Volume 5, Issue 5, May 2015 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

A Review of Indexing Techniques for XML Databases

Nisha
Department of Computer Science
University of Delhi, India

Abstract: As the large data travel over internet we need XML to send, store and process this data. Recently the research area which has come up with great opportunities is indexing and retrieving the Xml documents because it allows accessing the xml documents. This survey paper includes many indexing techniques in native databases and comparison between these techniques.

Keywords: Indexing techniques and semi-structured documents.

I. INTRODUCTION

As there is huge amount of data over the internet, it is difficult to get the exact information which user is looking for. Now XML is represented as a new standard which helps in standardizing the format of data which better describe the information regardless of data type. Semi structured document is a document which integrates meta- information and structured information other than original data.

Now Xml document is supported by relational database management system, which is the extension to RDBMS. The main role of XML is to exchange data over the internet but it is needed to store the XML data in the database. Native XML database management system (XDBMS) was introduced to manage XML data which store the XML data in native form. Semi-structured data do not have a fixed schema known in advance and store separate from data unlike traditional database systems. Now if the XML data is stored in database many questions arises: 1.how to make query for data? 2. How can XML data be indexed to speed up frequent queries? etc. Tools to access relevant information and that are integrated within specific information are react accordingly. So the main problem is of retrieval (indexing). Indexing has been a important issue always because if indexing is not good then user will not ever get proper information which is required.

Normally the indexing technique is consider a document as a sequence of words, now the problem is simple to map a word to a particular document but in semi structured data this technique is not usable because data is not in structured form. In XML data is stored in a tree like structure that is why indexing technique in XML is different from normal relational database system. So the main goal is to build indexing technique which give fast result with less resource usage.

The remainder of this paper is organized as: section 2 has different indexing techniques, and section 3 has conclusion part.

II. INDEXING TECHNIQUES FOR NATIVES XML DATABASES

I classify the different indexing techniques into three categories: elementary indexes, path look up indexes and navigational indexes. [1]

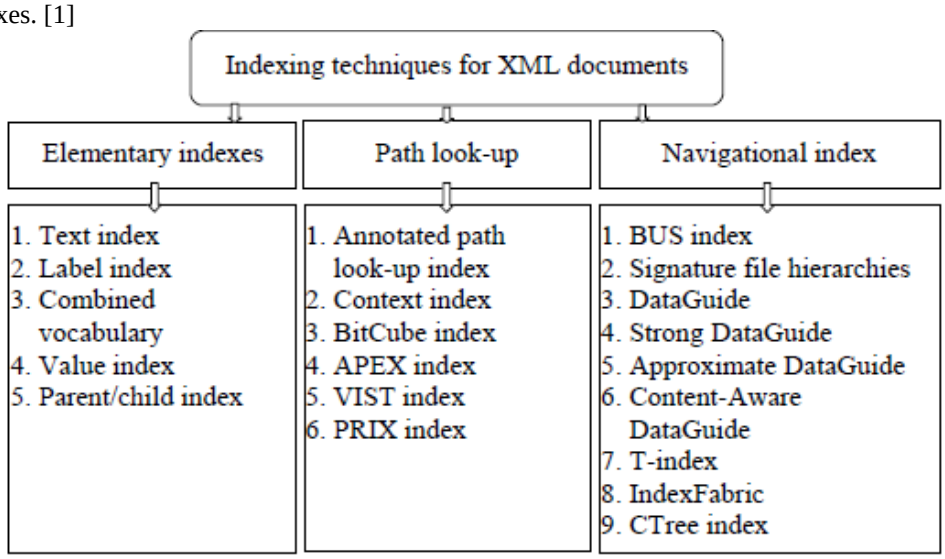

Figure 1: Indexing techniques classification.

