means increase in distance.

DAG-SHORTEST-PATHS [10] \((G, w, s)\)

1. Topologically sort the vertices of \( G \).
2. INITIALIZE-SINGLE-SOURCE \((G, s)\)
3. For each vertex \( u \), taken in topological sorted order
4. For each vertex \( v \in G. Adj [u] \)
5. Relax \((u, v, w)\)

The time complexity of above algorithm is easy to analyze. The topological sort of line 1 takes \( \Theta (V+E) \) time. The call of INITIALIZE-SINGLE-SOURCE in line 2 takes \( \Theta (V) \) time. The for loop of lines 3-5 makes one iteration per vertex. Hence total time complexity is \( \Theta (V+E) \) which is very less. Cursor can move so fast forward and backward. If root node contains the table, the cursor can go at the desired node by shortest path. The important point is applying DAG-SHORTEST-PATHS, Relax \((u, v, w)\) reduce the graph. Therefore there is a need of less no of ID/IDREF of each node.

THEOREM: If a weighted, directed graph \( G = (V, E) \) has source vertex \( s \) and no cycles, then at the termination of the DAG-SHORTEST-PATHS procedure, \( v.d = \delta (s, v) \) for all vertices \( v \in V \), and the predecessor sub graph \( G_s \) is a shortest-paths tree.

PROOF: We first show that \( v.d = \delta (s, v) \) for all vertices \( v \in V \) at the termination. If \( v \) is not reachable from \( s \), then \( v.d = (s, v) = \infty \) by no path property. Now suppose that \( v \) is reachable from \( s \), so that there is a shortest path \( p = v_0, v_1, \ldots, v_k \) where \( v_0 = s \) and \( v_k = v \), because we process the vertices in topologically sorted order, we relax the edges on \( p \) in the order \( (v_0, v_1) , (v_1, v_2) \), etc.
The path-relaxation property implies that $v_i d = \delta(s, v_j)$ at termination for $i = 0, 1, 2, \ldots, k$. Finally, by the predecessor sub graph property, $G_n$ is a shortest-paths tree.

This technique is also helpful when our data is stored in distributed system. In relational database we fragment the data in horizontal or vertical according to needs. But in xml database to efficiently query evaluation, fragment the data in xpath. Each fragmented xpath is stored in one or more than one nodes. Each node evaluates the query parallel and gives the combined result. This technique gives the illustration or strategies how to fragment the xml database.

V. CONCLUSION AND FUTURE SCOPE

Due to adoption of xml in internet it is spawned as a basic foundation of standards that are becoming basic structure of electronic data interchange, business and application development. With this revolution, we considered an effective way of storing and querying xml document. Firstly reduce xml graph of big data with an efficient approach and secondly emphasizes retrieving of complex xml query which comprises of a set of xml query. It is novel approach which provide expert system to take decision according to query requested by user which path is shortest because user request the query arbitrarily. If processor traverse the query as entered by user it can be go in longest path and cursor will take more time to execute it. This technique is useful when user entered query having more depth. Still there is room to be further improvement. Because it having some short comes when we request the query which inserts the data into xml database, xml graph will be changed. Then we will have to apply the again SCC and DAG-SHORTEST–PATH algorithm and further complexity will be increased which is extra overhead.

REFERENCES


[4] Wen-chiao Hsu, I-En Liao, CIS-X: A compacted indexing scheme for efficient query evaluation of XML documents, Department of Computer Science and Engineering, National Chung Hsing University, 250 Kuo Kuang Road, Taichung 402, Taiwan


[9] Samir Khuller, Balaji Raghavachari, Neal Young, strongly connected digraphs with bounded cycle length,

[10] Su-Cheng Haw*, Chien-Sing Lee, Extending path summary and region encoding for efficient structural query Processing in native XML databases, Faculty of Information Technology, Multimedia University 63100 Cyberjaya, Malaysia


[14] Denilson Barbosa and Alberto O. Mendelzon, Finding ID Attributes in XML Documents University of Toronto Toronto, Ontario, Canada dmh,mendel@cs.toronto.edu


[18] W. Lian, N. Mamoulist, W.L. Cheung, S.M. Yiu,
Indexing useful structural patterns for XML query processing, IEEE Transactions on Knowledge and Data Engineering 17 (7) (2005) 997–1009.