2.1 Elementary Indexes

Elementary indexes are represented as tables but with combined vocabulary and they also apply to graphs. They are only concerned with document structure. They are categories into five classes: text index, label index, combined vocabulary index, value index and parent/child index as shown in figure1. One thing which is common in between them is that they all return a set of node id in search result, all the node ids of documents which matches with the selection criteria. The difference between all these indexes is there input parameter. [1]

a. Text index: - It searches on the basis of its textual content.
b. Label index: - It searches on the basis of type of nodes.
c. Combined vocabulary: - It searches on the basis of label and keyword. It returns node ids of documents whose content matches with keyword.
d. Value index: - It searches like text index but add one more parameter called predicate.
e. Parent/child index: - It searches on the basis of node id which it takes as input. It searches all the child and parent nodes of this node id.

2.2 Path look-up indexes

For the research purpose indexes of this class uses path of the entire document for searching instead of using nodes. Query engine rebuild path to the document at the time of evaluation but path look up indexes store path to the document in the indexes, so there is no need for query engines to rebuild the path which is different from the elementary indexes.

2.2.1 Annotated Path Look-Up Index

It gathers two index structures: the Element Locator Scheme [2] and the CIS index [3]. Both of these are used for searching keyword occurrence and they both take the same input but they give different output. The Element Locator Scheme give label path for each node that contains keyword as an output and CIS index gives a path where a label and unique id identifies a node. They both have a single look up table which they used to map keyword to path to document. They can apply to tree based database but not to graphs because there can be multiple paths to reach the node.CIS index require each node to have unique id and element locator scheme uses sibling number to identify a node. CIS index update incrementally by itself. But in element locator scheme if a new node is inserted then the number of nodes sibling is increased by one.

2.2.2 Context Index

It is one of the path look up index similar to CIS index. In this index is used to search keyword in labels path which is given as the input and set of nodes as an output. It has a simple table in which each keyword is connected to a pair of nodes and label path.

2.2.3 BITCUB

Bitcub is a three dimensional bitmap index. It adds third dimension to each document. For XML document bitcube is defined as BITCUBE = (d, p, v, b) where d is document, p is path, v is textual content and b is bit 0 or 1 [1]. It supports six different look up tables. Each operation corresponds to different projections from three dimensional to two dimensional matrixes. It can also support searching of all matching occurrences with keyword by given label path. It can work with tree like database and update whole database incrementally, a new index slice is added when any new keyword or document is added [1].

2.2.4 APEX

APEX stores all the paths from the root. Depth of XML document decides the size of this structure. If depth is high it degrades the performance. APEX (Adaptive path index) does compromise with size and efficiency. It only indexes those paths which are frequently used rather than indexing all the paths from the root. It support graphs, each node in graph has unique id.

2.2.5 VIST

The VIST (Virtual Suffix Tree) index structure is proposed by Wang et al. [1]. It has some features. In VIST index every tree node is stored in (x, y) form where x is label node and y is its path in tree. It also supports some basic operations like insert, delete and update. They use virtual tree to store data in the form of B+ tree. XML data is represented in preorder traversal of this tree.

2.2.6 PRIX

VIST is top down transformation approach and PRIX is bottom up transformation approach. It is an improvement over VIST approach. These sequences are also indexed by virtual tree. PRIX reduces query processing time when bottom up transformation of XML data is used. It is also based on B+ tree. It uses post order numbering to marked node. It also supports delete, insert and modification operations.

2.3 Navigational Indexes

The indexes of this class use directed graphs as their main data structure. The following section has many navigational indexes.

2.3.1 Bus Index

The BUS (Bottom up scheme) [1] index is an approach that rank query result. If a keyword is matched with any document then that document has highest rank. It takes simple path and keyword as input and return a set of (DocId, NodeId, and Weight) in which DocId is document id, NodeId is node id and weight is rank [1]. This index is not better for databases which are changed frequently. It has three data structures scheme tree, content index and occurrence of each keyword

2.3.2 Signature File Hierarchy

Signature file hierarchy combines two things signature file and tries to index document. It discards all nodes which do not contain keyword. It takes keyword and simple path as input and give set of nodes as output. It discards a sub tree that does not contain keyword without any risk. It also handles textual data. As its name suggest it relies on signature that is represented as keyword.

2.3.3 DATAGUIDE

DATAGUIDE supports both tree and graph like database. It takes a simple path as input and return node ids as output. It is a graph where every simple path represented once. It also supports successive update which is costly.

2.3.4 STRONG DATAGUIDE

STRONG DATAGUIDE has two conditions, if a XML data-graph fulfil these two conditions then only it is a strong dataguide. Conditions are as follows:

1. In the source data every distinct root path appears only once in the graph index.
2. There should be no invalid paths in the graph.

Update operation is very simple in strong dataguide just like inserting a leaf in the tree structure.

2.3.5 APPROXIMATE DATAGUIDE

In general size of Strong dataguide is smaller than original database. But in some cases size of Strong dataguide become very large so to overcome this disadvantage Approximate Dataguide comes to picture. ADG does not keep the second requirement of strong dataguide but need first requirement of strong dataguide.

2.3.6 CONTENT-AWARE DATAGUIDE

Two new approaches are introduced by Weigel et al, first one is native content-centric approach in which every value has its separate Dataguide. CADG have one dataguide which refer to all the nodes that contains this value. This approach is only suitable for one key query because it waste lots of memory space. And the second approach is structure centric it has one dataguide and content information. This information is helpful in rejecting irrelevant paths when path expression is processed. Disadvantage of this approach is that it has limited capability for update operation.

2.3.7 CTREE INDEX

CTREE is a two level tree that has a level group which provides an overview of hierarchal structure and contains edge from parent groups to child groups and one element that contains information between levels. CTREE index is constructed in two stages first a tree summary is build and second step is to order the nodes. It takes two inputs one is simple path and other is attribute and return ids of node as output. It is suitable for tree databases and directed acyclic graphs.

2.3.8 T-INDEX

T-INDEX is also known as template index. It was designed to give a effective index structure for semi-structured data. This index used concept of template for frequent queries. There are two special cases of T-index first is 1-index and second is 2-index. 1-index is used for indexing of all absolute paths and 2-index is used to cover relative paths. T-index is good in answering specific queries whereas 1 and 2 indexes are suitable for path expressions.

2.3.9 INDEXFABRIC

INDEXFABRIC takes a simple path in input and return node ids as output. It has been designed to index both structure and content of databases. It can support incremental update operation. INDEXFABRIC is a PATRICIA Trie indexing label combinations and strings. Designator which is a symbol is assigned to each labels document collection. Designator dictionary is used to store the designators.

III. CONCLUSION

As I have discussed earlier that indexing techniques are very important in information retrieval. An information retrieval function depends on the indexing quality for document, because wrong indexing document may not be accessible by user. I have discussed three indexing techniques elementary indexes, path look up indexes and navigational indexes. These categories are different in the way of indexing document structural information.

By studying these approaches I can find out two conclusions: firstly, those approaches that use signatures principal for representing structural information like context index or those who use content as in signature file hierarchy can gain space and processing time. Secondly the use of Tries principal that use in the signature file hierarchy like DataGuide and Ctree can summarise structural relationships of large different types of document collection in very small structure.

REFRENCES

[1] Zemmar, Imen; Benouareth, Abdallah; Souici-Meslati, Labiba, "A SURVEY OF INDEXING TECHNIQUES IN NATIVES XML DATABASES", Naif Arab University for Security Sciences, 2011.

[2] Sacks-Davis R., Arnold-Moore T. and Zobel J., "Database Systems for Structured Documents", *In Proc. Int. Symp. ADT*I, 1994.

[3] Meuss H., "Logical Tree Matching with Complete Answer Aggregates for Retrieving Structured Documents", *Ph.D*, Dept.Computer Science, Univ. Munich, 2000.

[4] Atul D. Raut, Dr. M. Atique, "A Survey of Indexing Techniques for Xml Database", COMPUSOFT, An international journal of advanced computer technology, 3 (1), January-2014 (Volume-III, Issue-I)

[5] www.google.com

[6] www.wikipedia.com

